



SECURITY

Cisco Firewalls

Concepts, design and deployment for
Cisco Stateful Firewall solutions

<https://t.me/learningnets>

ciscopress.com

Alexandre M.S.P. Moraes, CCIE® No. 6063

www.it-ebooks.info

Cisco Firewalls

Alexandre Matos da Silva Pires de Moraes, CCIE No. 6063

Cisco Press

800 East 96th Street

Indianapolis, IN 46240

<https://t.me/learningnets>

www.it-ebooks.info

Cisco Firewalls

Alexandre Matos da Silva Pires de Moraes

Copyright © 2011 Cisco Systems, Inc.

Published by:

Cisco Press

800 East 96th Street

Indianapolis, IN 46240 USA

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the publisher, except for the inclusion of brief quotations in a review.

Printed in the United States of America

First Printing June 2011

Library of Congress Cataloging-in-Publication data is on file.

ISBN-13: 978-1-58714-109-6

ISBN-10: 1-58714-109-4

Warning and Disclaimer

This book is designed to provide information about Cisco Firewall solutions based on IOS and ASA platforms. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an “as is” basis. The authors, Cisco Press, and Cisco Systems, Inc. shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the discs or programs that may accompany it.

The opinions expressed in this book belong to the author and are not necessarily those of Cisco Systems, Inc.

Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Cisco Press or Cisco Systems, Inc., cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Corporate and Government Sales

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact: U.S. Corporate and Government Sales 1-800-382-3419 corpsales@pearsoned.com

For sales outside the United States please contact: International Sales international@pearsoned.com

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Readers' feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through email at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Publisher: Paul Boger

Associate Publisher: Dave Dusthimer

Executive Editor: Brett Bartow

Managing Editor: Sandra Schroeder

Project Editor: Seth Kerney

Editorial Assistant: Vanessa Williams

Book Designer: Sandra Schroeder

Cover Designer: Louisa Adair

Composition: Mark Shirar

Manager Global Certification: Erik Ullanderson

Business Operation Manager, Cisco Press: Anand Sundaram

Development Editor: Ginny Bess Munroe

Copy Editor: Apostrophe Editing Services

Technical Editor: Maurilio de Paula Gorito

Technical Editor: Allan Eduardo Sá Cesarini

Proofreader: Sarah Kearns

Indexer: Brad Herriman



Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems, Inc.
168 Robinson Road
#28-01 Capital Tower
Singapore 068912
www.cisco.com
Tel: +65 6317 7777
Fax: +65 6317 7799

Europe Headquarters
Cisco Systems International BV
Haarlerbergpark
Haarlerbergweg 13-19
1101 CH Amsterdam
The Netherlands
www-europe.cisco.com
Tel: +31 0 800 020 0791
Fax: +31 020 357 1100

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

©2007 Cisco Systems, Inc. All rights reserved. CCVP, the Cisco logo, and the Cisco Square Bridge logo are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, BPX, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, GigaStack, HomeLink, Internet Quotient, IOS, IP/TV, IQ Expertise, the IQ logo, IQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networking Academy, Network Registrar, Packet, PIX, ProConnect, RateMUX, ScriptShare, SlideCast, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (06059)

About the Author

Alexandre Matos da Silva Pires de Moraes, CCIE No. 6063, has worked as a systems engineer for Cisco Brazil since 1998, in projects that involve not only security and VPN technologies but also routing protocol and campus design, IP multicast routing, and MPLS networks design. He has supported large enterprise and public sector accounts and, for almost three years, coordinated a team of Security engineers in Brazil. Alexandre holds the CISSP, CCSP, and 03 CCIE certifications (routing/switching, security, and service provider).

Alexandre, a frequent speaker at Cisco Live, graduated in electronic engineering from the Instituto Tecnológico de Aeronáutica (ITA – Brazil) and has never hidden his sincere passion for mathematics (mainly the fields of synthetic geometry and trigonometry).

Alexandre maintains a personal blog in which he discusses topics related to Networking and Security technologies at <http://alexandremsporaes.wordpress.com/>.

About the Technical Reviewers

Maurilio de Paula Gorito, CCIE No. 3807, is a triple CCIE. He is certified in routing and switching, WAN switching, and security. Maurilio has more than 24 years of experience in networking, including Cisco networks and IBM/SNA environments. Maurilio's experience includes the planning, designing, implementing, and troubleshooting of large IP networks running RIP, IGRP, EIGRP, BGP, OSPF, QoS, and SNA worldwide, including Brazil and the United States. He has more than 10 years of experience in teaching technical classes at schools and companies. Maurilio worked for Cisco as part of the CCIE team as a CCIE lab proctor and program manager. He proctored CCIE Routing & Switching and CCIE Security Lab exams at the CCIE Lab in San Jose, California, United States. As program manager, Maurilio was responsible for managing the content development process for the CCIE Routing & Switching lab and written exams; Maurilio also has presented power sessions at Cisco seminars. Currently, Maurilio works for Riverbed Technology as a certification manager, managing the Riverbed's certification program. He holds degrees in mathematics and pedagogy.

Allan Eduardo Sá Cesarini, CCIE No. 5440, is a double CCIE, having certified in routing and switching in 1999 and in service providers in 2001. Working at Cisco for more than 12 years, and having supported customers ranging from banks, utility providers, government agencies, Enterprise-focused service providers, broadband services, and more recently, cable MSOs, Allan has worked with a myriad of technologies encompassing SNA/IBM, IPX, and IP routing from small-to-large scale networks, campus LAN and ATM networks, IP telephony and voice conferencing solutions, and Docsis-based data services and digital television. Allan is currently working for Cisco Advanced Services, in a consultant capacity, and has presented power sessions at Cisco seminars and Cisco Live events, in areas including LAN architecture, MPLS technology, and security solutions.

Allan holds a degree in computer engineering by the Instituto Tecnológico de Aeronáutica and is currently working on his MBA in enterprise management at Fundação Getúlio Vargas.

Dedications

This book is dedicated to my lovely wife, Rachel, and my wonderful kids, Eduardo and Gustavo, all of them daily acting as true sources of inspiration for my work. Besides their patience and support, I will never forget some of the phrases I heard during the writing process:

By Eduardo (six years old at the time):

“Daddy, is this book more important than your son?”

“Daddy, won’t we ever play chess and soccer again?”

“Daddy, don’t forget saying good night to your book.”

By Gustavo (three years old at the time and more concerned about the color of the Cisco Press book covers):

“Daddy, why isn’t it purple?”

“Daddy, when will you make a green one?”

This book is also dedicated to my mother, Lélia, someone who really set the example for me in terms of reaching goals and not giving up easily.

Finally, I would like to dedicate the book to three teachers who really influenced me and significantly contributed to my development: Seizi Amano, my eternal guru in Mathematics and a true supporter in many of my endeavors. You will never be forgotten, my friend. José Acácio Viana Santos, who taught me that writing is an exercise of reflection and convinced me that this should be deemed a solution rather than a problem. Roberto Stanganelli, for his continuous presence, expressed as lessons of optimism, despite the distance and circumstances.

Acknowledgments

I would like to express my thankfulness to three special friends who shared thoughts and perceptions about the content and approach that could make this book more useful for the readers: Frederico Vasconcelos, Gustavo Santana, and Diego Soares.

Thanks to my great friend Andre Lee for his contributions with the artistic illustrations. What a gift!

Thanks to my friend Jose Furst, Jr., who used only one phrase to convince me that I should write the original in English.

I would like to thank Marcos Yamamoto, Renier Souza, and Renato Pazotto for their support since the early days of the project.

Thanks to the technical reviewers Allan Cesarini and Maurilio Gorito, for their significant help on making this book more accurate.

I would like to thank some individuals in the IOS security group who have helped with some of the AAA or ZFW topics: Nelson Chao, Arshad Saeed, Srinivas Kuruganti, Umanath S. S., and Prashanth Patil.

Thanks to members of the Voice team who somehow contributed to Chapter 13: Christina Hattingh, Pashmeen Mistry, Dan Keller, and Praveen Konda.

Thanks to Andrew Cupp and Ginny Munroe for their help and patience during the review phase.

Thanks to all the Pearson production team, who materialized the final version of this work.

A big thank-you goes out to Brett Bartow for understanding that there was room for a firewall book with a different approach and for actually investing in this project.

Contents at a Glance

	Foreword	xviii
	Introduction	xix
Chapter 1	Firewalls and Network Security	1
Chapter 2	Cisco Firewall Families Overview	27
Chapter 3	Configuration Fundamentals	43
Chapter 4	Learn the Tools. Know the Firewall	89
Chapter 5	Firewalls in the Network Topology	133
Chapter 6	Virtualization in the Firewall World	199
Chapter 7	Through ASA Without NAT	247
Chapter 8	Through ASA Using NAT	287
Chapter 9	Classic IOS Firewall Overview	323
Chapter 10	IOS Zone Policy Firewall Overview	361
Chapter 11	Additional Protection Mechanisms	415
Chapter 12	Application Inspection	473
Chapter 13	Inspection of Voice Protocols	547
Chapter 14	Identity on Cisco Firewalls	617
Chapter 15	Firewalls and IP Multicast	669
Chapter 16	Cisco Firewalls and IPv6	715
Chapter 17	Firewall Interactions	787
Appendix	NAT and ACL Changes in ASA 8.3	849
Index		869

Contents

Foreword xviii

Introduction xix

Chapter 1 Firewalls and Network Security 1

Security Is a Must. But, Where to Start? 2

Firewalls and Domains of Trust 5

Firewall Insertion in the Network Topology 6

 Routed Mode Versus Transparent Mode 7

 Network Address Translation and Port Address Translation 8

Main Categories of Network Firewalls 10

 Packet Filters 10

 Circuit-Level Proxies 11

 Application-Level Proxies 12

 Stateful Firewalls 13

The Evolution of Stateful Firewalls 14

 Application Awareness 14

 Identity Awareness 15

 Leveraging the Routing Table for Protection Tasks 16

 Virtual Firewalls and Network Segmentation 17

What Type of Stateful Firewall? 18

 Firewall Appliances 18

 Router-Based Firewalls 18

 Switch-Based Firewalls 20

Classic Topologies Using Stateful Firewalls 20

Stateful Firewalls and Security Design 21

 Stateful Firewalls and VPNs 22

 Stateful Firewalls and Intrusion Prevention 23

 Stateful Firewalls and Specialized Security Appliances 25

Summary 26

Chapter 2 Cisco Firewall Families Overview 27

Overview of ASA Appliances 27

 Positioning of ASA Appliances 28

 Firewall Performance Parameters 29

 Overview of ASA Hardware Models 32

Overview of the Firewall Services Module 36

Overview of IOS-Based Integrated Firewalls	38
Integrated Services Routers	38
Aggregation Services Routers	39
Summary	41

Chapter 3 Configuration Fundamentals 43

Device Access Using the CLI	44
Basic ASA Configuration	44
Basic Configuration for ASA Appliances Other Than 5505	49
Basic Configuration for the ASA 5505 Appliance	52
Basic FWSM Configuration	55
Remote Management Access to ASA and FWSM	60
Telnet Access	61
SSH Access	62
HTTPS Access Using ASDM	63
IOS Baseline Configuration	67
Configuring Interfaces on IOS Routers	69
Remote Management Access to IOS Devices	70
Remote Access Using Telnet	70
Remote Access Using SSH	71
Remote Access Using HTTP and HTTPS	73
Clock Synchronization Using NTP	74
Obtaining an IP Address Through the PPPoE Client	77
DHCP Services	82
Summary	86
Further Reading	87

Chapter 4 Learn the Tools. Know the Firewall 89

Using Access Control Lists Beyond Packet Filtering	90
Event Logging	92
Debug Commands	97
Flow Accounting and Other Usages of Netflow	98
Enabling Flow Collection on IOS	100
Traditional Netflow	100
Netflow v9 and Flexible Netflow	105
Enabling NSEL on an ASA Appliance	112
Performance Monitoring Using ASDM	114
Correlation Between Graphical Interfaces and CLI	115

Packet Tracer on ASA	119
Packet Capture	122
Embedded Packet Capture on an ASA Appliance	123
Embedded Packet Capture on IOS	128
Summary	130

Chapter 5 Firewalls in the Network Topology 133

Introduction to IP Routing and Forwarding	134
Static Routing Overview	135
Basic Concepts of Routing Protocols	138
RIP Overview	140
Configuring and Monitoring RIP	142
EIGRP Overview	150
Configuring and Monitoring EIGRP	152
<i>EIGRP Configuration Fundamentals</i>	152
<i>Understanding EIGRP Metrics</i>	154
<i>Redistributing Routes into EIGRP</i>	158
<i>Generating a Summary EIGRP Route</i>	161
<i>Limiting Incoming Updates with a Distribute-List</i>	162
<i>EIGRP QUERY and REPLY Messages</i>	162
<i>EIGRP Stub Operation</i>	164
OSPF Overview	167
Configuring and Monitoring OSPF	169
<i>OSPF Configuration Fundamentals</i>	170
<i>OSPF Scenario with Two Areas</i>	177
Configuring Authentication for Routing Protocols	187
Bridged Operation	190
Summary	198

Chapter 6 Virtualization in the Firewall World 199

Some Initial Definitions	200
Starting with the Data Plane: VLANs and VRFs	201
Virtual LANs	201
VRFs	202
VRF-Aware Services	212
Beyond the Data Plane—Virtual Contexts	212
Management Access to Virtual Contexts	225
Allocating Resources to Virtual Contexts	228

Interconnecting Virtual Elements	232
Interconnecting VRFs with an External Router	232
Interconnecting Two Virtual Contexts That Do Not Share Any Interface	233
Interconnecting Two FWSM Contexts That Share an Interface	234
Interconnecting Two ASA Contexts That Share an Interface	238
Issues Associated with Security Contexts	241
Complete Architecture for Virtualization	242
Virtualized FWSM and ACE Modules	242
Segmented Transport	244
Virtual Machines and the Nexus 1000V	245
Summary	246

Chapter 7 Through ASA Without NAT 247

Types of Access Through ASA-Based Firewalls	248
Additional Thoughts About Security Levels	253
Internet Access Firewall Topology	254
Extranet Topology	254
Isolating Internal Departments	254
ICMP Connection Examples	254
Outbound Ping	255
Inbound Ping	257
Windows Traceroute Through ASA	258
UDP Connection Examples	260
Outbound IOS Traceroute Through ASA	261
TCP Connection Examples	265
ASA Flags Associated with TCP Connections	265
TCP Sequence Number Randomization	267
Same Security Access	272
Handling ACLs and Object-Groups	274
Summary	285

Chapter 8 Through ASA Using NAT 287

Nat-Control Model	288
Outbound NAT Analysis	290
Dynamic NAT	291
Dynamic PAT	293
Identity NAT	296

Static NAT	298
Policy NAT	299
<i>Static Policy NAT</i>	299
<i>Dynamic Policy NAT</i>	301
<i>Dynamic Policy PAT</i>	302
NAT Exemption	303
NAT Precedence Rules	304
Address Publishing for Inbound Access	308
Publishing with the static Command	308
Publishing with Port Redirection	309
Publishing with NAT Exemption	310
Inbound NAT Analysis	311
Dynamic PAT for Inbound	311
Identity NAT for Inbound	313
NAT Exemption for Inbound	314
Static NAT for Inbound	314
Dual NAT	315
Disabling TCP Sequence Number Randomization	317
Defining Connection Limits with NAT Rules	318
Summary	321

Chapter 9 Classic IOS Firewall Overview 323

Motivations for CBAC	324
CBAC Basics	325
ICMP Connection Examples	328
UDP Connection Examples	331
TCP Connection Examples	334
Handling ACLs and Object-Groups	338
Using Object-Groups with ACLs	340
CBAC and Access Control Lists	342
IOS NAT Review	343
Static NAT	346
Dynamic NAT	349
Policy NAT	350
Dual NAT	351
NAT and Flow Accounting	353
CBAC and NAT	355
Summary	360

Chapter 10 IOS Zone Policy Firewall Overview 361

Motivations for the ZFW	362
Building Blocks for Zone-Based Firewall Policies	365
ICMP Connection Examples	370
UDP Connection Examples	373
TCP Connection Examples	377
ZFW and ACLs	379
ZFW and NAT	391
ZFW in Transparent Mode	400
Defining Connection Limits	403
Inspection of Router Traffic	407
Intrazone Firewall Policies in IOS 15.X	410
Summary	414

Chapter 11 Additional Protection Mechanisms 415

Antispoofing	416
Classic Antispoofing Using ACLs	416
Antispoofing with uRPF on IOS	417
Antispoofing with uRPF on ASA	420
TCP Flags Filtering	425
Filtering on the TTL Value	429
Handling IP Options	430
Stateless Filtering of IP Options on IOS	434
IP Options Drop on IOS	437
IP Options Drop on ASA	438
Dealing with IP Fragmentation	439
Stateless Filtering of IP Fragments in IOS	443
Virtual Fragment Reassembly on IOS	445
Virtual Fragment Reassembly on ASA	446
Flexible Packet Matching	448
Time-Based ACLs	453
Time-Based ACLs on ASA	454
Time-Based ACLs on IOS	457
Connection Limits on ASA	458
TCP Normalization on ASA	463
Threat Detection on ASA	466
Summary	470
Further Reading	471

Chapter 12 Application Inspection 473

- Inspection Capabilities in the Classic IOS Firewall 474
- Application Inspection in the Zone Policy Firewall 478
- DNS Inspection in the Zone Policy Firewall 479
- FTP Inspection in the Zone Policy Firewall 481
- HTTP Inspection in the Zone Policy Firewall 487
- IM Inspection in the Zone Policy Firewall 494
- Overview of ASA Application Inspection 496
- DNS Inspection in ASA 500
 - DNS Guard 502
 - DNS Doctoring 505
 - DNS Inspection Parameters 508
 - Some Additional DNS Inspection Capabilities 511
- FTP Inspection in ASA 512
- HTTP Inspection in ASA 525
- Inspection of IM and Tunneling Traffic in ASA 534
- Botnet Traffic Filtering in ASA 537
- Summary 544
- Further Reading 545

Chapter 13 Inspection of Voice Protocols 547

- Introduction to Voice Terminology 548
- Skinnny Protocol 550
- H.323 Framework 560
 - H.323 Direct Calls 563
 - H.323 Calls Through a Gatekeeper 567
- Session Initiation Protocol (SIP) 573
- MGCP Protocol 584
- Cisco IP Phones and Digital Certificates 593
- Advanced Voice Inspection with ASA TLS-Proxy 596
- Advanced Voice Inspection with ASA Phone-Proxy 603
- Summary 616
- Further Reading 616

Chapter 14 Identity on Cisco Firewalls 617

- Selecting the Authentication Protocol 620
- ASA User-Level Control with Cut-Through Proxy 621
 - Cut-Through Proxy Usage Scenarios 622
 - Scenario 1: Simple Cut-Through Proxy (No Authorization)* 624
 - Scenario 2: Cut-Through Proxy with Downloadable ACEs* 625
 - Scenario 3: Cut-Through Proxy with Locally Defined ACL* 627

Scenario 4: Cut-Through Proxy with Downloadable ACLs 629

Scenario 5 - HTTP Listener 632

IOS User-Level Control with Auth-Proxy 634

Scenario 1: IOS Auth-Proxy with Downloadable Access Control
Entries 638

Scenario 2: IOS Auth-Proxy with Downloadable ACLs 639

Scenario 3: Combining Classic IP Inspection (CBAC) and Auth-Proxy 642

User-Based Zone Policy Firewall 645

Establishing user-group Membership Awareness in IOS - Method 1 645

Establishing user-group Membership Awareness in IOS - Method 2 646

Integrating Auth-Proxy and the ZFW 650

Administrative Access Control on IOS 654

Administrative Access Control on ASA 662

Summary 666

Chapter 15 Firewalls and IP Multicast 669

Review of Multicast Addressing 670

Overview of Multicast Routing and Forwarding 671

The Concept of Upstream and Downstream Interfaces 672

RPF Interfaces and the RPF Check 674

Multicast Routing with PIM 676

Enabling PIM on Cisco Routers 677

PIM-DM Basics 678

PIM-SM Basics 680

Finding the Rendezvous Point on PIM-SM Topologies 690

Inserting ASA in a Multicast Routing Environment 697

Enabling Multicast Routing in ASA 698

Stub Multicast Routing in ASA 702

ASA Acting as a PIM-SM Router 707

Summary of Multicast Forwarding Rules on ASA 712

Summary 714

Further Reading 714

Chapter 16 Cisco Firewalls and IPv6 715

Introduction to IPv6 716

Overview of IPv6 Addressing 717

IPv6 Header Format 722

IPv6 Connectivity Basics 724

Handling IOS IPv6 Access Control Lists 743

IPv6 Support in the Classic IOS Firewall 751

IPv6 Support in the Zone Policy Firewall	757
Handling ASA IPv6 ACLs and Object-Groups	766
Stateful Inspection of IPv6 in ASA	770
Establishing Connection Limits	774
Setting an Upper Bound for Connections Through ASA	774
IPv6 and Antispoofing	776
Antispoofing with uRPF on ASA	776
Antispoofing with uRPF on IOS	776
IPv6 and Fragmentation	778
Virtual Fragment Reassembly on ASA	783
Virtual Fragment Reassembly on IOS	783
Summary	785
Further Reading	785

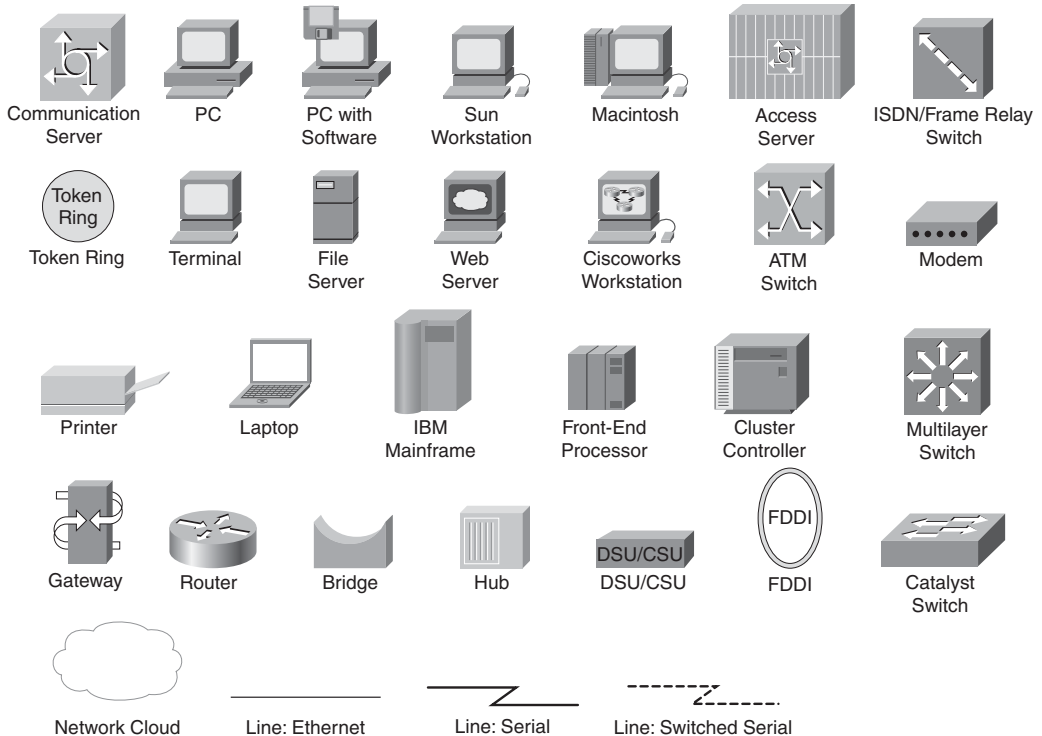
Chapter 17 Firewall Interactions 787

Firewalls and Intrusion Prevention Systems	788
Firewalls and Quality of Service	793
Firewalls and Private VLANs	794
Firewalls and Server Load Balancing	796
Firewalls and Virtual Machines	801
Protecting Virtual Machines with External Firewalls	802
Protecting Virtual Machines Using Virtual Firewall Appliances	803
Firewalls and IPv6 Tunneling Mechanisms	806
Firewalls and IPsec VPNs	812
Classic IPsec Site-to-Site for IOS	813
IPsec Site-to-Site Using a Virtual Tunnel Interface (VTI)	818
IPsec Site-to-Site Using a GRE Tunnel	822
NAT in the Middle of an IPsec Tunnel	823
Post-Decryption Filtering in ASA	826
Firewalls and SSL VPNs	828
Clientless Access	829
Client-Based Access (AnyConnect)	836
Firewalls and MPLS Networks	841
Borderless Networks Vision	845
Summary	848
Further Reading	848

Appendix NAT and ACL Changes in ASA 8.3 849

Index 869

Icons Used in This Book



Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in the IOS Command Reference. The Command Reference describes these conventions as follows:

- **Boldface** indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that are manually input by the user (such as a **show** command).
- *Italic* indicates arguments for which you supply actual values.
- Vertical bars (|) separate alternative, mutually exclusive elements.
- Square brackets ([]) indicate an optional element.
- Braces ({ }) indicate a required choice.
- Braces within brackets ([{ }]) indicate a required choice within an optional element.

Foreword

Networks today have outgrown exponentially both in size and complexity, becoming more multifaceted and increasingly challenging to secure. The blueprint of a core network requires a strong foundation, which can be simply provided with an integrated firewall architecture cemented at the core of the system. Today, the firewall has become a core entity within a network and an integral part of every network infrastructure.

Cisco Firewalls, by Alexandre M. S. P. Moraes, has taken a stab at unleashing some of the fundamentally missed concepts, providing readers with a complete library of the entire family of Cisco Firewall products in a single [book].

Alexandre has used a unique approach in explaining the concepts and architecture of the firewall technology. His distinct style has proven his skill at writing on a difficult subject using easy-to-understand illustrations that walk the reader through a step-by-step approach that shows the *theory in action*. He has combined some of the commonly used tools with the outputs from several commands to demonstrate the understanding of the technology and exemplifying *how it works*.

Cisco Firewalls is unlike any other book on this subject and cannot be categorized as a configuration guide or command syntax manual. It provides the readers with the key tools and essential techniques to understand the wide-ranging Cisco Firewall portfolio. Whether you are just a beginner trying to learn [about] Cisco Firewalls or an experienced engineer looking for a reference, there is something for everyone in this book at varying levels.

Cisco Firewalls is an essential reference in designing, implementing, and maintaining today's highly secured networks. It is a must read and a must have in your collection—Magnum Opus!

—Yusuf Bhajji

Sr. Manager, Expert Certifications (CCIE, CCDE, CCAr)

Introduction

Firewalls have ample recognition as key elements in the field of protecting networks. Even though this is not a new subject, many important concepts and resources that could be helpful to designing a secure network are often overlooked or even ignored.

This book is targeted at unveiling the potential of Cisco Firewall functionalities and products and how they can be grouped on a structured manner to build security solutions.

The motivation for writing this book is associated with a simple axiom assumed: The better you understand individual features, the better you can use them for design purposes. After all, producing better security designs is the aim of anyone truly committed to security.

“Happy is he who transfers what he knows and learns what he teaches.”—Cora Coralina, Brazilian poet

Goals and Methods

Typical firewall books are developed around two distinct philosophies:

- Configuration guides and handbooks focus on the set of commands to put a certain feature in place. These books have their importance but normally ignore the discussion of the value of each functionality and the motivation for use of a certain feature, and they do not contribute that much to building knowledge of the power of specific resources.
- There are the conceptual-only books that mainly talk about categories of firewalls in a more generic fashion, not paying particular attention to the materialization of the “functionalities on specific platforms” and not establishing the connection between theoretical and practical worlds.

Linking theory and practice aids in the understanding of main concepts and significantly contributes to the production of better designs. This perception comes from a mathematics and engineering background: investing time in learning theory and, in understanding how to derive the fundamental theorems, is key for succeeding in actual problem solving.

It is also worth mentioning that troubleshooting is frequently relegated to an appendix, in a position totally disconnected from the main text. This book proposes a completely different approach. The tools historically used for troubleshooting are employed in this book to illustrate how firewall features operate, thus establishing the linkages between theory and practice. After becoming familiar with these tools, you will consistently revisit them to reinforce the theoretical concepts presented in each chapter. This not only helps with the learning process but also contributes to avoid an eventual troubleshooting stage in your practical deployments.

Who Should Read This Book?

This book talks about firewall functionalities available on Cisco products and security design from the standpoint of the firewall devices. From beginners to seasoned engineers, there is useful content for everyone interested in the subject of Cisco Firewalls. The target audiences are summarized as the following:

- Security engineers and architects who design and implement firewall solutions
- Security administrators and operators who want to get a thorough understanding of the functionalities they are in charge of deploying
- Professional Services engineers and TAC engineers who need to support Cisco Network Firewalls
- People preparing for certifications in the Cisco security curriculum (CCNA Security, CCNP Security, and the Security CCIE exam)

Although this book contains a lot of configuration-related content, it by no means aims to be a configuration guide. It privileges the understanding of functionalities behavior and the best ways to use firewall features, be it individually or integrated on security design.

How This Book Is Organized

This book can be read cover-to-cover or moving between chapters. There are some ASA-centric and IOS-specific chapters, but overall the chapters deal with both families at the same time. One of the benefits of this approach is the possibility of easily contrasting the resources available on each family and selecting the implementation that best fits your needs. Another advantage is that the theoretical concepts are covered only once (instead of being repeated for each platform):

- **Chapter 1, “Firewalls and Network Security.”** After reviewing the importance of a high-level security policy, this chapter presents the classic types of network firewalls and the possibilities for their insertion in a network environment. The discussion then centers on stateful firewalls and how they have evolved to adapt to the demands of complex environments.
- **Chapter 2, “Cisco Firewall Families Overview.”** This chapter is aimed at presenting an overview of Cisco hardware platforms that host stateful firewall solutions. An important discussion about the performance parameters that needs to be taken into account when selecting a firewall solution is also presented.
- **Chapter 3, “Configuration Fundamentals.”** This chapter presents the initial configuration tasks for the Cisco Firewall families. Topics such as access via the command-line interface (CLI), boot process, IP addressing options, and remote management methods are covered. If you are an experienced user of Cisco devices, you can skip this chapter.

- **Chapter 4, “Learn the Tools. Know the Firewall.”** This chapter is the cornerstone for the approach adopted in this book and, therefore, highly recommended even for advanced readers. The set of tools presented are used throughout the book to detail the operations of Cisco Firewalls and provide the linkages between theory and practice.
- **Chapter 5, “Firewalls in the Network Topology.”** Before providing the security services they are designed for, firewalls need to be inserted in the network topology either using a Layer 3 or a Layer 2 connectivity model. This chapter covers bridging, static routing, and relevant concepts about dynamic routing protocols such as OSPF, EIGRP, and RIP. The way in which the chapter is organized makes it a useful reference for those security focused professionals that are not so familiar with the deployment of routing and bridging solutions.
- **Chapter 6, “Virtualization in the Firewall World.”** This chapter examines the typical meanings of virtualization in the networking arena and how some building blocks (VLANs, VRFs, virtual contexts, and the like) can be combined to deliver a robust and secure virtualization architecture.
- **Chapter 7, “Through ASA Without NAT.”** This ASA-centric chapter starts the actual discussion about security features. Important concepts such as security levels, connection setup and teardown, handling of ACLs, and object-groups are presented and largely exemplified.
- **Chapter 8, “Through ASA Using NAT.”** This chapter is the natural follow-up to Chapter 7, because it details Network Address Translation (NAT) concepts and illustrates the various NAT options for ASA-based firewalls. The often confused topic of NAT precedence rules is carefully analyzed. Chapters 7 and 8 are later complemented by Appendix A, “NAT and ACL Changes in ASA 8.3.”
- **Chapter 9, “Classic IOS Firewall Overview.”** This chapter covers the IOS Context Based Access Control (CBAC) feature set, which is now known as the Classic IOS Firewall. Other important topics such as NAT, ACL, and object-group handling are introduced and exemplified for IOS-based devices.
- **Chapter 10, “IOS Zone Policy Firewall Overview.”** This chapter introduces the Zone Policy Firewall (ZFW), the main option for Cisco IOS-based Firewall deployments. The chapter presents the building blocks for ZFW policy construction and is centered on security functionality that goes up to Layer 4 (generic inspection).
- **Chapter 11, “Additional Protection Mechanisms.”** This chapter focuses on protection resources that act up to Layer 4 and can add significant value to stateful inspection functionality. Features such as antispoofing, TCP normalization, connection limiting, and IP fragmentation handling are covered.

- **Chapter 12, “Application Inspection.”** This chapter presents the application-layer inspection capabilities for all the families of Cisco network Firewalls. This type of functionality is employed by Cisco Firewalls to adapt to the particularities of application protocols that are not well behaved when crossing stateless packet filters or stateful firewalls that are limited to Layer 4. This application knowledge may also be directed to more sophisticated filtering activities.
- **Chapter 13, “Inspection of Voice Protocols.”** This chapter builds upon the application inspection knowledge introduced in Chapter 12 to promote a detailed analysis of classic IP telephony protocols such as SCCP, H.323, SIP, and MGCP. The chapter goes a bit further by analyzing advanced ASA functionality (TLS-proxy and Phone-proxy) that permit the use of voice confidentiality solutions without losing the benefits of stateful inspection. For those security professionals who are not familiar with IP telephony terminology, this chapter can provide a good starting point.
- **Chapter 14, “Identity on Cisco Firewalls.”** This chapter analyzes how the concept of identity can be leveraged to produce user-based stateful functionality in all the Cisco Firewall families. The chapter also discusses the AAA architecture and contrasts the RADIUS and TACACS+ protocols, clearly establishing which one is more suitable for each type of task: controlling access through the firewall or to the firewall (administrative access control).
- **Chapter 15, “Firewalls and IP Multicast.”** This chapter introduces important theoretical aspects pertaining to IP multicast routing and forwarding tasks and later details how multicast traffic is handled through firewalls. The chapter was conceived to serve as a useful reference for readers who are not familiar with the topic.
- **Chapter 16, “Cisco Firewalls and IPv6.”** As the available IPv4 addresses deplete, a careful look at the next-generation Internet Protocol (IP version 6) becomes more compelling. The chapter introduces important IPv6 concepts and presents the IPv6 security features that exist on Cisco Firewall families.
- **Chapter 17, “Firewall Interactions.”** This chapter is centered on security design. Information about the typical interactions of firewall functionality with other features (or systems) that may add value to the overall security practice is presented. In some cases, the definition of “interaction” has more to do with the challenges that should be taken into account when deploying firewalls in some specific environments.
- **Appendix A, “NAT and ACL Changes in ASA 8.3.”** This appendix is aimed at highlighting the changes in the NAT deployment models introduced by ASA 8.3 and, in this sense, is a natural companion of Chapter 8. The new possibility of defining global ACLs is also covered.

Firewalls and Network Security

This chapter covers the following topics:

- Security is a must. But where to start?
- Firewalls and domains of trust
- Firewall insertion in the network topology
- Main categories of network firewalls
- The evolution of stateful firewalls
- What type of stateful firewall?
- Classic topologies using stateful firewalls
- Stateful firewalls and security design

“In preparing for the battle I have always found that plans are useless, but planning is indispensable”—Dwight D. Eisenhower

Voice, Video, Web 2.0, mobility, content, speed, virtualization, cloud....

The reliance of corporations on the services provided by Intelligent Networks rises every day. The flexibility of simultaneously transporting data, voice, and video, and the capability of rapidly deploying new business applications are key factors for customer productivity and success. Networks are now perceived as a true business asset.

But you cannot talk about an *Intelligent Network* without mentioning security.

New security products are frequently proposed in the market place, promising innovations to deal with threats, actual attacks, and management tasks. Although some of these offerings might sound quite appealing, it is advisable not to forget that the effectiveness of the defense is deeply associated with the existence and continuous maintenance of a high-level security policy, reflecting an organization's vision, mission, and objectives.

Cisco has a holistic vision of what security means and proposes a layered defense system, in which each component plays its role and collaborates with the other elements that are part of the network. This approach is totally different from those that simply rely on point products and promote *magic black boxes that solve all problems*. And in many of the points and layers of this system, there are firewalls.

Firewalls are the central element when implementing any network security project. They are in charge of establishing the basic reference on the network topology, separating the trusted zones from the untrusted and enforcing access control rules between them.

The underlying goal of this book is to provide you with a clear understanding of the available Cisco Firewall features, products, and solutions and how they can add value to security, not only on an individual basis but also in terms of security design and operations. After all, *the whole solution should be more valuable than the mere sum of the parts*.

Security Is a Must. But, Where to Start?

The impatient reader's perspective: *Well, this is a firewall book and of course it is time to skip these introductions and start configuring some firewall rules....*

"There is a time for everything, and a season for every activity." Time to design and time to implement. Time to monitor and time to test. Time to manage and time to improve. Time to reflect about all you have done and admit for a while that, no matter what the previous efforts have been, there might be new challenges and risks that should be taken care of.

Although there is no news in what I say, it must be stated clearly and fearlessly: *Start with a security policy*.

The organizational security policy is a high-level document that sets the foundation for all security-related initiatives and activities in the organization. It should be conceived at (and supported by) the executive level and always take into account the vision, mission, and objectives of the organization.

Figure 1-1 summarizes information regarding the high-level security policy and its relationship with some other fundamental components. Among the typical inputs that guide policy creation, some deserve special attention:

- Business objectives are the main drivers of policy definition.
- Regulatory requirements specific to the industry segment in which the organization is inserted must be taken into account.
- After performing a careful risk analysis, the acceptable level of risk from senior management's standpoint should be documented in the policy.
- The selection of countermeasures (for risk mitigation) should always be guided by a comparison between their cost and the value of assets being protected.

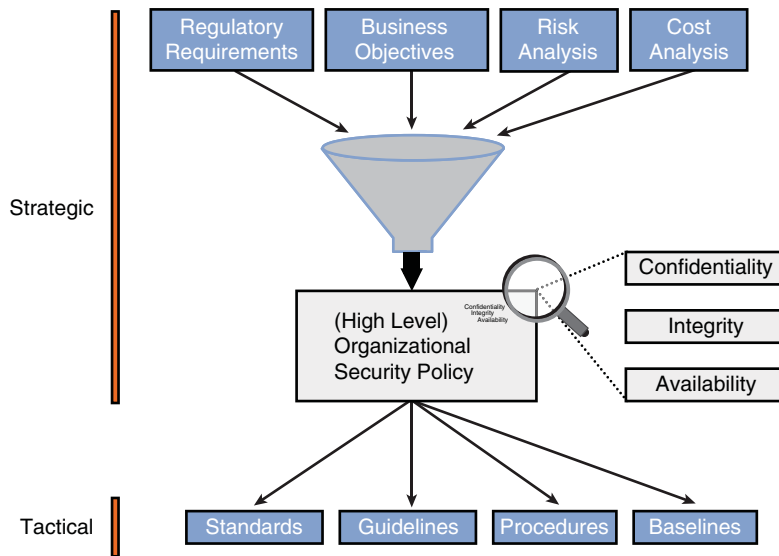


Figure 1-1 *Security Policy*

Some security principles (and how they are dealt with) should permeate policy definitions. These include the following:

- **Confidentiality:** This is concerned with preventing unauthorized disclosure of sensitive information and ensuring that the adequate level of secrecy is enforced at all phases of data processing. Encryption is the classic example of technology aimed at providing confidentiality.
- **Integrity:** This focuses on preventing unauthorized modification of data and ensuring that information is accurate. Hash Message Authentication Codes, such as HMAC-MD5 and HMAC-SHA (widely employed in IPsec), are examples of keyed hash functions designed to provide integrity to transmitted data.
- **Availability:** The observance of this principle ensures reliability and an acceptable level of performance when authorized users request access to resources.

Security policies remain at the strategic level and belong to that category of documents written using broad terms. To have a practical effect, however, the general rules and principles it describes need to be materialized somehow. You can accomplish this by the set of supporting documents (tactical in essence) shown in Figure 1-1 and described in the following:

- **Standards:** Specify *mandatory* rules, regulations, or activities. For instance, there can be an internal standard establishing that all traffic transported through the WAN should be encrypted using a certain cryptographic algorithm.
- **Guidelines:** Provide recommendations, reference actions, and operational guides for users under circumstances to which standards do not apply.

- **Procedures:** Provide detailed step-by-step instructions for performing specific tasks. Procedures define how policies, standards, and guidelines are implemented within the operating environment.
- **Baselines:** Typically define the minimum level of security required for a given system type. For example, a list of unnecessary network services that should be disabled on every router (for hardening purposes) provides a baseline of protection. Other configuration actions are needed depending on the specific uses of that device.

Figure 1-2 depicts the *Security Wheel*, a model for security operations built around the concept of a security policy and that recognizes that security is a continuous and cyclical process. This model consists of five steps:

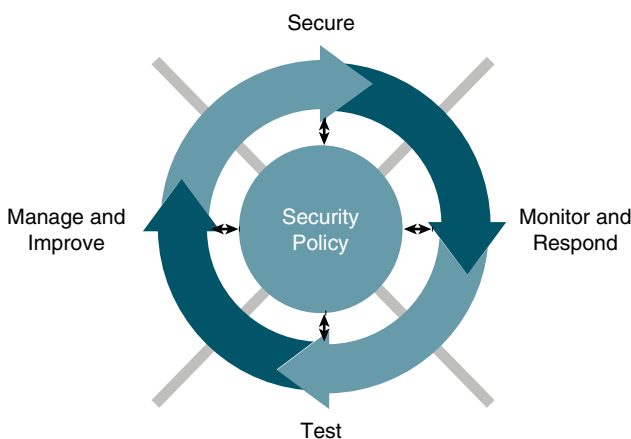


Figure 1-2 *The Security Wheel*

- Step 1. Develop a security policy:** Start with a high-level policy that defines and documents the strategic goals. This document should contain the pointers to the appropriate standards, guidelines, procedures, and baselines that guide and guarantee effective implementation.
- Step 2. Implement protection measures:** Having defined what needs to be protected and the extent of protection provided, it is necessary to implement security solutions that cannot mitigate the risks. Firewalls, encryption technologies, and authentication are sample components of a network security solution.
- Step 3. Monitor and respond:** In this phase, tools such as intrusion detection and logging unveil eventual violations to the security policy.
- Step 4. Test:** The efficacy of the implementation should be verified. Vulnerability scanning and system audits are examples of tools to be used in this phase.
- Step 5. Manage and improve:** Feedback from stages 3 and 4 should be seriously taken into consideration so that improvements to stage 2 can be incorporated. The identification of new vulnerabilities and the evaluation of the risk they pose to the organization should be drivers for improving the security policy.

Note There are plenty of other models describing security operations. The *Security Wheel* not only recognizes the importance of security policy concepts but also proposes a humble approach: It continuously reminds you that the wheel keeps turning and, what seemed to be secure in one particular instant, might prove ineffective (or at least insufficient) in a later one.

Firewalls and Domains of Trust

Before starting the discussion about the firewalls, you need to revisit two fundamental concepts:

- A *computer network* is a collection of autonomous computing devices sharing a data communications technology that enables them to exchange information.
- An *internetwork* is a set of individual networks, interconnected by the appropriate devices, in such a way that they can behave as a single larger network.

These initial definitions are built around the ideas of *providing connectivity* and making *information exchange* possible, two goals achieved with undoubted success. Just think about your daily tasks, and you can see that the dependence on the services delivered by networks (and internetworks) does not cease to grow. Internetworking (particularly the global Internet) has definitely changed the way people live, learn, play, and work.

This is poetic and appealing, but it is always important to remember that the Internet also brings to the scene dangerous features such as anonymity, the ability to remotely control computers, and automated task execution. And, as in other domains of human life, the same resource might be used for good and evil. (Mainly when such a resource provides global reach, 24 hours a day, and offers virtually infinite possibilities of generating profit.)

Will the Internet be used for the purposes of wrongdoing? (*Of course it will....*) It is already hard to know what is on someone else's mind, let alone billions of users spread all around the globe.

There should be some means to compensate for the absence of natural boundaries in the Internet. Ways should exist to define (and enforce) conditions of use instead of liberally granting connectivity and relying on other people's goodwill. A reference must be set, establishing what is acceptable and what is not. And this answer should be provided within the context of each organization.

Within the realm of computer networks, a firewall is a security system that lends itself to the task of isolating areas of the network and delimiting *domains of trust*. Building upon this initial state of isolation created by the firewall, access control policies that specify the traffic types entitled to go from one domain to another can be defined.

The firewall acts as a sort of *conditional gateway*. The criteria to permit traffic are normally defined in the *firewall policy* and, ideally, should relate to (and help on materializing) what is stated in the security policy of the organization.

Figure 1-3 depicts a simple scenario in which there is a firewall controlling access from clients on the trusted domain to servers on the untrusted domain:

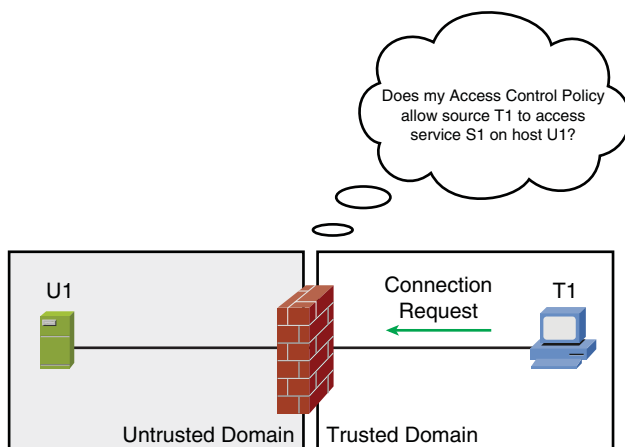


Figure 1-3 *Firewall and Security Domains*

- The enforced conditions corresponding to the question “*Does my access control policy allow...?*” depend on the specific category of the firewall in place. This is the subject of a later section in this chapter.
- Each domain of trust can include one or more networks.
- A firewall is only capable of controlling traffic that *passes through* it. This implies that clear knowledge about the location of clients and servers in the network is needed before beginning policy definition.

Before moving to the next section, following are a few words about *enforcement*, an important concept concerning security.

Life in society is a sequence of vows of confidence. For instance, when driving your car, you believe that if the traffic light is green for you, it will be red for the perpendicular street. (And you also assume that the drivers on that street understand and respect the red sign). This is a well-known convention, and there are rules stating this is the right thing to do to avoid collisions. But there is no physical blockage there.... There is no enforcement.

What about corporate networks used to run the business? Would you give a vow of confidence for any user requesting access to networked systems?

Firewall Insertion in the Network Topology

To enforce access control policies between domains of trust, firewalls first need to be inserted into the network topology. The following sections examine the two basic forms of promoting this insertion: Routed mode and Transparent mode.

Routed Mode Versus Transparent Mode

Although a firewall can be simultaneously connected to multiple domains (with possibly many interfaces within each domain), two interfaces are usually sufficient for the analysis of the main concepts. Figure 1-4 depicts the two basic forms of connecting firewalls to network environments:

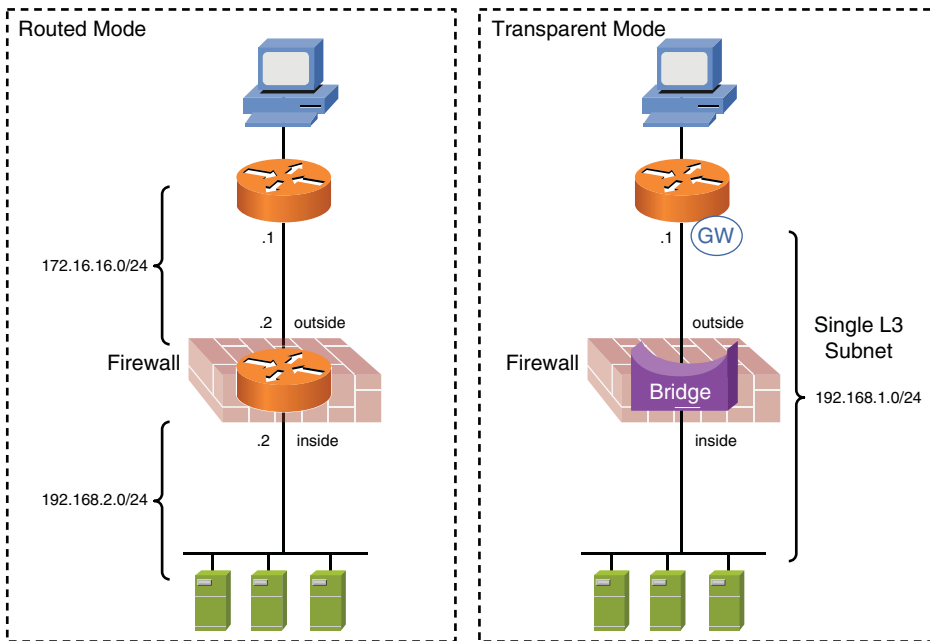


Figure 1-4 *Routed Mode Versus Transparent Mode.*

- **Routed mode:** The firewall works as a Layer 3 element (such as a router) from the perspective of hosts connecting to it. Each of its interfaces is assigned to a different logical subnet and the packets are (conditionally) routed between them. For instance, interface1 (inside) has the IP address 192.168.2.2, and interface2 (outside) uses the address 172.16.16.2. Because the hosts are interconnected by the firewall, machines on the inside need to configure the address 192.168.2.2 as their L3 gateway to reach outside destinations.
- **Transparent mode:** The firewall acts as a conditional (transparent) bridge, forwarding frames between interfaces based on Layer 2 information. In this case, the two interfaces represented in the figure connect to the same L3 subnet and the inside hosts use the external router (192.168.2.1) as their gateway to reach outside destinations.

Although Routed mode is the most well-known and widespread firewall placement, Transparent mode is a convenient option in scenarios in which minimal network reconfiguration is a premise. Detailed analysis of these firewall connectivity aspects is the subject of Chapter 5, “Firewalls in the Network Topology.”

Network Address Translation and Port Address Translation

When IPv4 was defined, the theoretical number of addresses it could provide (2^{32}) seemed more than enough to cover any need. However, the explosion of the Internet has shown that this original perception did not correspond to reality, and address exhaustion started to haunt the internetworking world. Clearly, a search for new solutions was needed.

One possible approach to solve this issue was to create a technology that had more bits in the addressing fields. The capability to deal with the problem of IPv4 address depletion was precisely one of the most important design goals of IPv6. However, despite the huge addressing capabilities of IPv6 (2^{128} addresses), its adoption was delayed because of the modifications it would impose to the whole structure of the Internet.

Given that most organizations have an amount of internal addresses much larger than the publicly routable ones (in the sense of RFC 1918), the second solution envisaged was to define a technique that could create *mappings* between the internal and external address spaces. This technique is called Network Address Translation (NAT) because it can *translate* between address spaces.

In its simplest mode of operation, NAT establishes a one-to-one correspondence between the *real* source address and the *virtual* (or translated) source address. Other more elaborate modes enable changing the destination addresses and building many-to-one translations. The various NAT types can be employed to accomplish several different tasks. Some of the most relevant ones follow:

- **Hiding private addresses from the global Internet:** The private addresses defined by RFC 1918 are commonly used inside companies but cannot be routed on the Internet. NAT lends itself to the task of confining these IPs to the organization boundaries.
- **Mitigating the problem of IPv4 address depletion:** The number of publicly routable addresses assigned to companies is frequently much smaller than the amount of internal hosts, thus creating the need for a technology that enables many-to-one mappings. The classic solution is called *Dynamic Port Address Translation* (*Dynamic PAT*).
- **Concealing the details of the internal network from the outside world:** Dynamic PAT not only handles the reduction of valid Internet addresses but also makes it possible to define unidirectional translations for hosts that should not be accessible from external sources.

Figure 1-5 depicts a sample scenario in which the firewall performs static address translation between the internal and external address spaces. Specifically, the source IP addresses R1 and R2 are respectively mapped to S1 and S2.

Figure 1-6 portrays an environment for which Dynamic PAT has been configured. The firewall associates to each source address in the internal space, a combination of a fixed external IP (the PAT-IP) and a randomly generated L4 source port. For each mapping, an entry in the translation table is added so that the firewall can handle the return packets.

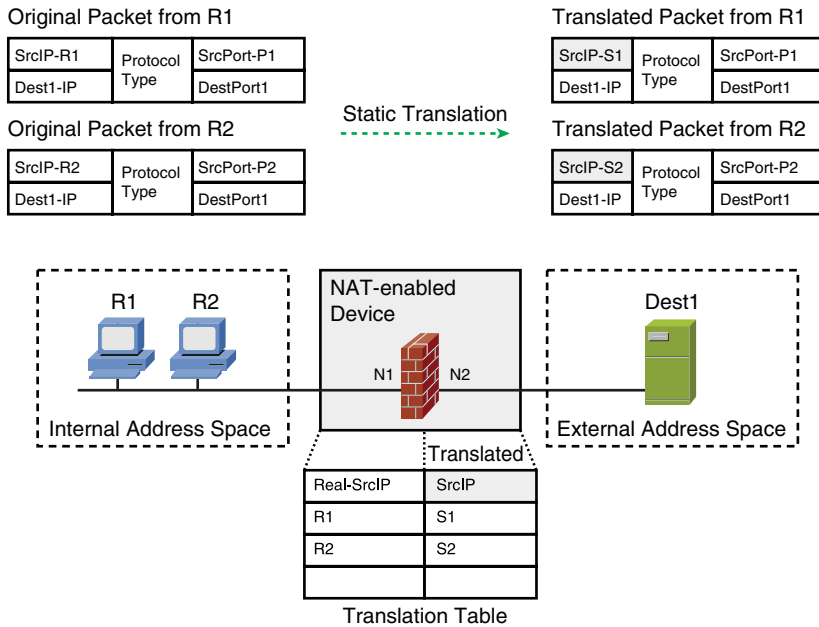


Figure 1-5 Network Address Translation in Action

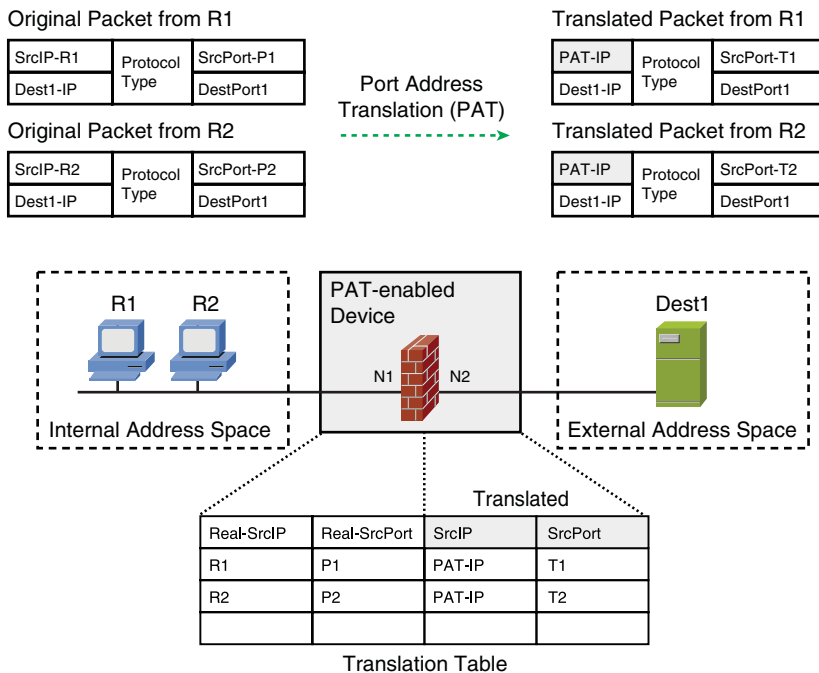


Figure 1-6 Dynamic Port Address Translation in Action

As just presented, the main motivation for deploying NAT and PAT relate to their capability to alleviate the problem of IPv4 address depletion. Nevertheless, many security and network professionals tend to consider that hiding private addresses from the public Internet is the most relevant application of these translation mechanisms. This creates a perception that NAT is a must for securely connecting an organization to the Internet, and NAT has quickly become a mandatory feature in any modern firewall.

Note Chapter 8, “Through ASA Using NAT,” presents a more detailed discussion about NAT categories.

Note In February 2011, an emblematic event took place: The last IPv4 address blocks were allocated to the Regional Internet Registries (RIR) by the Internet Assigned Numbers Authority (IANA). This represents an inflection point in the history of the next-generation Internet based on IPv6.

Note Chapter 16, “Cisco Firewalls and IPv6,” covers IPv6.

Main Categories of Network Firewalls

Although this book is centered around stateful firewalls, it is instructive to quickly review the main classes of network firewalls.

Packet Filters

Packet filters are first-generation firewalls that concentrate their access control efforts on some network and transport layer parameters of individual packets. They are *stateless* in nature because they do not have the concept of state table or connection.

Figure 1-7 shows a scenario for a which a packet filter is chosen as the firewall. The figure highlights the header fields that this type of firewall can specify when building its access control rules.

A practical implementation of this class of firewalls is the Access Control Lists (ACL) available on Cisco IOS routers and L3 switches.

Note Chapter 11, “Additional Protection Mechanisms,” examines some more advanced implementations of packet filters that go beyond simple header parameters.

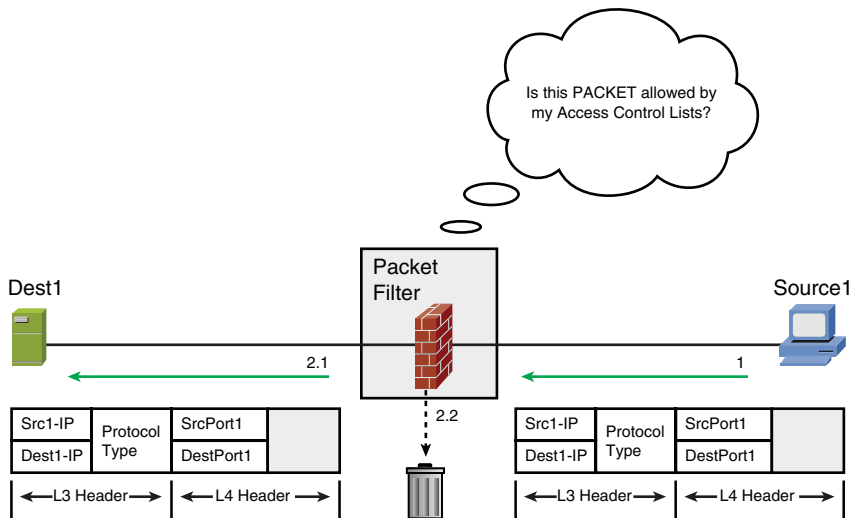


Figure 1-7 Overview of Packet Filters

Circuit-Level Proxies

Circuit-level proxies establish *sessions* to intended destinations on behalf of requesting hosts. The term session here refers to the Layer 5 of the OSI reference model, which is, conceptually, responsible for creating, managing, and terminating logical connections between application processes that reside on different machines.

These firewalls are sometimes referred to as *generic proxies* because they do not require an application-specific proxy software on the client side. On one hand, this provides flexibility because it is not necessary to develop a dedicated client for each application. On the other hand, this class of firewall does not understand how individual applications operate.

Figure 1-8 shows a high-level description of a SOCKS5 operation, one of the most well-known practical implementations of circuit-level proxies. The basic steps involved on session setup for this category of firewall follow:

- Step 1.** The SOCKS5 client opens a connection to the SOCKS server on a reserved TCP port and negotiates the authentication method to be used.
- Step 2.** The client authenticates with the agreed method and sends a *relay request* to the proxy server. This request contains the destination L4 port and IP address of the remote host reachable through the firewall (in this scenario, DestPort1 on Server1).
- Step 3.** The SOCKS server establishes the connection to Server1 on behalf of the client.
- Step 4.** The packets sent from the client to the proxy are then relayed to Server1.

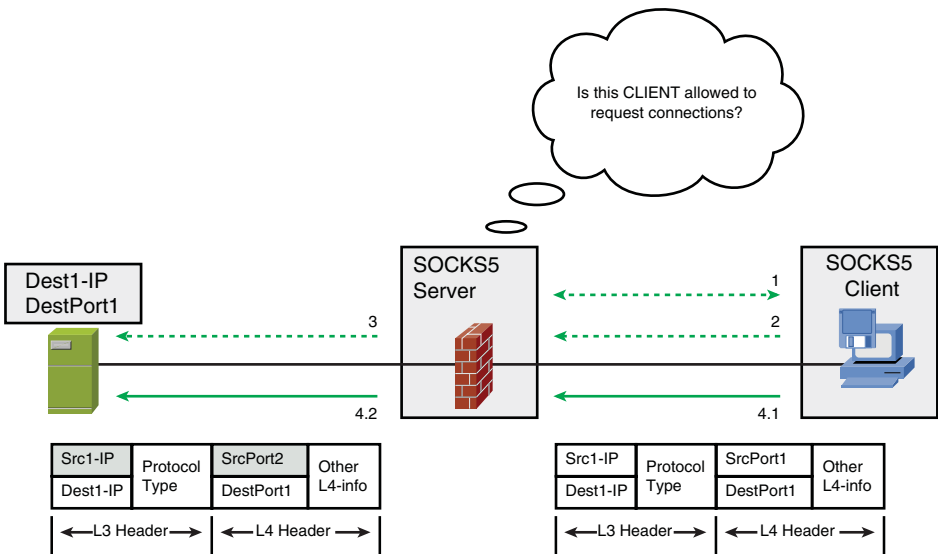


Figure 1-8 Overview of Circuit-Level Proxies

In this type of arrangement, the remote server has the perception that all traffic was originated by the proxy server.

Note Some confusion may arise because the connection-related activities typical of the OSI session layer are incorporated by the *transport layer* within the TCP/IP model.

Note RFC 1928 describes SOCKS5 operations.

Application-Level Proxies

An application-level proxy understands and interprets the commands of the application protocol it is providing proxy services for. Given that they require specific proxy software on the client side, they are also known as *dedicated proxies*.

Figure 1-9 illustrates the high-level operation of an application-level proxy dedicated to the HTTP protocol. In this case, the main tasks involved in connection setup follow:

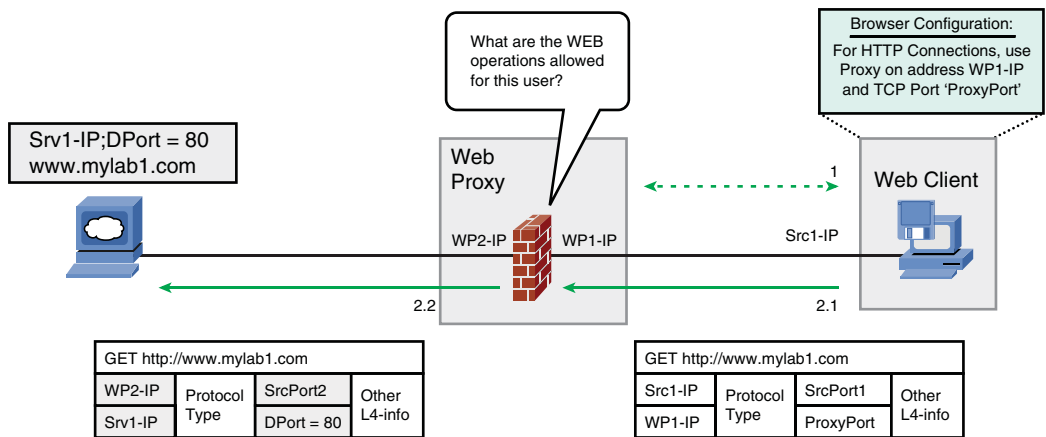


Figure 1-9 Overview of Application-Level Proxies

- Step 1.** The web client (browser), configured for using proxy services, authenticates to the web proxy. According to the user profile, the proxy can provide content filtering at the application level.
- Step 2.** All web traffic directed to the remote host is sent to the proxy (2.1), which changes the header information so that all packets seem to have been originated by it (2.2). All the packets from the web server get back to the client through the proxy.

Note Because of their application knowledge, this class of proxies can provide detailed logging and services such as authentication and caching. The shortcomings are the need to develop specific client software for each application and, normally, the CPU-intensive nature of this implementation.

Stateful Firewalls

This class of firewalls incorporates the concept of connections and *state* to packet filter implementations. Groups of packets belonging to the same connection (or *flow*), rather than individual packets, are used for access control.

Figure 1-10 depicts an environment in which the client C1 needs to initiate a connection through a stateful firewall to reach the TCP/Y1 service on destination host H1:

- Step 1.** The firewall checks its access control rules (much like a packet filter) to see if this type of connection initiation is allowed.
- Step 2.** For acceptable connections, an entry is created in the firewall state table, containing parameters such as source/destination IP addresses and TCP ports, the pertinent TCP flags, and the SEQ and ACK numbers.

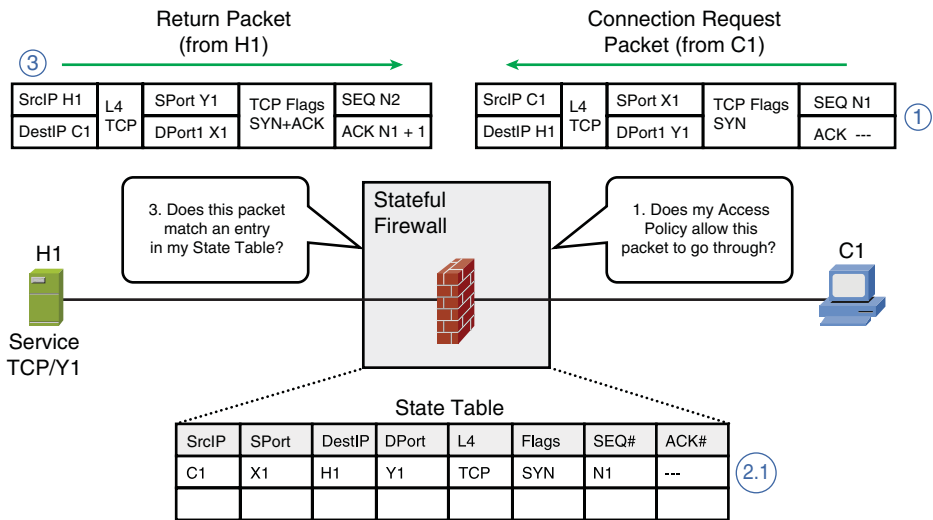


Figure 1-10 Overview of Stateful Firewalls

Step 3. The return packets from the remote server are compared to the state table and only allowed through if their parameters are coherent with the definitions of the TCP finite state machine.

Note The notions of state and connection originally refer to TCP, a *connection-oriented* transport layer protocol. Notwithstanding, the term connection is still used for protocols such as UDP and ICMP because a stateful firewall keeps track of parameters such as UDP port numbers and ICMP message types (and codes) in its state table. One way to dynamically terminate non-TCP connections is to enforce inactivity timeouts.

The Evolution of Stateful Firewalls

Stateful firewalls are a successful technology that enables the creation of flexible control rules, without compromising performance. This section briefly presents some important security capabilities that have been incorporated to this class of firewalls, enabling even more refined access control.

Application Awareness

Figure 1-11 shows an example of application-level inspection on a stateful firewall. The firewall understands the information on the application protocol headers and can leverage it for filtering purposes. For instance, it can enable the use of a small set of HTTP request types and completely block others.

It is relevant to emphasize that application visibility does not mean that stateful firewalls behave as application proxies. As shown in Figure 1-11, the packets are always exchanged

between client and server under the firewall's supervision. (It is insightful to compare this scenario with that of Figure 1-9.)

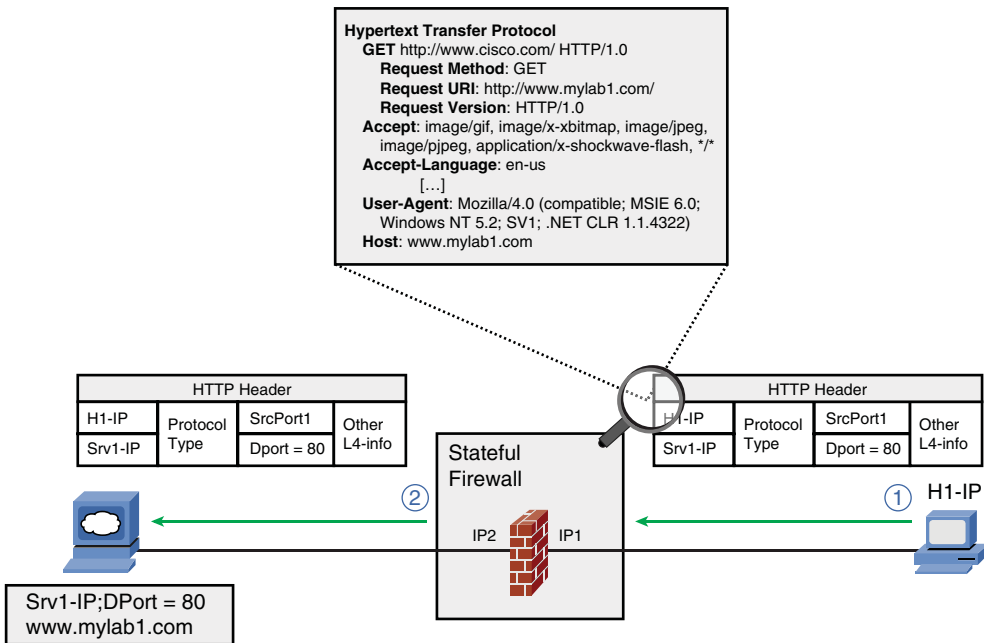


Figure 1-11 *Stateful Firewalls and Application Awareness*

Note To benefit from application awareness on stateful firewalls, it is not necessary to have an application-specific proxy client installed on hosts.

Identity Awareness

Stateful firewalls can leverage identity information for filtering purposes. Such an approach can provide administrators with the possibility of creating a distinct set of rules on a per-user (or per user-group) basis.

Figure 1-12 brings a high-level description of this process that will be later explored, in great level of detail, in Chapter 14, “Identity on Cisco Firewalls.”

- Step 1.** The client C1 intends to connect to remote server Dest1 through the firewall, which is configured to act as an authentication proxy.
- Step 2.** The firewall intercepts the connection request at the application level and asks the user to presents its credentials.
- Step 3.** The firewall forwards the user credentials to the policy server (for instance, using the RADIUS protocol).

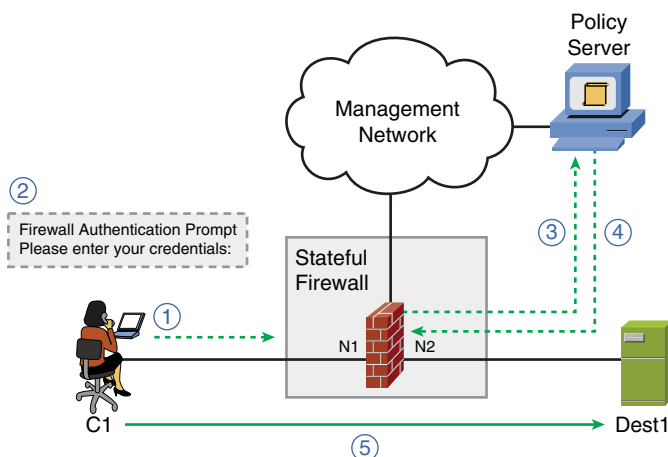


Figure 1-12 *Stateful Firewalls and Identity Awareness*

- Step 4.** The policy server authenticates the user and replies with an authorization profile, specifying what the user is allowed to do.
- Step 5.** Following authentication and authorization, the user can directly access the remote destination (much like traditional connections through stateful firewalls).

Leveraging the Routing Table for Protection Tasks

IP spoofing is an action through which a potential intruder copies or falsifies a trusted source IP address. This is typically employed as an auxiliary technique for a plethora of network-based attacks. Some possible motivations behind IP spoofing follow:

- Impersonating some trusted user or host and taking advantage of the privileges that arise from this trust relationship.
- Diverting attention away from the actual originator of the attack, with the intent of remaining undetected.
- Cast suspicion on legitimate hosts or users.

Among the antispoofing methods, the unicast Reverse Path Forwarding (uRPF) verification deserves special attention because of its scalability and ease of implementation. This elegant feature leverages the contents of the IP Forwarding table on firewalls to mitigate source address spoofing.

Figure 1-13 illustrates the Strict uRPF operation for two basic scenarios, the first corresponding to a successful uRPF check and the second to a uRPF failure. More details are provided here:

- In Scenario 1, a packet with source address *S* arrives on interface *Int1*. Given that the firewall verifies that this source address is reachable via the same interface on which it arrived, the packet passes the uRPF check and, therefore, is allowed.

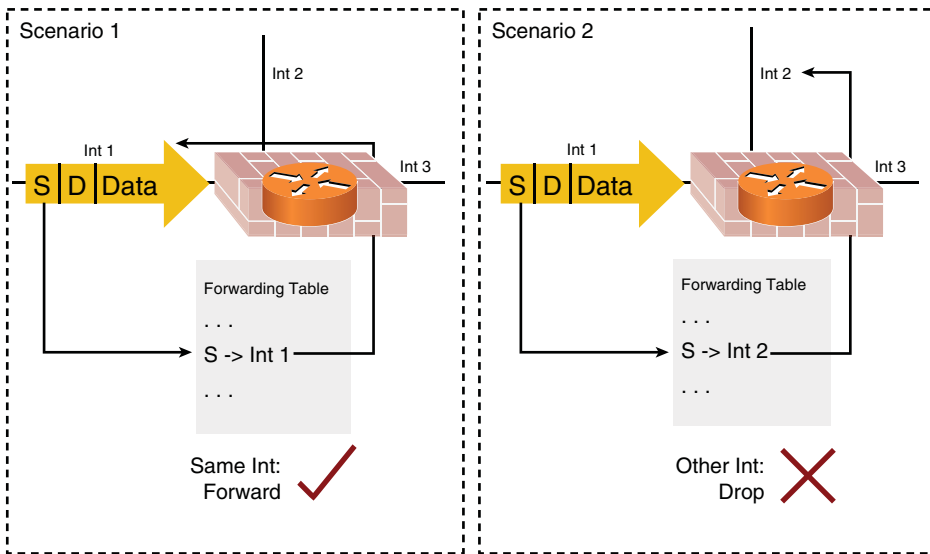


Figure 1-13 Antispoofing Using the Strict uRPF Technique

- In Scenario 2, the packet with source address *S* also arrives on Int1. But in this latter case, the firewall's forwarding table states that this address should be reachable via Int2. This inconsistency between the interface of arrival and the known reverse path to the source IP means that the uRPF check failed, and the packet is dropped.

Note The topic of antispoofing is thoroughly analyzed in Chapter 11.

Note uRPF antispoofing is a stateless feature that adds value to the work of both stateful firewalls and packet filters.

Virtual Firewalls and Network Segmentation

The word *virtualization* has been historically employed almost as a synonym of partitioning server resources. Nevertheless, security and networking devices may also have various features virtualized, therefore significantly contributing to improve the utilization levels of various classes of IT assets.

By carefully combining virtualized features and devices, an end-to-end architecture for virtualization becomes possible. The main building blocks for this architecture follow:

- **Virtual LANs (VLAN):** The classic deployment of VLANs is port-based, which means that specific physical ports of a LAN switch become part of a given VLAN.

- **VRFs:** A Virtual Routing and Forwarding (VRF) instance is a container of segmented routing information within Layer 3 devices. The basic components of a VRF follow:
 - An IP routing table.
 - A forwarding table, derived from the IP routing table.
 - A set of interfaces over which the L3 device exchanges routing information and forwards the packets whose destinations are reachable through the VRF.
- **Device Contexts:** These correspond to multiple logical elements running on a single physical device, therefore representing one step further toward complete device virtualization. Within Cisco Firewalls such as ASA appliances and the FWSM, each context owns a set of interfaces, security policies (ACLs, NAT rules, and inspection policies), a routing table, and management settings (configuration files, admin users and so on).

Figure 1-14 shows a classic scenario in which virtualization has been used to achieve network segmentation. In this environment, users belonging to User Group1 (UG1) willing to access their departmental servers use a sequence of virtual elements (user-side VLANs, a dedicated VRF, a virtual firewall and, finally, a server-side VLAN). This approach facilitates segmentation of network-based resources in the complete path from source to destination. One immediate consequence is that an eventual problem on a given logical segment does not impact other segments. (A similar analysis applies to the other user groups represented in the figure.)

What Type of Stateful Firewall?

This section examines the typical implementation options for stateful firewalls: dedicated appliances, router-based firewalls, or switch-based firewalls.

Firewall Appliances

These are purpose-built appliances (consisting of customized hardware and software), in which a stateful firewall is the main functionality.

The majority of modern appliances such as the Cisco Adaptive Security Appliances (ASA) are multifunctional, including features such as VPN and intrusion prevention. However, on environments such as the headquarters of large companies, it is common (for performance reasons) to dedicate these appliances to firewall or VPN services.

Router-Based Firewalls

Places in the network, such as remote branches, have a stronger appeal for flexibility and integration capabilities (rather than throughput), and normally are much more subject to budget constraints. Cisco IOS router-based firewalls fit well in these scenarios as part of

the all-in-one solution provided by Cisco *Integrated Services Routers (ISR)*. Among the services available in the ISR platforms, some are briefly described here:

- **Multiprotocol routing:** This is the original service that made Cisco routers so successful in the networking industry. Feature-rich implementations of routing protocols such as Open Shortest Path First (OSPF), Enhanced Interior Gateway Protocol (EIGRP), and the Border Gateway Protocol (BGP) are available in most ISR models.

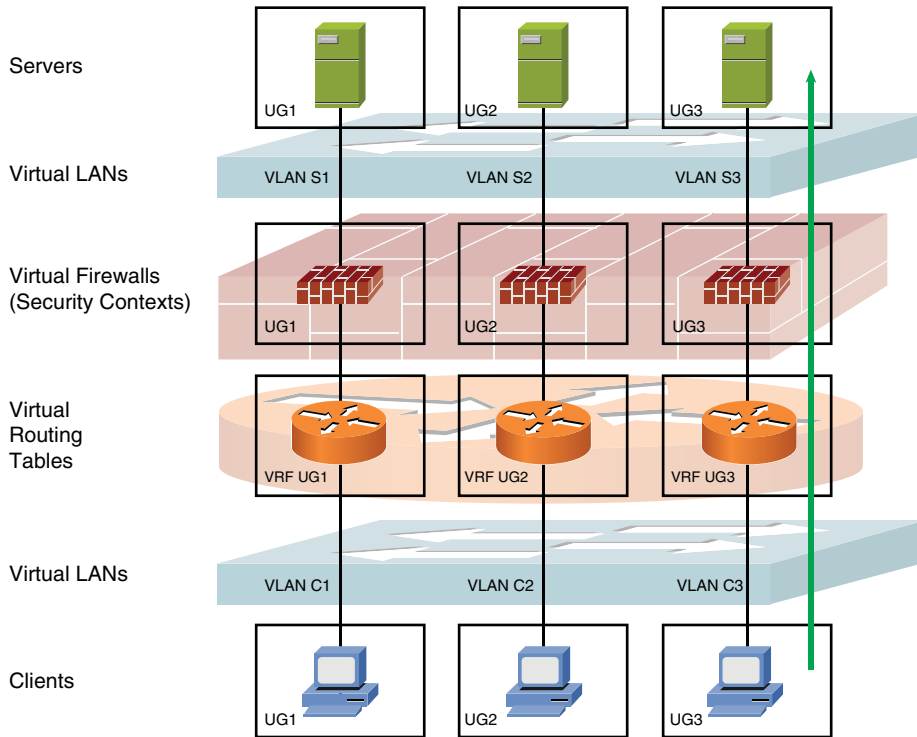


Figure 1-14 *Sample Network Segmentation Scenario*

- **Stateful inspection with a high level of application awareness:** From the Classic IOS Firewall to the modern option known as the Zone-based Policy Firewall (ZFW), full stateful firewall functionality is provided.
- **Identity services:** Authentication-proxy functionality, user-based firewall features, and Network Admission Control (NAC) are some of the services in this category.
- **Wide spectrum of secure connectivity models:** From the classic IPsec service to advanced VPN options such as Group Encrypted Transport (GET VPN), Cisco IOS offers a wide spectrum of secure connectivity solutions.
- **Integrated intrusion prevention:** Cisco IOS IPS is a natural companion to the integrated stateful firewall functionality in Cisco IOS routers.

- **Convergence services:** Cisco is a historic sponsor of *network convergence*, supporting VoIP, VoFR, and VoATM since the early days of these technologies. Voice gateway and IP PBX features (*Call Manager Express*) are other examples of IOS-based convergence services.

Note Although dedicated firewalls (such as Cisco ASA) are *intrinsically closed*, a router is a connectivity provider and, by default, does not impose access restrictions by means of any inherent packet filters. The Cisco IOS ZFW enables administrators to easily convert this classic router behavior into the *implicit deny* model that characterizes dedicated firewalls.

Switch-Based Firewalls

The Cisco Firewall Services Module (FWSM) for Cisco Catalyst 6500 Series switches is a firewall solution based on the ASA stateful inspection technology. Any physical port on the hosting Catalyst switch can be configured for firewall policies, saving time and efforts related to cabling.

The FWSM is well suited for intranets and virtualized environments. One of its premium features is the capability to run any combination of Transparent mode and Routed mode security contexts.

Note The FWSM is also supported on Cisco 7600 Series routers.

Classic Topologies Using Stateful Firewalls

Figure 1-15 summarizes classic network topologies in which stateful firewalls have been largely deployed:

- **Internet access:** This simple arrangement protects internal users that need access to the Internet. The firewall policy specifies what traffic is enabled outbound and simply prohibits inbound connection initiation. Inbound packets are allowed only when they correspond to an existent entry in the firewall state table.
- **Internet presence:** This scenario includes a *Demilitarized Zone (DMZ)*, in which the publicly accessible servers are located. The firewall policy basically defines which services in each DMZ host (server) should be visible on the outside. The firewall watches dmz-bound connection setup and keeps track of return traffic. Typically, the DMZ servers are not allowed to initiate outbound connections. For instance, this measure helps on blocking content upload to the outside in the eventuality of a server compromise.

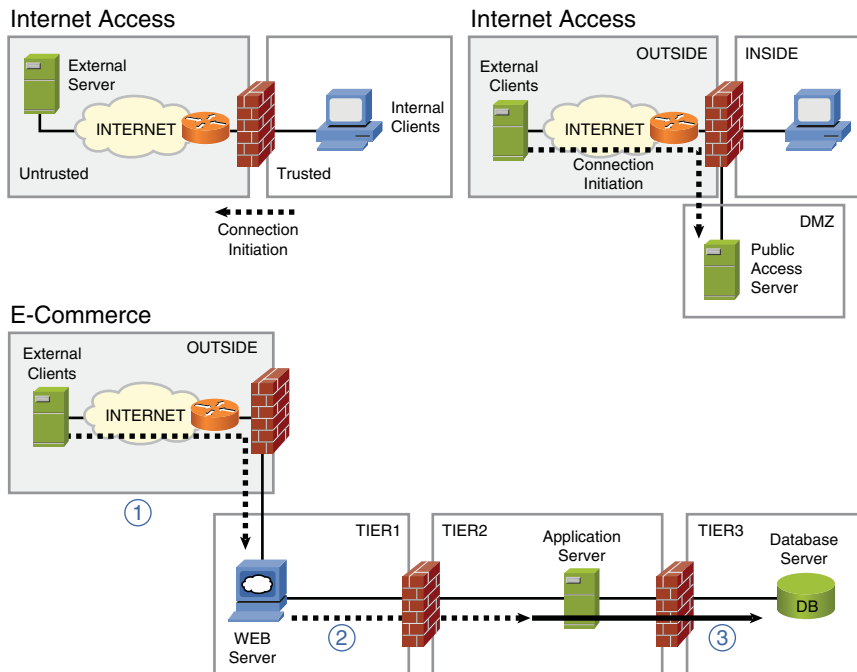


Figure 1-15 *Classic Firewall Topologies*

- **E-commerce:** This topology enhances the Internet presence arrangement by adding extra separation tiers between the various servers involved in a typical e-commerce environment. It is convenient to combine different classes of firewall (commonly stateful firewalls and application proxies) to increase security in such a critical part of the network.

Stateful Firewalls and Security Design

Stateful firewalls can be used with other security devices (including other firewall categories) to materialize the principle of having multiple layers of defense. This section presents a few sample combinations as a preview of the detailed analysis that is the subject of Chapter 17, “Firewall Interactions.”

Stateful Firewalls and VPNs

Virtual private networks were conceived to extend the reach of corporate networks (and the applications they provide) without the need to use a dedicated infrastructure based on expensive circuits. Although several VPN technologies are in use (each of them providing a certain level of security), the IPsec framework is doubtlessly the major technology responsible for the large proliferation of this cost-effective way to provide connectivity to remote entities (users and sites).

Figure 1-16 portrays a scenario in which a VPN client uses the Internet to establish an encrypted tunnel to a VPN concentrator. Following the decryption performed by the VPN concentrator, the stateful firewall is exposed to clear text traffic and then can control access to resources hosted on the corporate servers.

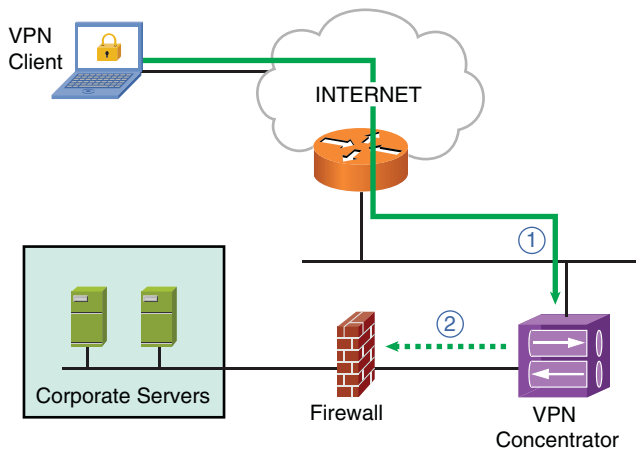


Figure 1-16 *Integrating Stateful Firewall and VPNs*

Figure 1-17 illustrates a router-based stateful firewall that has integrated VPN support. This enables the branch router to use the Internet as a backup connection to the corporate headquarters in a secure way. The VPN tunnel may also be used to transport noncritical traffic while the main WAN circuit is fully operational.

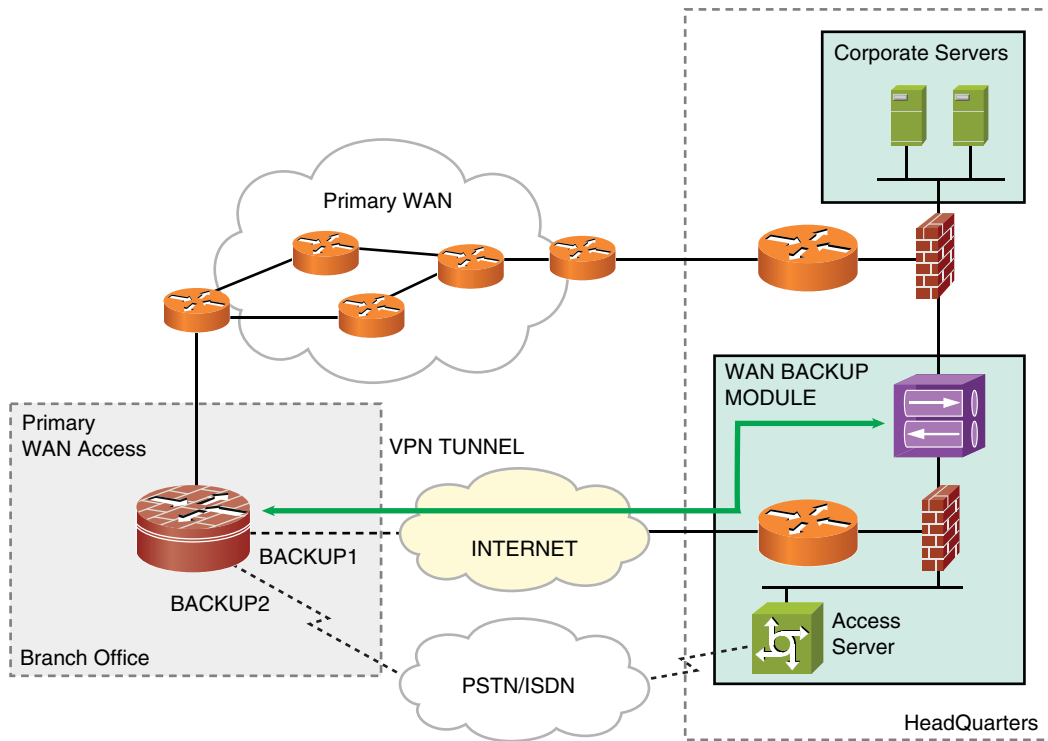


Figure 1-17 Sample WAN Backup Scenario Using Stateful Firewall and VPN

Note Chapter 17 analyzes scenarios that involve the integration of firewall and VPN functionalities in the same device.

Stateful Firewalls and Intrusion Prevention

Intrusion detection and prevention technologies can provide an in-depth look at the data inside network packets so that the occurrence of malicious traffic can be determined. IDS systems are mainly targeted at monitoring and alerting activities, whereas IPS solutions have the intrinsic goal to prevent attacks and, therefore, avoid actual damage to systems.

IPS systems can search for well-known attack patterns within packet streams and take an action according to the configured policy. These actions include (but are not limited to) dropping packets, blocking connections, denying access to the address that originated the attacks, and alerting when an attack indication (*signature*) is detected.

If an IPS is deployed like in Figure 1-18, it can further inspect the connections enabled by the firewall, therefore significantly increasing the level of security achieved:

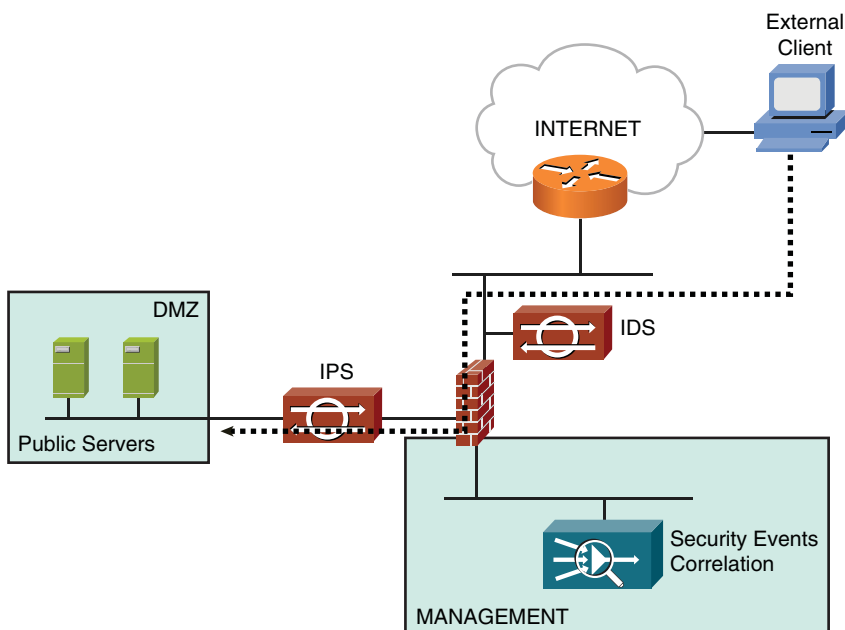


Figure 1-18 *Integrating Stateful Firewall and Intrusion Prevention Technologies*

- The first part of the job is performed by the firewall, which selects acceptable traffic according to combinations of source/destination addresses and ports and keeps track of connections using a state table.
- The IPS inspects only the valid traffic (from the firewall's standpoint) and can do a more focused analysis, thus avoiding the waste of CPU resources to block packets that did not comply with simpler rules. For instance, you do not need to use a sophisticated Layer 7 inspection (of header and data) to block a packet that was sent by an invalid source. (The firewall can do that easily.)
- If firewalls block lots of packets before they can even be directed to the IPS, there will be fewer events in the IPS management console, and the administrator will spend less time on meaningless alerts.

Some security administrators like to deploy an extra IDS appliance (monitoring only) in the outside of the firewall, in a configuration similar to that of Figure 1-18. The purpose of such an approach is to gain more insight about *who is looking around*, even if attacks are not effective. Solutions that can correlate events generated by firewalls, IPS, and IDS solutions add great value to security operations.

Note Chapter 17 goes a bit further on the discussion of firewall and IPS interactions, by using multifunctional elements or dedicated devices for each function.

Stateful Firewalls and Specialized Security Appliances

Stateful firewalls can be successfully integrated with specialized security appliances to provide a greater level of control to critical applications. These specialized appliances can be considered application-level proxy firewalls.

Figure 1-19 illustrates two sample integration scenarios described here:

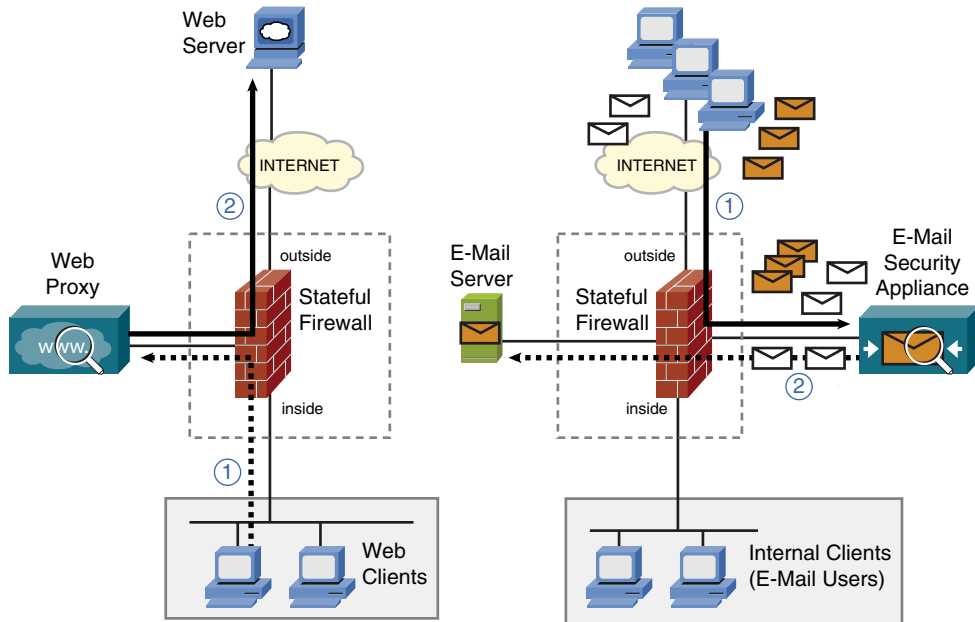


Figure 1-19 *Stateful Firewalls and Specialized Security Appliances*

- A web proxy can enhance an Internet access scenario by controlling the access of internal users to outside web servers. One example of such a proxy is the Cisco Ironport Web Security Appliance (WSA). Among many other features, the WSA is capable of filtering malware, enforcing acceptable-use policies, filtering URLs, and controlling access based on the concept of web reputation.
- An email proxy is deployed to control the interaction of external clients with the email server. An example of such a specialized proxy is the Cisco Ironport E-mail Security Appliance (ESA). The ESA supports virus scanning, Data Loss Prevention (DLP) and spam prevention, just to name a few of its capabilities.

Summary

This chapter started by underlining the importance of creating a security policy before any actual security deployment takes place.

The chapter also presented some other important topics:

- The concept of network firewalls and the options for their insertion in the network topology
- A brief review of the main categories of network firewalls
- How stateful firewalls have incorporated other security capabilities
- Use of stateful firewalls as a pivotal element for security design

Chapter 2, “Cisco Firewall Families Overview,” goes a bit further by describing the actual firewall models currently offered by Cisco.

Cisco Firewall Families Overview

This chapter covers the following topics:

- Overview of ASA appliances
- Overview of the Firewall Services Module
- Overview of IOS-based integrated firewalls

“Common sense is the best distributed commodity in the world, for every man is convinced that he is supplied with it.”—René Descartes

Chapter 1, “Firewalls and Network Security,” introduced the subject of firewalls and the role they play within the context of network security. After exploring the main categories of firewalls, the evolution of the specific technology of stateful firewalls has been outlined.

This chapter is intended to complement the previous one, by promoting a quick review of Cisco hardware platforms that host stateful firewall solutions. It also discusses some of the relevant parameters for firewall selection (either from a performance or services integration standpoint). The remaining chapters cover the most important software features.

Before starting the actual work, it is convenient to give a word of warning: This chapter by no means should be compared with the online resources available on the Cisco official website. Because new hardware models are frequently added to the portfolio, it is always advisable to search information online and consult your Cisco sales representatives so that you know about the latest possibilities for your firewall projects. Sometimes a higher-performance model is introduced in the marketplace with a lower price than earlier members of the same family.

Overview of ASA Appliances

On the hardware side, the Cisco ASA family of security appliances has its roots on the classic PIX Firewall, which employs Adaptive Security Algorithm (ASA) instead of the

current meaning for ASA, which is *Adaptive Security Appliance*. On the software side, all the original firewall features available in the PIX Firewalls are present on ASA appliances. However, ASA not only significantly increased performance-related numbers but also added support to new services such as full intrusion prevention and SSL-VPN.

Positioning of ASA Appliances

Figure 2-1 summarizes ASA positioning according to the *place in the network* where a given model is designed to be used. This figure separates the ASA models into two groups:

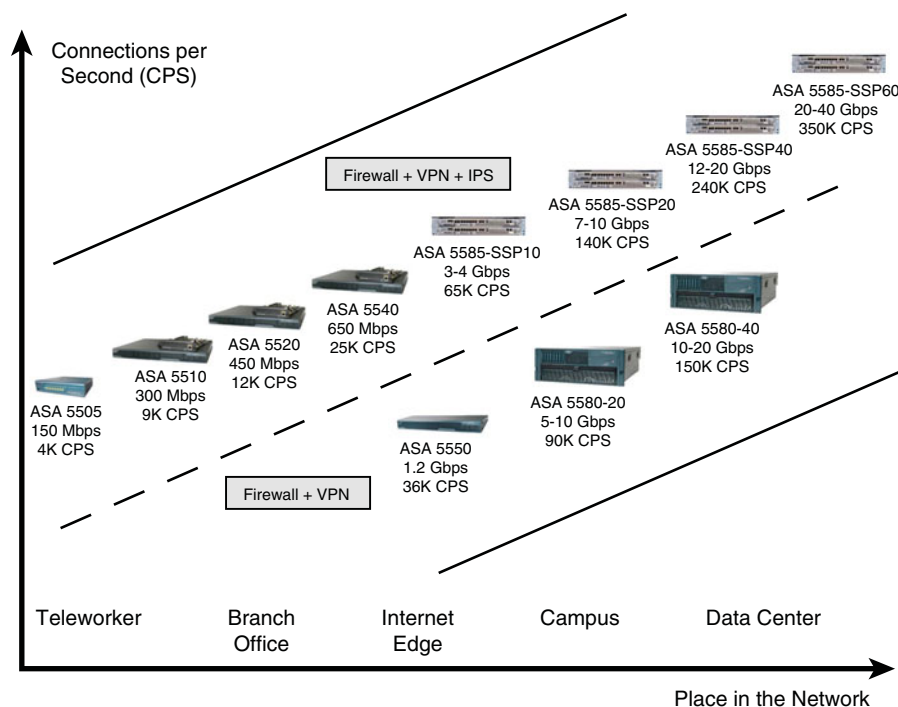


Figure 2-1 A High-Level View of ASA Positioning

- Appliances that support firewall, intrusion prevention (by means of a dedicated hardware module), and VPN (IPsec and SSL) functionalities. This group is composed of the ASA 5505, 5510, 5520, and 5540 appliances and the ASA5585 series (containing the SSP10, SSP20, SSP40, and SSP60 models).
- Appliances that support firewall and VPN (IPsec and SSL) services. This group consists of the ASA 5550 and ASA5580 models (5580-20 and 5580-40).

All the appliances belonging to the ASA family have stateful firewall and IPsec VPN functionality enabled by default. SSL VPN, on the other hand, is licensed separately on a per-user basis.

In Figure 2-1, you can notice that there is always a 5585 model that supersedes a given 5580 in all parameters, with the significant advantage of also supporting intrusion prevention. The same reasoning applies to the 5550 when compared to the ASA5585-SSP10.

Note ASA5580 models have their End of Sale (EoS) and End of Life (EoL) programmed to August 2011 and August 2016, respectively. This is a compelling reason to consider ASA5585 models on newer projects.

Firewall Performance Parameters

Figure 2-2 summarizes the main parameters typically used for firewall selection for the ASA models that support intrusion prevention functionality. If you need information about the parameters for the other members of the ASA family, refer to the online documentation.

	5505	5510	5520	5540	5585 SSP10	5585 SSP20	5585 SSP40	5585 SSP60
Max CPS Rate	4K	9K	12K	25K	65K	140K	245K	350K
Max PPS Rate	85K	190K	320K	500K	1.5M	3.2M	6M	10.5M
Simultaneous Connections	10K / 25K	50K / 130K	280K	400K	1M	2M	4M	10M
Firewall Throughput (bps)	150M	300M	450M	3G	2G	7G	12G	20G
Max Firewall Throughput (bps)	150M	300M	450M	650M	4G	10G	20G	40G
Max FW + IPS Throughput (bps)	75M	300M	450M	650M	2G	3G	5G	10G
Max 3DES/AES VPN Throughput (bps)	100M	170M	225M	325M	1G	2G	3G	5G
Max Site-to-Site / RA VPN Sessions	10/25	250	750	5K	5K	10K	10K	10K
Max SSL VPN Sessions	25	250	750	2500	5K	10K	10K	10K
Max VLANs	3 / 20	50 / 100	150	200	1024	1024	1024	1024
HA Active-Standby	Stateless (SecPlus)	(SecPlus)	Yes	Yes	Yes	Yes	Yes	Yes
HA Active-Active	No	(SecPlus)	Yes	Yes	Yes	Yes	Yes	Yes

Figure 2-2 Main Performance Parameters of the IPS-Enabled ASA Models

At this point, it is interesting to use the backdrop provided by Figures 2-1 and 2-2 to examine some of the firewall-related numbers. Despite its criticality for a stateful device, the Connections per Second (CPS) parameter is often overlooked. Most of the published firewall data sheets focus on throughput (bps). Although it might sound intuitive to adopt the reasoning *the more Gbps, the better*, this conclusion does not hold for stateful firewalls.

The Gbps attribute is particularly important for a Layer 2 switch, a type of device that forwards frames based on association of a destination MAC address to a physical port. A stateful firewall uses *conditional forwarding* because it performs various checks before actually using its backplane to transfer a packet between the ingress and egress interfaces. A sample ASA packet flow for firewall and IPS features is shown in Figure 2-3. This flow chart highlights that connection verification is the first step in the path from incoming to outgoing interface. Further, for the case of a new connection, there are various decision points in which the packet could have been dropped (before the firewall can send it to the egress interface).

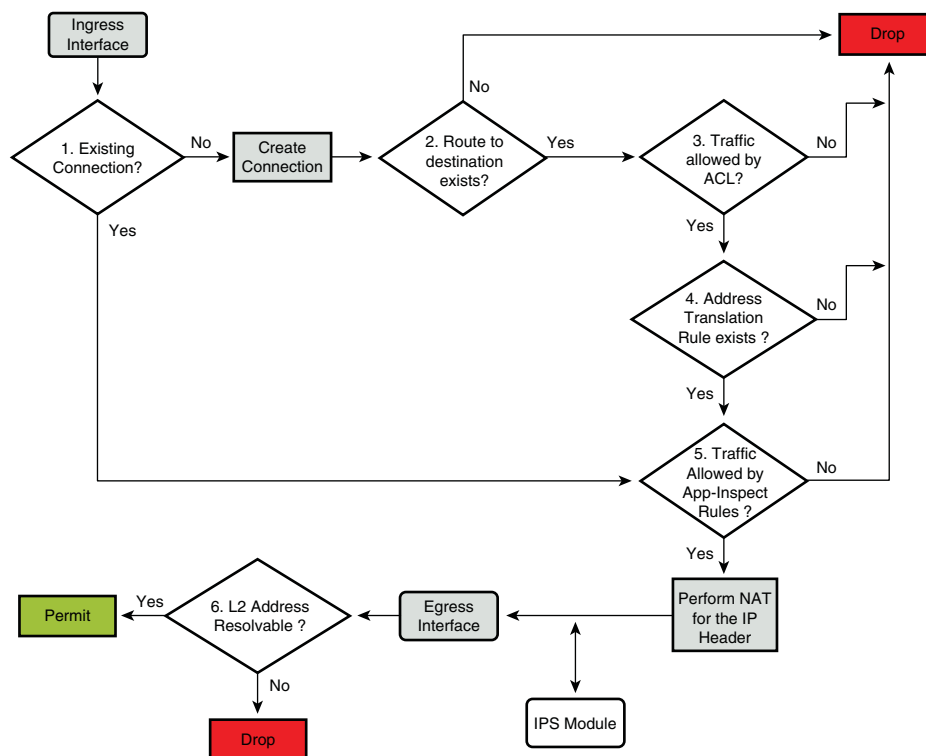


Figure 2-3 Sample Packet Flow for an ASA Appliance

Packets per Second (PPS) is the classic metric for evaluating router performance. Considering that, most of the time, firewalls are inserted in the network topology as Layer 3 elements, this parameter cannot be despised. Additionally, from the perspective of a stateful firewall, a connection might contain multiple packets. Two examples that illustrate the relationship between packets and connections are registered here:

- **The TCP three-way handshake process:** When using TCP, before the application flow starts, there should be at least three packets involved.

- **Large file transfers:** The transfer of large files using protocols such as FTP or TFTP can involve a large quantity of packets within the same data connection.

Many customers complain about their firewalls facing problems with *small packets*. A few of them even had the impression that their products had some kind of limitation when dealing with this class of packets. In essence, what was going on in most of the situations could be described as follows:

- The data sheet numbers were calculated using Jumbo Frames (~9000 bytes) and expressed in Gbps.
- Traffic in a real-world environment had an IMIX-like profile (an average of 450 bytes per packet). Considering that customers associated performance with Gbps, the disappointment was significant: a performance 20 times lower than the marketing numbers. (Just think about discovering that a claimed 30 Gbps throughput was not more than 1.5 Gbps....)

The simplified calculations that follow can provide a reasonable guidance for linking the PPS and Gbps values:

- $450 \text{ bytes / packet} = 3600 \text{ bits/packet} = 3.6 \times 10^3 \text{ bits / packet (a)}$
- $1 \text{ Gbps} = 10^9 \text{ bits / second (b)}$
- $10^9 \text{ bits/second} = (\text{Y packets / second}) \times (3.6 \times 10^3 \text{ bits / packet (c)})$
- (a), (b), and (c) yield **Y ~ 278,000 packets / second**, meaning that you would roughly need **278K PPS** for each Gbps of IMIX traffic.

If your deployment scenario involves smaller packets, your computations should be even more conservative.

The *Simultaneous Connections* parameter is directly related with the amount of memory reserved for the state table. This variable is also somewhat interconnected with the CPS value. For instance, the ASA5585-SSP10 supports 1M concurrent connections and a CPS rate of 50 K (see Figure 2-2). $1000\text{K connections} / 50 \text{ K CPS} = 20 \text{ seconds}$, meaning that this ASA model can accept new connections at the maximum setup rate of 50 K CPS for a period of 20 seconds (without rejecting connections).

Note In Figure 2-2, whenever a cell contains values in the form X / Y for the 5505 and 5510 platforms, X means the parameter value for the *Base License*, whereas Y corresponds to the *Security Plus* (SecPlus) license.

Note Whenever planning for the CPS parameter, it is important to keep in mind that the selected value should process setup rates higher than those observed during normal firewall operations. For example, there should be some room for dealing with DoS attack situations. (Well-dimensioned firewalls can significantly contribute to DoS mitigation.)

Figure 2-3 presents a possible packet flow for the ASA appliance in a situation for which there is neither VPN nor user-based features deployed. The purpose of this figure is to illustrate the classic tasks that are part of ASA firewall operations. Details about each step are the subject of later chapters.

Overview of ASA Hardware Models

Figure 2-4 registers the front and rear views for the ASA5505, the smallest model in the ASA family. Some of its noteworthy characteristics follow:

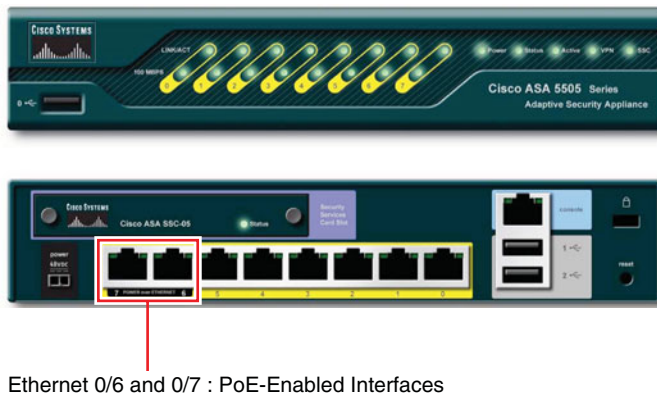


Figure 2-4 *Hardware Overview for the ASA 5505 Appliance*

- It has an 08-port integrated switch. Two of the ports are PoE-enabled and can be used for delivering power to IP Phones or Wireless Access Points in home or small of-office deployments.
- Referring to the table in Figure 2-2, the 5505 supports only three VLANs with the Base license (without VLAN Trunking). 802.1Q capability and additional VLANs (up to 20) come with the Security Plus license.
- The 5505 is typically positioned for Enterprise teleworkers or branch office deployments.

Figure 2-5 shows the front and rear views for the ASA 5510, 5520, and 5540 models. Relevant information about these IPS-capable devices follows:

- The Advanced Inspection and Prevention Security Services Modules (AIP-SSM) have their own Out-of-Band (OOB) management port. Their communication with the ASA (for firewall and IPS integration) happens through an interface that is internal to the ASA hosting chassis.
- Although each appliance in this group supports more than one AIP-SSM model, there is always one specific SSM designed to match firewall and IPS performance

values. For instance, while the AIP SSM-20 delivers 500 Mbps of IPS services on an ASA 5540, the AIP SSM-40 can match the maximum firewall throughput of 650 Mbps for this ASA model.

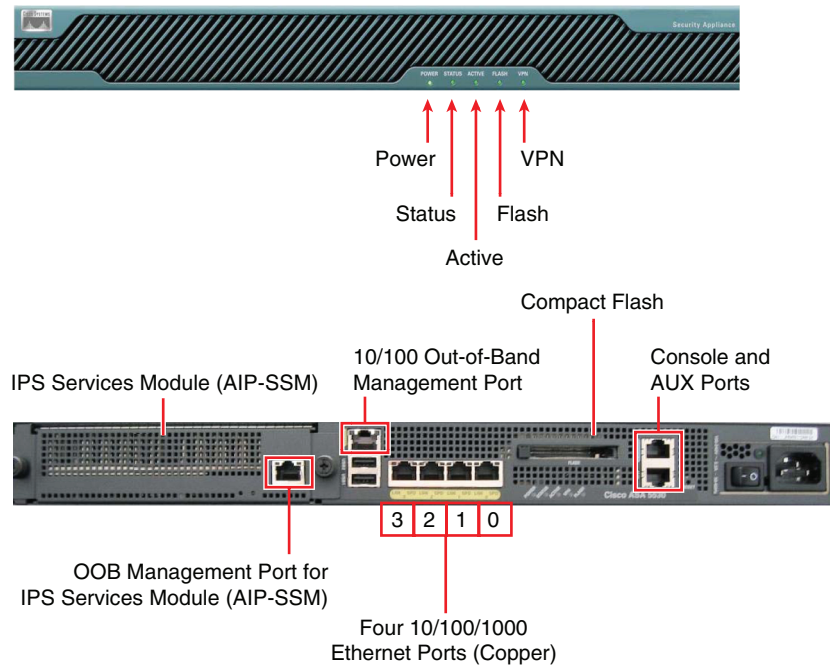


Figure 2-5 *Hardware Overview for the ASA 5510, 5520, and 5540 Appliances*

- High-Availability (HA) in the ASA5510 is enabled by the Security Plus license.
- The Security Plus license is required in the ASA 5510 to enable ports 0 and 1 to operate as 1000 Mbps. These ports are limited to 10 / 100 speeds when the 5510 has only the Base license.
- The OOB Management port can be used as any other port for passing traffic through the firewall. Chapter 3, “Configuration Fundamentals,” covers the command to enable this behavior.

Note The ASA 5550 differs from those appliances represented in Figure 2-5 in one noticeable point: Instead of having a slot that might receive an AIP-SSM module, it ships with 4 extra factory-installed Gigabit interfaces on that slot. The ASA 5550 does not support IPS functionality.

Figure 2-6 shows the rear view of an ASA 5580 appliance. One critical fact to be aware of is that slots 5, 7, and 8 are PCI x8 (versus the remaining PCI x 4), meaning that these specific slots are the most suited to be populated with 10-Gigabit Ethernet modules.

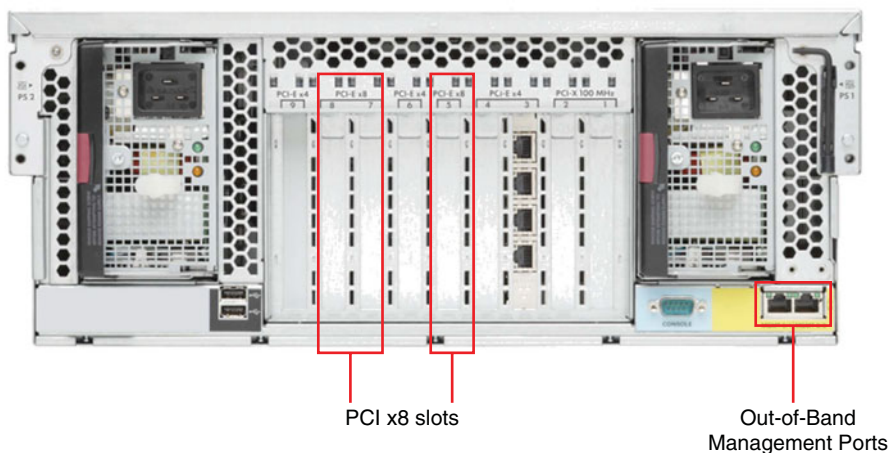


Figure 2-6 *Hardware Overview for the ASA 5580 Appliance*

As previously stated, the ASA 5585 models should be used in place of the 5580 for any new deployments.

As summarized in Figures 2-1 and 2-2, the 5585 subfamily of ASA appliances significantly increase critical performance parameters such as PPS and CPS, while making integration of IPS functionality possible on higher-end devices.

In the terminology used to describe the ASA 5585 models, you can see the acronym SSP, which stands for *Security Services Processor*. The ASA-SSP module is mandatory for all ASA5585 deployments and is installed on slot 0, whereas the IPS-SSP (always residing on slot 1) does not need to be included initially. A basic view of slot arrangement in the 5585 platforms is shown in Figures 2-7 and 2-8.

The rear view of ASA 5585 appliances, all of which support redundant and hot-swappable power supplies, is portrayed in Figure 2-7, which also shows the front view of the ASA5585 SSP10 and SSP20 models, whereas Figure 2-8 concentrates on the hardware overview for the ASA5585 SSP40 and SSP60 appliances. One noticeable difference from the two high-end models when compared with the two lower-end ones (SSP10 and SSP20) is the combination of copper and SFP-based interfaces:

- The SSP10 and SSP20 ship with 8 integrated 10/100/1000 copper interfaces and two slots for Small Form-Factor Pluggable (SFP) transceivers, which offer either Gigabit or 10GigabitEthernet connectivity (in this last case requiring the use of a SFP+ adapter).
- The SSP40 and SSP60 offer 6 fixed 10/100/1000 copper interfaces and 4 slots for SFP/SFP+ flexible connectivity.

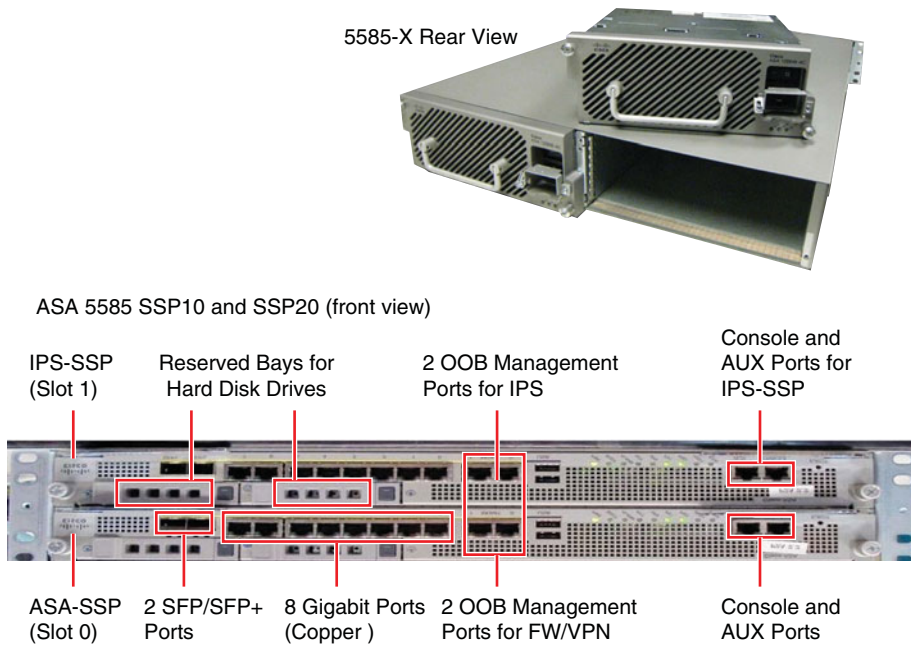


Figure 2-7 *Hardware Overview for the ASA 5585 SSP10 and SSP20 Appliances*

ASA 5585 SSP40 and SSP60 (front view)

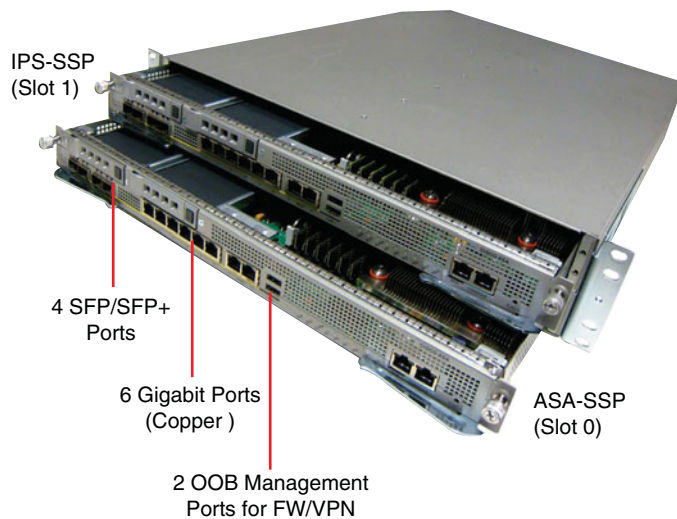


Figure 2-8 *Hardware Overview for ASA5585 SSP40 and SSP60 Appliances*

Figure 2-7 highlights that the IPS-SSP not only includes dedicated OOB Management ports (as other Cisco IPS products) but also specific console and auxiliary ports. In terms of external interfaces, the initial options offered for the ASA-SSP and IPS-SSP are identical. Both have fixed Gigabit Ethernet (Copper) ports and mini-slots for use with SFP and SFP+ adapters (used for 10GigabitEthernet). Another noticeable characteristic is the existence of two hard disk drive bays on each SSP, which are reserved for future use.

Overview of the Firewall Services Module

The Cisco Firewall Services Module (FWSM) for Cisco Catalyst 6500 series switches is a dedicated firewall solution that builds upon Cisco ASA (*Adaptive Security Algorithm*) stateful inspection technology. As already happened with the firewall functionality on ASA appliances, the FWSM also has Cisco PIX Firewall as its main ancestor. One natural consequence of that influence is the similarity that can be readily detected on the *Command Line Interface (CLI)* and configuration philosophy for all these three products.

The FWSM is a specialized, firewall-only device. It can be combined though with other Catalyst 6500 services modules, such as the Application Control Engine (ACE) module, which enables a single chassis to provide additional services such as Server Load Balancing (SLB).

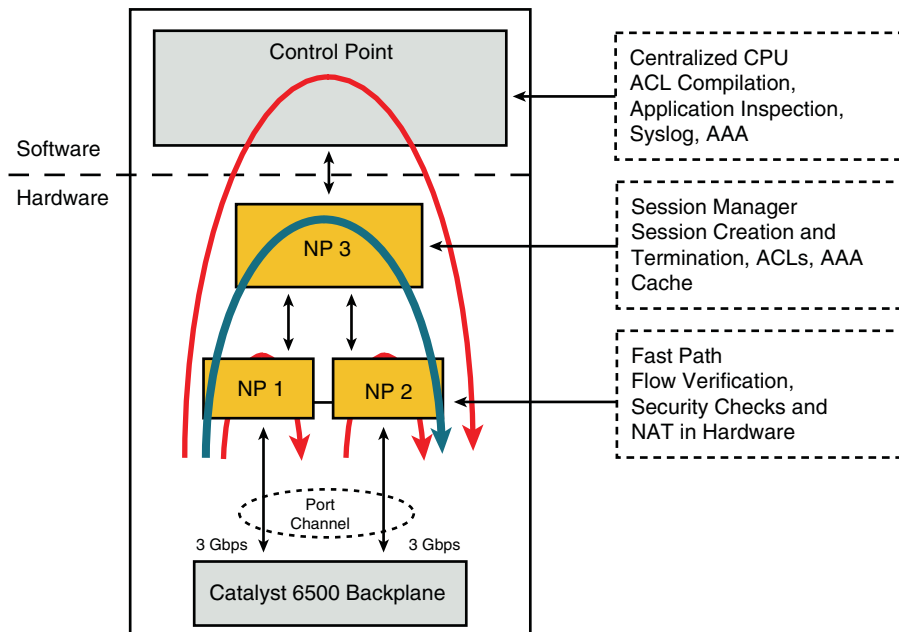
The communication between the FWSM and the underlying Catalyst 6500 happens through the switch backplane. When using FWSM, any physical port on the switch can be enabled for firewall policies, saving time and efforts related to cabling. On the other hand, only those VLANs in the hosting chassis that have been explicitly assigned to the module will have access to FWSM services. This means that the switch can keep behaving as usual for the remaining VLANs.

The FWSM is well suited for intranet Data Centers, where it can, for instance, control client access to the server-side VLANs. It also fits well for virtualized deployments, such as segmenting departments inside a company or offering firewall to customers, as an optional infrastructure service, within a service provider environment. One distinctive feature of the FWSM resides on its capability to run any combination of Transparent mode and Routed mode security contexts. (Virtualization is examined in Chapter 6, “Virtualization in the Firewall World.”)

Performance numbers for the FWSM are summarized in the following:

- 100,000 connections per second (CPS).
- 3 million packets per second (PPS).
- More than 5 Gbps firewall throughput per module. Up to 4 modules can reside on a given chassis, delivering an aggregate throughput of 20 Gbps.
- 1 million concurrent connections.

Figure 2-9, in which NP means network processor, depicts the main components of the FWSM’s hardware architecture:



9

Figure 2-9 Overview of the FWSM Architecture

- NP3 is frequently referred to as the Session Manager. It processes the first packet in a connection, counts established and incomplete connections, and creates translations. It is also in charge of performing TCP sequence number randomization (a topic covered in Chapter 7, “Through ASA Without NAT,” and Chapter 8, “Through ASA Using NAT,”) and dealing with TCP and UDP checksums.
- NP1 and NP2 are called the Fast Path. These processors, among other tasks, are responsible for maintaining the connection table, performing per-packet session lookup, translating addresses (Network Address Translation [NAT] and Port Address Translation [PAT]), and reassembling fragments.
- The Control Point (CP) corresponds to the main CPU: It handles traffic destined to or sourced from the FWSM, which basically corresponds to management (SNMP, SSH, Telnet, HTTPS, and so on) and control protocols (failover tasks, routing protocols, TACACS+, and so on). The CP also takes care of application protocol inspection (translating addresses embedded on the application layer, filtering application commands, and so on). The CP, the processor running the FWSM code, is often called the Slow Path.

Note The FWSM is also supported on Cisco 7600 series routers. Nevertheless, for the sake of simplicity, all the references to it hereafter will mention only the Catalyst 6500.

Note The expression *ASA-based* Firewalls refers to those firewalls that use the *Adaptive Security Algorithm* for stateful inspection purposes. Given that the PIX Firewalls are *end-of-life* products (with no further development), this group is now composed of the FWSM and the ASA appliances.

Overview of IOS-Based Integrated Firewalls

Having discussed the dedicated Cisco Firewall options, it is time to turn your attention to Cisco router-based stateful firewall solutions.

Integrated Services Routers

Parameters like flexibility (in terms of connectivity) and service integration capabilities are typically more relevant than raw throughput for *places in the network* such as remote branches. Cisco IOS router-based Firewalls fit quite well in these scenarios as part of the all-in-one solution provided by its Integrated Services Routers (ISR):

- **Multiprotocol routing:** Not only the classic routing protocols but also innovative solutions such as performance routing are available.
- **Stateful inspection with high level of application awareness:** Either on the Classic IOS Firewall or the Zone-based Policy Firewall (ZFW). The Cisco current recommendation for IOS-based Firewall deployments is the ZFW. These two IOS-based firewall options are covered in Chapter 9, “Classic IOS Firewall Overview” and Chapter 10, “IOS Zone Policy Firewall Overview.”
- **Wide spectrum of secure connectivity models:** Cisco is not limited to traditional IPsec and offers solutions such as EasyVPN, Dynamic Multipoint VPN, and GET VPN.
- **Integrated intrusion prevention:** Either the software-only solution provided by IOS IPS or a dedicated network module that runs the same software used by a Cisco IPS 4200 appliances or an ASA AIP-SSM Module.
- **Convergence services:** Cisco ISRs can act as voice gateways, call managers (IP-PBXs) or both. ISRs also provide rich Quality of Service (QoS) functionality to fit all service integration demands, mainly over WAN access circuits, which are often bandwidth constrained.
- **Identity services:** These include the user-based firewall solution (Chapter 14, “Identity on Cisco Firewalls”), Layer-3 NAC, or 802.1X (for those routers that support LAN-switching modules).

The high-level ISR G2 performance positioning represented in Figure 2-10 is built upon the services integration philosophy. Instead of presenting only firewall-related numbers, the chart provides the overall performance ranges for concurrent services (including rich media collaboration).

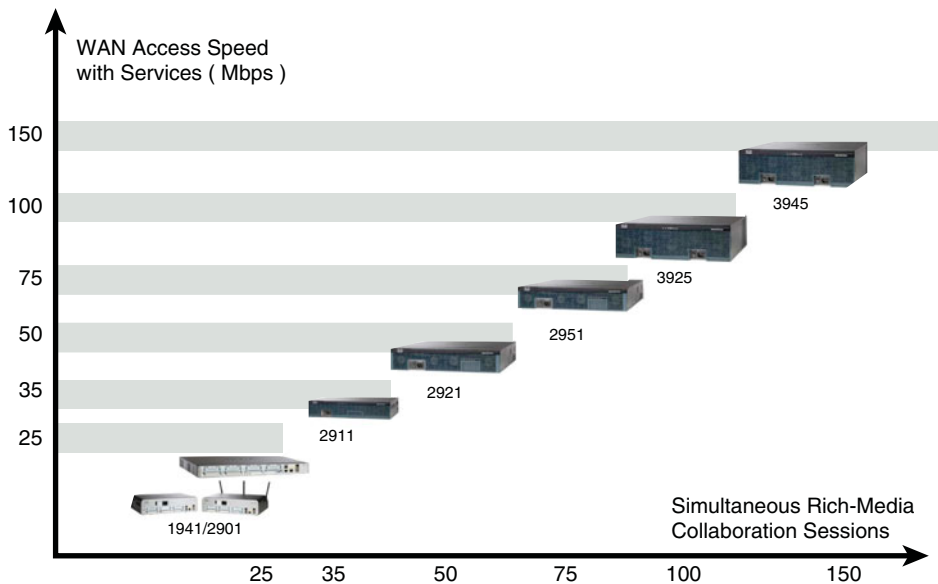


Figure 2-10 A high-Level view of ISR G2 Positioning

Cisco performs detailed tests involving combination of services on its router platforms. Although these results are not public, they may be made available for the specific demands of your project by contacting your Cisco sales representatives. This is certainly of great value for any real-life deployment.

Aggregation Services Routers

Although the ISR G2 routers are positioned for branch deployments, Cisco offers a series of firewall-capable higher-end routers known as the ASR1000 models (*Aggregation Services Routers*). These routers support multiple options of Embedded Services Processor (ESP) modules, which provide hardware-based assistance to sustain high throughput even when advanced features such as stateful firewall, VPN, and Netflow are enabled.

The ASR 1000 models are summarized in Figure 2-11, and the performance numbers for the various ESP processors are shown in Figure 2-12.

ASR 1000 runs IOS-XE and not the original Cisco IOS code. Nonetheless, the CLI are virtually identical for these operating systems.

Some *places in the network*, for which the ASR1000 routers are optimized, follow:

- Central site of large enterprises, acting as a site-to-site VPN concentrator and running the Zone Policy Firewall functionality. The encryption performance of the ESPs is shown in Figure 2-12.

- Central site of large enterprises, deployed as a branch aggregator and configured with stateful firewall (ZFW) and QoS features.
- Service provider sites for aggregation of broadband users with integrated stateful firewall services.

	ASR 1002-F	ASR 1002	ASR 1004	ASR 1006	ASR 1013
					
SPA Slots	1-slot	3-slot	8-slot	12-slot	24-slot
ESP Slots	Integrated	Integrated	1	2	2
SIP Slots	Integrated	Integrated	2	3	6
IOS Redundancy	Software	Software	Software	Hardware	Hardware
Built-in GE	4	4	N/A	N/A	N/A
Height	3.5" (2RU)	3.5" (2RU)	7" (4RU)	10.5" (6RU)	22.7" (13RU)
Throughput	2.5 Gbps	5-10 Gbps	10-20 Gbps	10-40 Gbps	40+ Gbps

Figure 2-11 *High-Level Models Comparison for the Cisco ASR 1000 Routers*

	ESP-2.5G	ESP-5G	ESP-10G	ESP-20G	ESP-40G
System Bandwidth	2.5Gbps	5Gbps	10Gbps	20Gbps	40Gbps
Performance (PPS)	3M	8M	17M	24M	24M
# of Processors	10	20	40	40	40
Clock Rate	900 Mhz	900 Mhz	900 Mhz	1.2 GHz	1.2 GHz
Crypto Engine BW (1400 bytes/packet)	1Gbps	1.8Gbps	4.4Gbps	8.5Gbps	11Gbps
QFP Resource Memory	256MB	256MB	512MB	1GB	1GB
Packet Buffer	64MB	64MB	128MB	256MB	256MB
Control CPU	800 MHz	800 MHz	800 MHz	1.2 GHz	1.8 GHz
Control Memory	1GB	1GB	2GB	4GB	8GB
TCAM	10MB	10MB	10MB	40MB	40MB
Chassis Support	ASR 1002-F (Integrated)	ASR 1002	ASR 1002, 1004, 1006	ASR 1004, 1006	ASR 1006, 1013

Figure 2-12 *High-Level Performance Comparison for the ESP Modules on the ASR1000*

Figure 2-13 shows the hardware overview for the ASR1006, a model that supports redundant ESPs and Route Processors (RP). In the figure, SPA means *Shared Port Adapter* and corresponds to interface modules, and SIP stands for *SPA Interface Processor* (the modules that host SPAs). If you need detailed information about other ASR models, refer to the Cisco online resources.

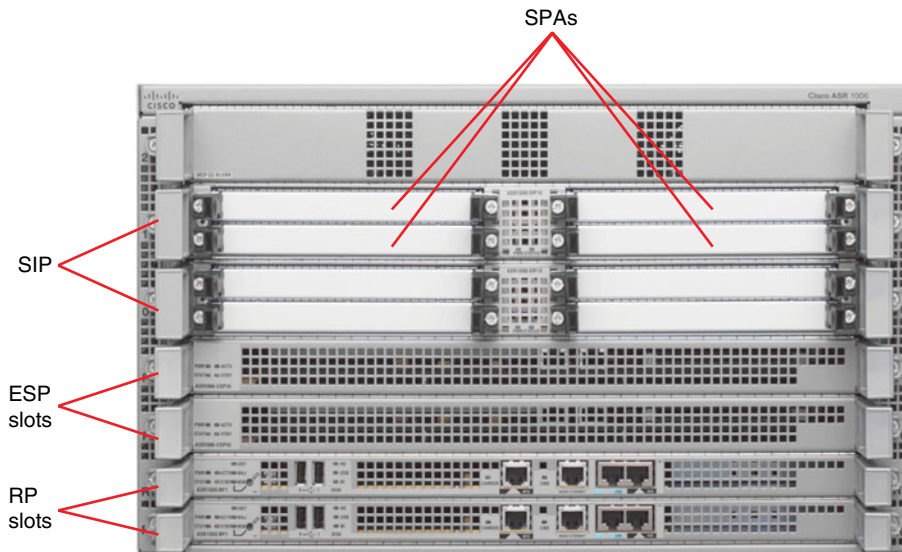


Figure 2-13 *Hardware Overview for the ASR1006 Model*

Summary

This chapter outlined the hardware platforms that host Cisco stateful firewall solutions and discussed the most relevant parameters to be taken into consideration when selecting a firewall model. Because Cisco is frequently offering newer options of firewall hardware, it is convenient to stay tuned for product launches.

The software features corresponding to the various protection mechanisms provided by Cisco Firewalls are the subject of the next chapters.

This page intentionally left blank

Configuration Fundamentals

This chapter covers the following topics:

- Device access using the CLI
- Basic ASA configuration
- Basic FWSM configuration
- Remote management access to ASA and FWSM
- IOS Baseline configuration
- Remote management access to IOS devices
- Clock synchronization using NTP
- Obtaining an IP address through the PPPoE client
- DHCP services

“All rising to great places is by a winding stair.”—Francis Bacon

After the introductory lessons of the first two chapters, it is time to begin the practical work with the Cisco Classic Network Firewalls. This chapter focuses on topics such as IP address assignment, Command Line Interface (CLI) usage and how to prepare the devices to be remotely managed using protocols such as Telnet, Secure Shell (SSH) and HTTPS.

The contents presented are simple, so if you are already familiar with Cisco Classic Firewalls, you can skip this chapter altogether. If you are just beginning, this chapter's topics are relevant and helpful.

Device Access Using the CLI

Even when planning to manage a Cisco Firewall using a Graphical User Interface (GUI), you probably need to take some initial configuration steps via the CLI. The good news, in this case, is that intelligible and intuitive CLIs have always been a recognized asset of Cisco devices. The CLI is typically accessible through a serial console port or by means of terminal access protocols such as Telnet and SSH. In either situation, a terminal emulation program such as *TeraTerm*, *Putty*, or *HyperTerminal* is necessary.

Throughout the book, unless otherwise stated, CLI access is always assumed.

Tip The parameters for the CLI serial (physical) connection are frequently represented as **9600-8-N-1**, meaning 9600 bits per second, 08 data bits, no parity, and 1 stop bit.

Basic ASA Configuration

Before dealing with any specific configuration procedure for the Adaptive Security Appliance (ASA), you need to understand a set of basic concepts. Example 3-1 shows a summary of the boot process for an ASA 5505 appliance whose factory settings have not been changed yet. Following a successful load of the OS image, a prompt offering an *interactive preconfiguration* of the device using menus is presented. You do not need to analyze this option because it provides little flexibility.

The initial prompt after boot completion is `ciscoasa>`, in which *ciscoasa* is the default **hostname** for the equipment. The symbol `>` characterizes that the *EXEC* or *nonprivileged* mode is in place, meaning that a limited set of tasks can be accomplished.

The use of the *question mark* (`?`), as illustrated in Example 3-2, shows the available commands in a given CLI mode. Typing `?` after a command, such as `show ?`, displays the supported parameters for this command. This CLI help is useful, quickly becoming part of everyday life for anyone who works with Cisco equipment.

Example 3-1 Summary Boot Sequence for ASA

```
Evaluating BIOS Options ...
Launch BIOS Extension to setup ROMMON
Cisco Systems ROMMON Version (1.0(12)6) #0: Mon Aug 21 19:34:06 PDT 2006
Platform ASA5505
Use BREAK or ESC to interrupt boot.
Use SPACE to begin boot immediately.
Launching BootLoader...
Boot configuration file contains 1 entry.
Loading disk0:/asa821-k8.bin... Booting...
Platform ASA5505
Loading...
[output suppressed]
```

```

This platform has an ASA 5505Security Plus license.
Encryption hardware device : Cisco ASA-5505 on-board accelerator (revision 0x0)
                                Boot microcode   : CN1000-MC-BOOT-2.00
                                SSL/IKE microcode: CNLite-MC-SSLm-PLUS-2.03
                                IPSec microcode  : CNLite-MC-IPSECm-MAIN-2.04
Cisco Adaptive Security Appliance Software Version 8.2(1)
[output suppressed]
Cryptochecksum (changed): d41d8cd9 8f00b204 e9800998 ecf8427e
Pre-configure Firewall now through interactive prompts [yes]? No

Type help or '?' for a list of available commands.
ciscoasa>

```

Example 3-2 *Commands Available on ASA exec (Nonprivileged) Mode*

```

! Displaying all the commands available on the EXEC mode with a "?"
ciscoasa> ?
clear          Reset functions
enable        Turn on privileged commands
exit           Exit from the EXEC
help           Interactive help for commands
login          Log in as a particular user
logout         Exit from the EXEC
no             Negate a command or set its defaults
ping           Send echo messages
quit           Exit from the EXEC
show           Show running system information
traceroute     Trace route to destination

!
! Viewing available options for the show command (while on EXEC mode)
ciscoasa> show ?
checksum       Display configuration information cryptochecksum
curpriv        Display current privilege level
disk0:         Display information about disk0: file system
flash:         Display information about flash: file system
history        Display the session command history
inventory      Show all inventory information for all slots
power          Power attributes
version       Display system software version

```

Example 3-3 displays summary information obtained with the **show version** command. The data returned after its execution includes OS *version* and hardware components, licensed features, the serial number, and even the uptime since the last reboot.

The **enable** command with a *BLANK* password (for a device with no initial configuration) provides access to the *privileged* mode, recognized by the symbol # after the device **hostname**. This is illustrated in Example 3-4, which also shows the access to *config mode* through the **configure terminal** command. Some of the specific *config modes* (**aaa**, **interface**, **ip**, **router**, and so on) are included in the example.

Note At this point, you are encouraged to sequentially explore the help (?) options available for some of the existent *config modes*. In a similar fashion, it is also worth it to start playing with the **show** command parameters accessible from the ASA *privileged* mode.

Note It is naturally advisable to use the **enable password** command (at *configuration level*) to change the default *BLANK* password into a new one defined by the device administrator. This simple action restricts access to the *privileged mode* and consequently the right to issue any configuration command.

Example 3-3 Sample show version Command for ASA

```
! Displaying basic information about device hardware and software
```

```
ciscoasa>show version
```

```
Cisco Adaptive Security Appliance Software Version 8.2(1)
```

```
Device Manager Version 6.2(1)
```

```
Compiled on Tue 05-May-09 22:45 by builders
```

```
System image file is "disk0:/asa821-k8.bin"
```

```
Config file at boot was "startup-config"
```

```
ciscoasa up 17 mins 10 secs
```

```
Hardware: ASA5505, 256 MB RAM, CPU Geode 500 MHz
```

```
Internal ATA Compact Flash, 128MB
```

```
BIOS Flash M50FW080 @ 0xffe00000, 1024KB
```

```
[output suppressed]
```

```
Licensed features for this platform:
```

```
Maximum Physical Interfaces : 8
```

```
VLANs : 20, DMZ Unrestricted
```

```
Inside Hosts : Unlimited
```

```
Failover : Active/Standby
```

```
VPN-DES : Enabled
```

```
VPN-3DES-AES : Enabled
```

```
SSL VPN Peers : 10
```

```
Total VPN Peers : 25
```

```
Dual ISPs : Enabled
```

```
VLAN Trunk Ports : 8
```

```
[output suppressed]
```

```
Total UC Proxy Sessions : 2
```

```
Botnet Traffic Filter : Disabled
```

```

This platform has an ASA 5505 Security Plus license.
Serial Number: JMX1144Z1AK
Running Activation Key: 0x62306449 0x981618fd 0x2cf05948 0xaf78e074 0x481dedb7
Configuration register is 0x1
Configuration has not been modified since last system restart.

```

Example 3-4 *Moving to Privileged Mode and Viewing Configuration Options*

```

! Moving from EXEC mode to privileged mode (ENABLE)
ciscoasa> enable
Password:
ciscoasa#
!
! Entering configuration (config) mode and displaying available options
ciscoasa# configure terminal
ciscoasa(config)# ?
    aaa                                Enable, disable, or view user authentication,
authorization and accounting
    aaa-server                         Configure a AAA server group or a AAA server
    access-group                       Bind an access-list to an interface to filter
traffic
    access-list                       Configure an access control element
[output suppressed]
    interface                         Select an interface to configure
    ip                                Configure IP addresses, address pools, IDS, etc
    ipsec                             Configure transform-set and IPsec SA lifetime
    ipv6                              Global IPv6 configuration commands
[output suppressed]
    route                             Configure a static route for an interface
    route-map                         Create route-map or enter route-map configura-
tion mode
    router                           Enable a routing process
    routing                          Configure interface specific unicast routing
parameters
    same-security-traffic             Enable same security level interfaces to communi-
cate
[output suppressed]
    wccp                             Web-Cache Coordination Protocol Commands
    webvpn                           Configure the WebVPN service
    zonelabs-integrity               ZoneLabs integrity Firewall Server Configuration
ciscoasa(config)#

```

Note The access to the console port can be controlled with the **aaa authentication serial console LOCAL** command, in which the keyword **LOCAL** means that the local user data-

base is used for validation. Local users are defined with the **username** command, whose usage is exemplified in the “Remote Management Access to ASA and FWSM” section. Other user databases are analyzed in Chapter 14, “Identity on Cisco Firewalls.”

Example 3-5 illustrates the usage of some CLI output filters (all of them are *case-sensitive*), which constitutes a useful resource. The first sample uses the **| begin** filter and instructs the OS to start displaying the line of configuration (or **show** command) where the keyword being searched (**snmp** in this case) first appears.

The second exemplified filter uses the **| include** option, which determines that all lines of the command output that include a match for the searched string should be displayed.

The **show running-configuration** command displays the active configuration of the device and typically results in a large amount of data. More recent versions of ASA OS enable the output of this command to be broken in configuration blocks related to a specific topic. Example 3-5 illustrates how to employ this resource to restrict the output only to the commands related to **timeout** information.

Note Another classic filter is the combination **| exclude**, which displays only the lines that do not contain the searched string. Even more complex regular expressions may also be constructed, and a whole chapter could be devoted to this subject, which is not the plan for this book. The book (starting at this chapter) brings numerous examples of practical usage, but an investigative spirit is the main asset that you must possess to benefit from these powerful resources.

Example 3-5 Using CLI Output Filters

```
! Establishing the point where the display should begin
ciscoasa# show running-config | begin snmp
no snmp-server location
no snmp-server contact
snmp-server enable traps snmp authentication linkup linkdown coldstart
crypto ipsec security-association lifetime seconds 28800
crypto ipsec security-association lifetime kilobytes 4608000
telnet timeout 5
!
! Searching for all matches of a certain string (case sensitive)
ciscoasa# show running-config | include icmp
icmp unreachable rate-limit 1 burst-size 1
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 icmp 0:00:02
!
! Delimiting the section of the running configuration to be displayed
ciscoasa# show running-config timeout
timeout xlate 3:00:00
```

```

timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 icmp 0:00:02
timeout sunrpc 0:10:00 h323 0:05:00 h225 1:00:00 mgcp 0:05:00 mgcp-pat 0:05:00
timeout sip 0:30:00 sip_media 0:02:00 sip-invite 0:03:00 sip-disconnect 0:02:00
timeout sip-provisional-media 0:02:00 uauth 0:05:00 absolute
timeout tcp-proxy-reassembly 0:01:00

```

Note A useful feature of the ASA CLI is the completion of a command (as long as it is not ambiguous) that can be achieved using the <Tab> key. It makes life a lot easier by reducing the time you spend typing commands. For example, although **show in** produces an ambiguous command and a correspondent *ERROR message*, **show int**, followed by the <Tab> key is enough to unequivocally identify the **show interface** command.

```

ciscoasa# show in
ERROR: % Ambiguous command: "show in"
ciscoasa# show int<Tab>
ciscoasa# show interface

```

The suggestion at this point is to practice! (It is certainly worth the investment.)

Basic Configuration for ASA Appliances Other Than 5505

Having powered up an ASA appliance and knowing the basics about command execution modes, it is time to examine some of the fundamental interface configuration tasks:

- Enter interface configuration mode and enable the interface for transmitting and receiving traffic: This mode is indicated by the prompt (config-if)#. At this phase the **no shutdown** command needs to be issued to remove the interface from the administratively down state. If logical subinterfaces have been configured, the pertinent VLAN information (to match that of the switch to which the firewall connects to) should be added. You can also change the speed and duplex.
- Assign a logical name to the interface: This is accomplished with the **nameif** command, and the configured name is used in any future reference to the interface. (It is typically easier to remember the logical meaning of an interface than the physical.)
- Assign a security-level to the interface: To reflect the degree of trustworthiness of a given firewall interface, Cisco introduced in the early days of the PIX Firewalls the concept of Security Level. The value of the Security Level (or simply **sec-lvl**) ranges from 0 to 100, in which the highest possible value, 100, is used for the most trusted network under the firewall control, which is, by default, called inside. Conversely, the name outside is reserved for the less trusted network (frequently the connection to the Internet) and, by default, the value 0 is assigned to its **sec-lvl**.
- Assign an IP Address to the interface: You can do this either through static or dynamic means using the **ip address** command.

Figure 3-1 shows the physical and logical reference topologies for the analysis of an ASA 5510 basic setup. Example 3-6 assembles the typical configuration commands, and Example 3-7 displays some commands to obtain information about the ASA interfaces.

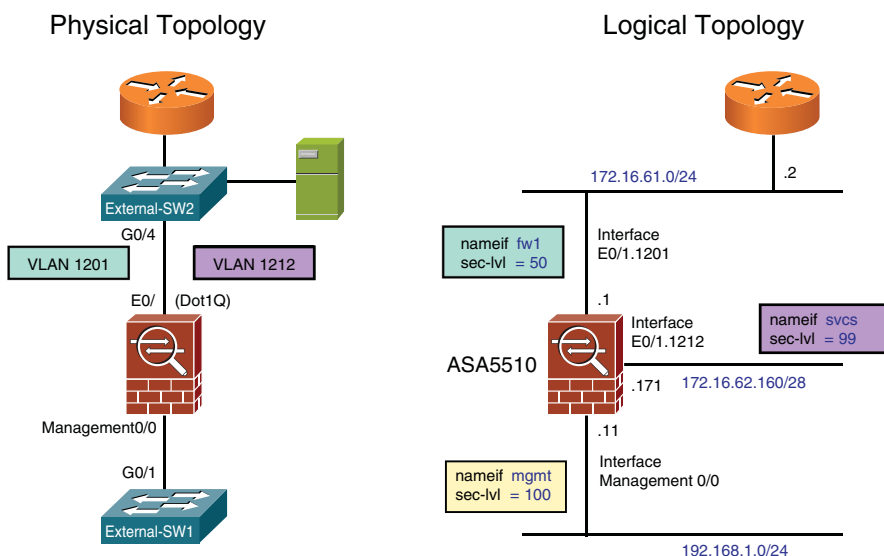


Figure 3-1 Sample Topology Using an ASA 5510 Appliance

Example 3-6 Baseline ASA5510 Configuration

```
! Dedicated Management Interface
interface Management0/0
  nameif mgmt
  security-level 100
  ip address 192.168.1.11 255.255.255.0
  no management-only
  no shutdown
!
! Physical Interface
interface Ethernet0/1
  no nameif
  no security-level
  no ip address
  no shutdown
!
! Creating Subinterfaces on interface E0/1 (two logical networks)
interface Ethernet0/1.1201
  vlan 1201
```

```

nameif fw1
security-level 50
ip address 172.16.61.1 255.255.255.0
!
interface Ethernet0/1.1212
  vlan 1212
  description *** Direct Access to Services Segment ***
  nameif svcs
  security-level 99
  ip address 172.16.62.171 255.255.255.240

```

Example 3-7 Viewing Information About ASA Interfaces

```

! Viewing logical interfaces and correspondent security levels
ASA1# show nameif

```

Interface	Name	Security
Ethernet0/1.1201	fw1	50
Ethernet0/1.1212	svcs	99
Management0/0	mgmt	100

```

!
! Summary IP Addressing information for the interfaces in use
ASA1# show interface ip brief

```

Interface	IP-Address	OK?	Method	Status
Ethernet0/1	unassigned	YES	unset	up
Ethernet0/1.1201	172.16.61.1	YES	CONFIG	up
Ethernet0/1.1212	172.16.62.171	YES	CONFIG	up
Management0/0	192.168.1.11	YES	CONFIG	up

```

!
! Displaying information about a physical interface
ASA1# show interface e0/1
Interface Ethernet0/1 "", is up, line protocol is up
  Hardware is i82546GB rev03, BW 1000 Mbps, DLY 10 usec
    Auto-Duplex(Full-duplex), Auto-Speed(1000 Mbps)
    Available but not configured via nameif
    MAC address 0014.6a21.b4ef, MTU not set
    IP address unassigned
    0 packets input, 0 bytes, 0 no buffer
    Received 0 broadcasts, 0 runs, 0 giants
    0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
    0 L2 decode drops
    2 packets output, 128 bytes, 0 underruns
    0 output errors, 0 collisions, 0 interface resets
    0 late collisions, 0 deferred

```

```

    0 input reset drops, 0 output reset drops, 0 tx hangs
    input queue (blocks free curr/low): hardware (255/255)
    output queue (blocks free curr/low): hardware (255/253)
!
! Displaying information about a subinterface
ASA1# show interface e0/1.1212
Interface Ethernet0/1.1212 "svcs", is up, line protocol is up
  Hardware is i82546GB rev03, BW 1000 Mbps, DLY 10 usec
    VLAN identifier 1212
    Description: *** Direct Access to Services Segment ***
    MAC address 0014.6a21.b4ef, MTU 1500
    IP address 172.16.62.171, subnet mask 255.255.255.240
  Traffic Statistics for "svcs":
    0 packets input, 0 bytes
    1 packets output, 28 bytes
    0 packets dropped
!
! Testing the reachability of local hosts with the ping command
ASA1# ping 172.16.61.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.61.2, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms

```

Basic Configuration for the ASA 5505 Appliance

The ASA 5505, the smallest available model at the time this book was written, comes with an embedded Ethernet switch and has some particularities regarding the initial setup. The other ASA models have only routed interfaces. The operational aspects that deserve special attention are discussed in this section.

Figure 3-2 shows a sample topology that serves as the base for exploring the ASA 5505 platform. Physical interfaces can be configured in either *access* or *trunk* mode, as registered in Example 3-8. (This is much like any Cisco Catalyst switch configuration.) Each VLAN allowed on the IEEE 802.1Q trunk must be explicitly defined at the physical interface level (**switchport trunk allowed vlan** command), and the correspondent logical attributes (*nameif*, *sec-lvl*, and so on) are configured under the associated **Interface VLAN**.

Note An **Interface VLAN** is also known as a *Switched Virtual Interface (SVI)* and represents the Layer 3 counterpart of the L2 VLAN. On Catalyst switches, the L2 portion must be defined by means of the **Vlan** command. On ASA, this command is not available, and the assignment of a port to a VLAN (**switchport access vlan** command) or the VLAN addition to a Dot1Q trunk already creates the L2 definition.

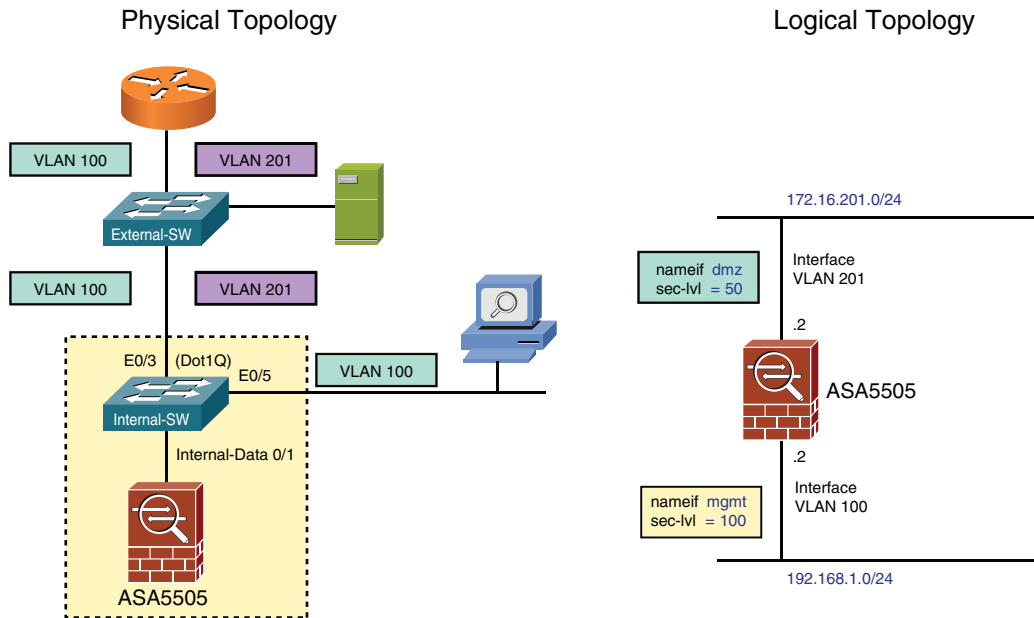


Figure 3-2 Sample Topology Using an ASA 5505 Appliance

Example 3-9 relates to Figure 3-2 and documents the commands employed to verify the existent VLANs and their assigned ports. The typical outputs of the **show interface** options (*physical interface* and *interface vlan*) are registered in this example, which also displays a changed **hostname** for the appliance (ASA 5505 instead of the default *ciscoasa*).

Example 3-8 Baseline ASA 5505 Configuration

```
! Physical Interface operating as an Access Port
interface Ethernet0/5
  switchport access vlan 100
  no shutdown
!
! Physical Interface operating as a 802.1Q (Dot1Q) Trunk Port
interface Ethernet0/3
  switchport trunk allowed vlan 100,201
  switchport mode trunk
  no shutdown
!
! Logical Interface associated with VLAN 100
interface Vlan100
  description *** Management Interface
```



```

nameif mgmt
security-level 100
ip address 192.168.1.2 255.255.255.0
no shutdown
!
! Logical Interface associated with VLAN 201
interface Vlan201
description *** DMZ Network
nameif dmz
security-level 50
ip address 172.16.201.2 255.255.255.0
no shutdown

```

Example 3-9 Viewing Information About ASA 5505 Interfaces

```

! VLAN Assignment on ASA5505 internal switch (L2 information)
ASA5505# show switch vlan

```

VLAN	Name	Status	Ports
1	-	down	Et0/0, Et0/1, Et0/2, Et0/4 Et0/6, Et0/7
100	mgmt	up	Et0/3 , Et0/5
201	dmz	up	Et0/3

```

!
! Displaying information about a physical interface on ASA5505
ASA5505# show interface e0/5
Interface Ethernet0/5 "", is up, line protocol is up
Hardware is 88E6095, BW 100 Mbps, DLY 100 usec
Auto-Duplex(Full-duplex), Auto-Speed(100 Mbps)
Available but not configured via nameif
MAC address 001e.4a05.2005, MTU not set
IP address unassigned
30 packets input, 1920 bytes, 0 no buffer
Received 30 broadcasts, 0 runts, 0 giants
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
0 L2 decode drops
0 switch ingress policy drops
4 packets output, 256 bytes, 0 underruns
0 output errors, 0 collisions, 0 interface resets
0 late collisions, 0 deferred
0 input reset drops, 0 output reset drops
0 rate limit drops
0 switch egress policy drops
!

```

! Displaying information about a VLAN interface on ASA5505

```
ASA5505# show interface vlan 201
Interface Vlan201 "dmz", is up, line protocol is up
  Hardware is EtherSVI, BW 100 Mbps, DLY 100 usec
    MAC address 001e.4a05.2008, MTU 1500
    IP address 172.16.201.2, subnet mask 255.255.255.0
  Traffic Statistics for "dmz":
    0 packets input, 0 bytes
    0 packets output, 0 bytes
    0 packets dropped
  1 minute input rate 0 pkts/sec,  0 bytes/sec
  1 minute output rate 0 pkts/sec,  0 bytes/sec
  1 minute drop rate, 0 pkts/sec
  5 minute input rate 0 pkts/sec,  0 bytes/sec
  5 minute output rate 0 pkts/sec,  0 bytes/sec
  5 minute drop rate, 0 pkts/sec
```

Note The configuration commands issued on the CLI are stored in the RAM (as the **running-config**) and immediately become active. To be available after a router reboot, these commands need to be moved to the **startup-config** (stored in nonvolatile RAM or, briefly, NVRAM). This is accomplished with the command **copy running-config startup-config** or with the well-known old version **write memory**.

Basic FWSM Configuration

Before having access to the Firewall Services Module (FWSM), you need to perform some configurations on the Catalyst 6500 chassis where it resides.

Example 3-10 teaches how to locate a FWSM in a given 6500 chassis and verify the status of the module using the **show module** command. It also shows the *Etherchannel* connection (consisting of six Gigabit Ethernet ports) to the Switching Fabric. (You can see the logical representation of the Etherchannel connection in Figure 3-3.)

Example 3-10 Viewing Information About Modules on a Catalyst 6500

! Displaying Information about installed modules on a Catalyst 6500 switch

```
CAT6500B# show module
```

Mod	Ports	Card Type	Model	Serial No.
1	48	CEF720 48 port 10/100/1000mb Ethernet	WS-X6748-GE-TX	SAL1026SYKR
4	6	Firewall Module	WS-SVC-FWM-1	SAD11270BNW
5	2	Supervisor Engine 720 (Active)	WS-SUP720-3B	SAL1015JH6H

Mod	MAC addresses	Hw	Fw	Sw	Status

```
-----
1 0017.5916.59b8 to 0017.5916.59e7 2.4 12.2(14r)S5 12.2(18)SXF1 Ok
4 001b.d59c.0ce0 to 001b.d59c.0ce7 4.2 7.2(1) 4.0(3) 0k
5 0013.c43a.ced8 to 0013.c43a.cedb 5.2 8.4(2) 12.2(18)SXF1 Ok

Mod Sub-Module Model Serial Hw Status
-----
1 Centralized Forwarding Card WS-F6700-CFC SAD102308FL 2.0 Ok
5 Policy Feature Card 3 WS-F6K-PFC3B SAL1015JHTB 2.3 Ok
5 MSFC3 Daughterboard WS-SUP720 SAL1010F7PX 2.5 Ok

Mod Online Diag Status
-----
1 Pass
4 Pass
5 Pass
!
! Verifying Etherchannel information for the FWSM
CAT6500B# show etherchannel summary
Flags: D - down P - bundled in port-channel
I - stand-alone s - suspended
H - Hot-standby (LACP only)
R - Layer3 S - Layer2
U - in use f - failed to allocate aggregator
M - not in use, minimum links not met
u - unsuitable for bundling
w - waiting to be aggregated
Number of channel-groups in use: 1
Number of aggregators: 1

Group Port-channel Protocol Ports
-----+-----+-----+-----
273 Po273(SU) - Gi4/1(P) Gi4/2(P) Gi4/3(P) Gi4/4(P)
Gi4/5(P) Gi4/6(P)
```

Example 3-11 shows the baseline configuration tasks that should be accomplished on the Catalyst 6500 before using the FWSM. These tasks include the following:

- **Creating VLANs:** Using exactly the same procedure used for any other VLAN.
- **Creating VLAN Groups:** Instead of directly assigning VLANs to the FWSM, the configuration uses VLAN-Groups.

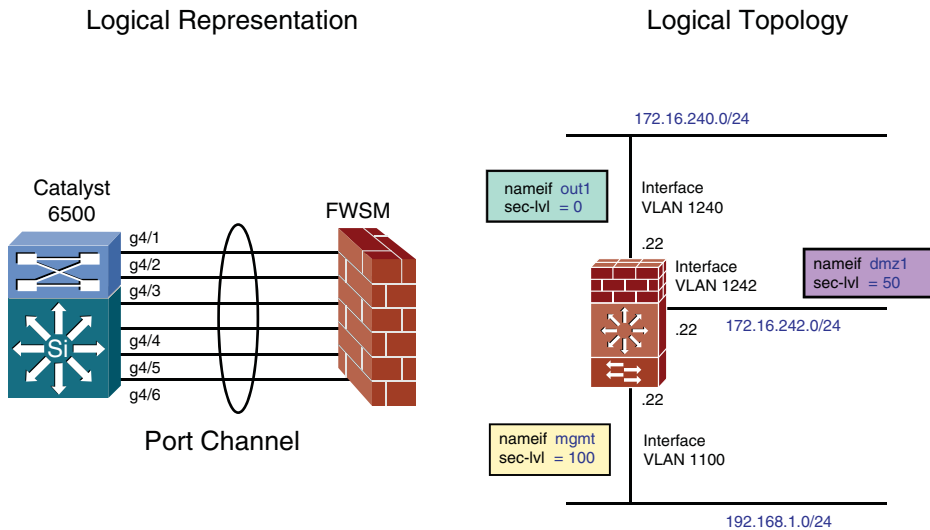


Figure 3-3 *Logical Representation and Logical Topology for FWSM Analysis*

- **Associating VLAN Groups with the physical module:** Only the VLAN Groups defined with the firewall module **vlan-group** command become visible in the FWSM. The Catalyst 6500 behaves as a regular multilayer switch for all the VLANs that were not explicitly assigned to the services module.

Note The VLAN-Groups for the FWSM can be created using either the **firewall vlan-group** or the **svclc vlan-group** command. VLAN-Groups created using the **svclc vlan-group** command can be associated not only with the FWSM but also with a Cisco Application Control Engine (ACE) services module. (This is required when an ACE module resides in the same chassis and needs to communicate with the FWSM.) When the first method is used, the VLAN-Groups are not allowed to be associated to a non-FWSM module.

Example 3-11 also registers the commands to verify the VLAN and VLAN-Group information (related to the FWSM) in the Catalyst 6500 chassis.

Example 3-11 *Baseline Configuration for the Catalyst 6500*

```
! Creating VLANs on the Catalyst 6500
vlan 1100
  name SEC-MGMT
!
vlan 1240
  name FWSM-OUT1
!
vlan 1242
  name FWSM-DMZ1
```

```

!
! Creating VLAN Groups (SVCLC = Services Line Card)
svclc vlan-group 1 1100
svclc vlan-group 2 1240,1242
!
! Assigning VLAN Groups to the Firewall Module (installed in slot 4)
firewall module 4 vlan-group 1,2
!
! Verifying VLAN and VLAN-Group information
CAT6500B# show firewall vlan-group
Display vlan-groups created by both ACE module and FWSM
Group      Created by      vlans
-----
1          ACE          1100
2          ACE          1240,1242
!
CAT6500B# show firewall module 4 state
Firewall module 4:
Switchport: Enabled
Administrative Mode: trunk
Operational Mode: trunk
Administrative Trunking Encapsulation: dot1q
Operational Trunking Encapsulation: dot1q
Negotiation of Trunking: Off
Access Mode VLAN: 1 (default)
Trunking Native Mode VLAN: 1 (default)
Trunking VLANs Enabled: 1100,1240,1242
Pruning VLANs Enabled: 2-1001
Vlans allowed on trunk: 1100,1240,1242
Vlans allowed and active in management domain: 1100,1240,1242
Vlans in spanning tree forwarding state and not pruned:
1100,1240,1242

```

Example 3-12 shows the procedure for getting access to the FWSM that resides in module 4, from the Catalyst 6500 console. This access is actually a Telnet connection that uses a reserved *loopback* address (belonging to network 127.0.0.0/8). This example also displays the source and destination IP addresses and L4 ports for the Telnet session.

Example 3-12 Accessing the FWSM from the Catalyst 6500 Console

```

CAT6500B# session slot 4 processor 1
The default escape character is Ctrl-^, then x.
You can also type 'exit' at the remote prompt to end the session
Trying 127.0.0.41... Open

```

```

User Access Verification
Password:*****
Type help or '?' for a list of available commands.
FWSM2> enable
Password: *****
FWSM2#
!
! Viewing the Telnet connection from the Catalyst 6500 to the FWSM
CAT6500B# show tcp brief

```

TCB	Local Address	Foreign Address	(state)
46DCBBD0	127.0.0.51.38778	127.0.0.41.23	ESTAB

Figure 3-3 displays the logical representation of the FWSM *Etherchannel* connection to the Catalyst backplane. It also shows the logical topology that serves as the base for the analysis of the configuration fundamentals related to the FWSM.

Example 3-13 refers to the topology on Figure 3-3 and assembles the fundamental commands for initial FWSM configuration. The FWSM does not have external network interfaces. All its logical interfaces are VLANs created on the underlying chassis and assigned to it through the **firewall module vlan-group** command (refer to Example 3-11).

One important difference between ASA appliances and the FWSM is that Internet Control Message Protocol (ICMP) traffic needs to be explicitly permitted on a per-interface basis (using **icmp permit** commands) on the Firewall Module. Conversely, the default behavior of ASA is to accept ICMP packets directed to its interfaces (refer to Example 3-7).

Example 3-13 Baseline FWSM Configuration

```

! Configuring Logical Interfaces
interface Vlan1100
  description *** Management Access ***
  nameif mgmt
  security-level 100
  ip address 192.168.1.22 255.255.255.0
!
interface Vlan1240
  nameif out1
  security-level 0
  ip address 172.16.240.22 255.255.255.0
!
interface Vlan1242
  nameif dmz1
  security-level 50
  ip address 172.16.242.22 255.255.255.0

```

```

!
! Enabling ICMP Ping to and from logical interfaces
icmp permit any echo mgmt
icmp permit any echo-reply mgmt
icmp permit any echo out1
icmp permit any echo-reply out1
icmp permit any echo dmz1
icmp permit any echo-reply dmz1

```

Example 3-14 assembles some **show** commands that enable the visualization of interface-related information on the FWSM. The VLANs visible on the FWSM side can be seen from the Catalyst 6500's CLI with the aid of the **show firewall** commands presented in Example 3-11.

Example 3-14 *Displaying Information About Interfaces and VLANs on the FWSM*

```

FWSM2# show nameif
Interface          Name          Security
Vlan1100           mgmt          100
Vlan1240           out1          0
Vlan1242           dmz1          50
!
FWSM2# show vlan
1100, 1240, 1242
!
FWSM2# show interface vlan 1100
Interface Vlan1100 "mgmt", is up, line protocol is up
  Hardware is EtherSVI, BW Unknown Speed-Capability, DLY 10 usec
    Description: *** Management Access ***
    MAC address 001b.d4de.3580, MTU 1500
    IP address 192.168.1.22, subnet mask 255.255.255.0
  Traffic Statistics for "mgmt":
    798 packets input, 130180 bytes
    15 packets output, 1270 bytes
    55112 packets dropped

```

Remote Management Access to ASA and FWSM

The examples presented so far have considered that there was physical access to the console port of the appliance (or to the hosting Catalyst 6500 for the FWSM). This section examines management connections that rely on remote access protocols (Telnet, SSH, and HTTPS).

Note This chapter assumes that the *LOCAL* user database is employed to authenticate users who start remote management connections to Firewall devices. Chapter 14 covers the use of AAA services for centralized control of administrative access.

Telnet Access

Telnet is a classic terminal access protocol that has received much criticism because of its *clear text* nature. It is highly recommended you replace it with SSH, which provides confidentiality.

At any rate, Telnet can still be useful for testing purposes mainly during initial setup. The commands shown in Example 3-15 specify the following:

- Telnet access is accepted only when it is initiated from source addresses on network 192.168.1.0/24. Further, the packets must arrive through the logical interface called mgmt.
- The authentication of users who have permission to Telnet to the firewall is done using the LOCAL database. (LOCAL is a reserved keyword for ASA and FWSM.)
- The username admin is included in the LOCAL database.

Example 3-15 also displays a sample Telnet session coming from address 192.168.1.201.

Example 3-15 *Configuring and Verifying Telnet Access*

```
! Creating a local user
username admin password cisco123 privilege 15
!
! Telnet access is authenticated using the LOCAL Data Base
aaa authentication telnet console LOCAL
!
! Defining source addresses that can initiate Telnet Access to the Firewall
telnet 192.168.1.0 255.255.255.0 mgmt
!
! Initiating a Telnet session to ASA
DMZ# telnet 192.168.1.2
Trying 192.168.1.2 ... Open
User Access Verification
Username:admin
Password: *****
Type help or '?' for a list of available commands.
ASA5505>enable
Password: *****
ASA5505#
```



```

!
! Displaying connections to the Firewall
ASA5505# show conn all | include Identity
TCP mgmt 192.168.1.201:35313 NP Identity Ifc 192.168.1.2:23, idle 0:00:00, bytes
1017, flags UOB
!
ASA5505# who
      0: 192.168.1.201

```

SSH Access

If remote CLI access to the firewalls is needed, SSH is the protocol of choice. It provides the same terminal services that Telnet does but with the significant advantage of encrypting traffic between *client* and *server* (the firewall receiving the connection).

Because SSH uses RSA public keys to encrypt the sessions, you need to have consistent timing information. Example 3-16 shows not only how to manually adjust and verify timing information, but also how to create a domain name and generate RSA keys.

Example 3-17 shows how to visualize SSH-related information in the Running-config. Notice that the default timeout value for SSH sessions is 5 minutes.

Note The `clock` command is not available in the FWSM. All timing information in this case comes from the Catalyst switch. The section “Clock Synchronization with NTP” discusses this.

Example 3-16 *Recommended Tasks Before Starting SSH Configuration*

```

! Setting Local Time (before generating the cryptographic keys)
ASA5505# clock set 19:05:00 november 15 2009
!
! Verifying Time Information
ASA5505# show running-config clock
clock timezone BRT -3
!
ASA5505# show clock detail
19:18:00.569 BRT Sun Nov 15 2009
Time source is user configuration
!
! Configuring a domain-name
ASA5505(config)# domain-name mylab.lab
!
! Removing (if needed) any previously generated RSA keys
ASA5505(config)# crypto key zeroize rsa

```

```

WARNING: All RSA keys will be removed.
WARNING: All device digital certificates issued using these keys will also be
removed.
Do you really want to remove these keys? [yes/no]: yes
!
! Generating new RSA Cryptographic Keys
ASA5505(config)# crypto key generate rsa modulus 1024
INFO: The name for the keys will be: <Default-RSA-Key>
Keypair generation process begin. Please wait...
!
! Displaying the RSA Public Keys
ASA5505# show crypto key mypubkey rsa
Key pair was generated at: 19:24:29 BRT Nov 15 2009
Key name: <Default-RSA-Key>
Usage: General Purpose Key
Modulus Size (bits): 1024
Key Data:
30819f30 0d06092a 864886f7 0d010101 05000381 8d003081 89028181 008e60c4
bce3e63a 47aa12c4 e78c0a76 f2faf41c 5d8d461a 4978a5f6 0a4ac11b 26585f61
d6b5adcb f5ce2430 a96c6fb9 d09f2187 3525255a 349e015e 37d0dd79 90e2b2f1
5e968993 b9bb9cde 557ba395 e0b20f7c 0049b0d8 5d901902 fe8269ce 74f06a7f
16713eea 8fe2a0a8 9dde2c3 1d258249 d16e6fc4 5a3b4fb6 be977bbf 55020301 0001

```

Example 3-17 SSH Configuration

```

ASA5505# show running-config | include ssh
aaa authentication ssh console LOCAL
ssh 192.168.1.0 255.255.255.0 mgmt
ssh timeout 5

```

HTTPS Access Using ASDM

The *Adaptive Security Device Manager (ASDM)* is an intuitive and easy-to-use GUI that accompanies every member of the ASA family. The interface provides a nice graphical abstraction for the actual commands that are used not only to implement the features but also to verify their operation, thus allowing users who are already familiar with classic firewall concepts (even from other vendors) to easily adapt their knowledge to the new GUI and immediately start working.

Documenting ASDM usage with its uncountable configuration and monitoring screens is beyond the scope of this book. However, the preparation of firewall devices to accommodate ASDM management is covered.

ASDM uses the HTTPS protocol for communications between the management station and the firewall. After properly loading the ASDM image on the device's *flash* memory, a

web browser can be employed for the first access to the device, with the underlying goal of installing the *ASDM launcher* application on the administrator's PC.

Example 3-18 shows the preliminary tasks for enabling HTTPS access and assumes that the remote user has been granted the highest privilege level (*priv-lvl* = 15) and that the requests arrive through the logical interface called *mgmt*. In the example, the user named *admin* is authenticated against the *LOCAL* database and should start the management session from a host that belongs to the 192.168.1.0/24 subnet. The example also contains information about location of the ASDM image in the device flash (*disk0*: in this case) and how to find it within the **show version** output.

Example 3-18 Enabling HTTPS Access on ASA

```
! Assigning the highest privilege level (15) to the HTTPS user
username admin password ***** privilege 15
!
! Defining allowed source IP Addresses. Authentication using the LOCAL Data Base
http 192.168.1.0 255.255.255.0 mgmt
aaa authentication http console LOCAL
!
! Enabling the HTTPS server
http server enable
!
! Defining the location of the ASDM image
asdm image disk0:/asdm-621.bin
!
! Verifying Operating System and ASDM versions
ASA5505# show version | include Version
Cisco Adaptive Security Appliance Software Version 8.2(1)
Device Manager Version 6.2(1)
```

Figure 3-4 portrays the web browser access to the HTTPS server on the device whose management interface is configured with the address 192.168.1.2 (<https://192.168.1.2>).

From this screen, select the **Install ASDM Launcher and Run ASDM** option and follow these steps:

- Step 1.** Authenticate with the credentials configured in Example 3-18 when the Connect to 192.168.1.2 window displays.
- Step 2.** From the File Download - Security Warning window, save the “.msi” file locally.
- Step 3.** Run the “.msi” file and install the ASDM Launcher application.
- Step 4.** After starting the **ASDM Launcher**, fill in the IP address (192.168.1.2 in this case) and the credentials (username/password).

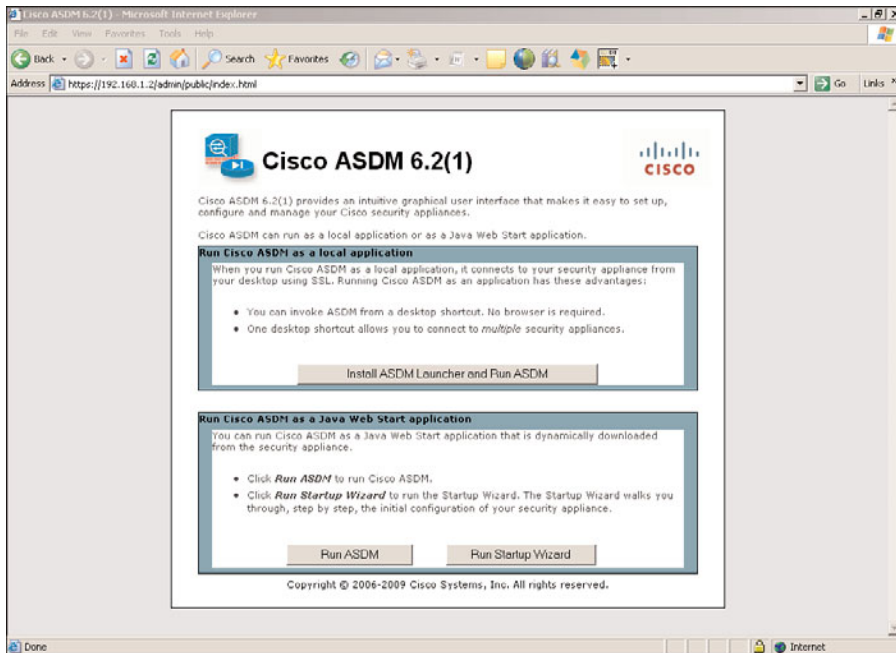


Figure 3-4 First HTTPS Access and Initial ASDM Page

Step 5. After accepting the device certificate, the main ASDM page displays (Figure 3-5). This screen summarizes information for the device, including available licenses, interface status, and system resources status.

Figure 3-6 depicts the base ASDM screen for *Interface Configuration* on an ASA 5505 appliance. Notice that the full path to this particular screen, **Configuration > Device Setup > Interfaces**, displays on the top of the right pane.

Figure 3-7 shows a sample ASDM screen that helps perform the Monitoring task of verifying the ARP table. The complete path for viewing this table is represented at the top of the right pane (**Monitoring > Interfaces > ARP Table**).

Note This was just a brief initiation to the ASDM world, highlighting the preliminary activities that need to be performed to launch the tool. This user-friendly GUI is flexible and useful for the daily duties of a firewall administrator and therefore deserves some serious *hands-on* practice.

At this point, simply start navigating through the GUI to get a high level overview of the available options.

You might also review the CLI examples in this chapter and look for the corresponding options in the GUI.

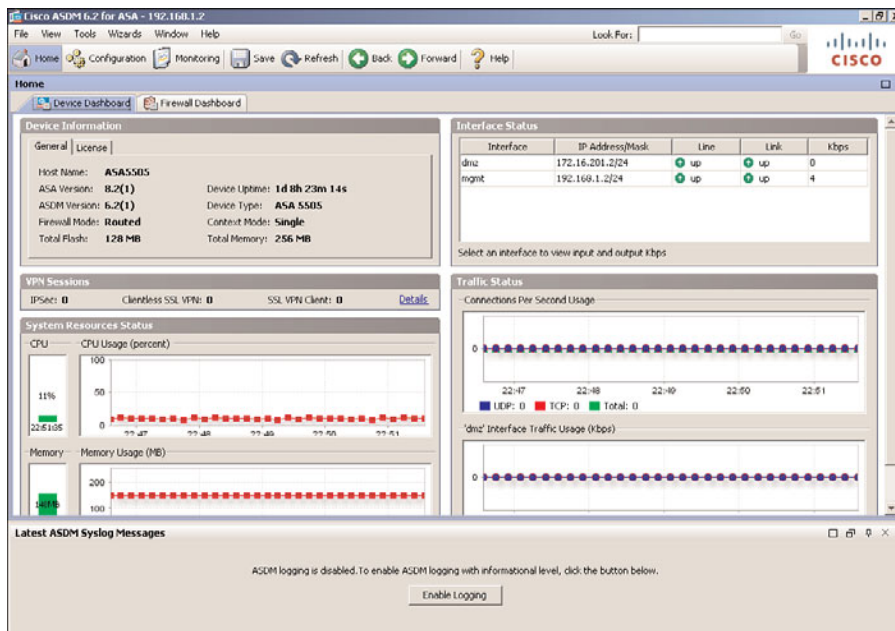


Figure 3-5 ASDM Home Page - Device Dashboard

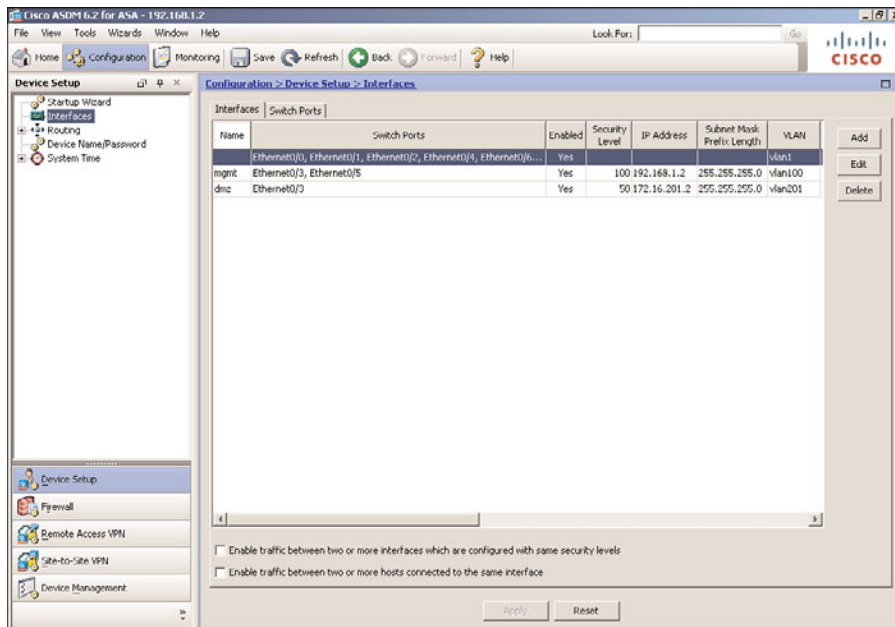


Figure 3-6 Base ASDM Page for Interface Configuration

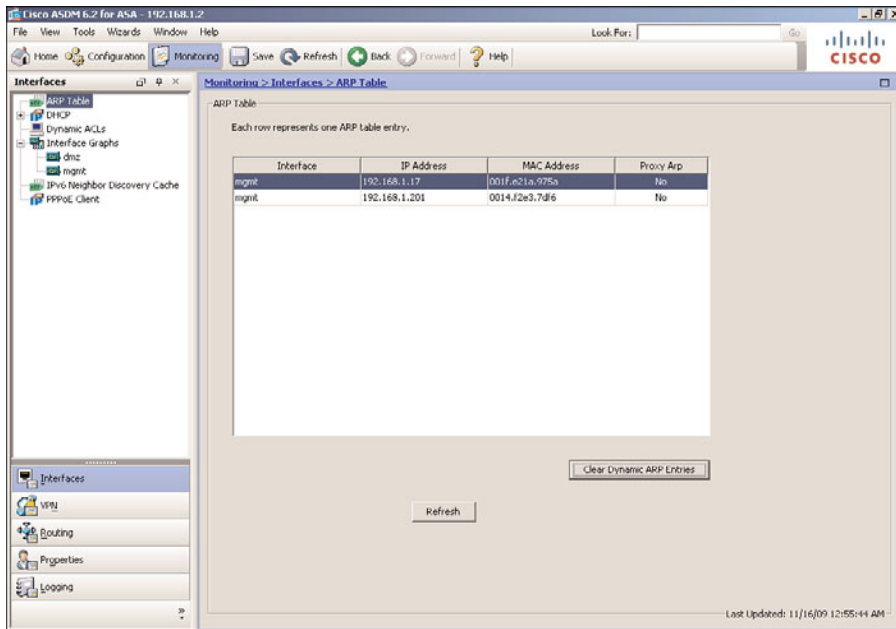


Figure 3-7 Base ASDM Page for ARP Table Monitoring

IOS Baseline Configuration

This section covers some basic concepts of the IOS CLI, most of which are in close proximity to ASA. Example 3-19 displays the summary boot sequence for a router that had no initial configuration. A successful OS image load culminates in the offer of configuring the device through interactive menus (*initial configuration dialog*), which was refused in this case. After that, the default EXEC (*nonprivileged*) **Router>** mode appears. As previously studied for ASA, the default **enable** password, whose usage provides access to **privileged** mode, is **BLANK**, meaning there is *no password* and that you just need to press **Enter**.

Example 3-20, the IOS counterpart of Example 3-3, registers a typical **show version** output for a Cisco router. Some relevant information that can be obtained from this output includes OS version, physical interfaces, and the amount of memory (RAM, Flash, and NVRAM). The last line of the output informs the value of the *Configuration Register*, an important boot control parameter. For instance, an **HEX** value of **2102** for this variable instructs the router to boot using the image stored on its Flash memory and obey what is determined by its *startup-config*, which is saved in the nonvolatile memory (NVRAM).

Example 3-19 Summary Boot Sequence for an IOS Router

```
System Bootstrap, Version 12.3(8r)YI, RELEASE SOFTWARE
Technical Support: http://www.cisco.com/techsupport
```

```

Copyright (c) 2005 by cisco Systems, Inc.
C870 series (Board ID: 3-148) platform with 131072 Kbytes of main memory
Booting flash:/c870-advipservicesk9-mz.124-24.T1.bin
Self decompressing the image :
#####
##### [OK]
[output suppressed]
Cisco IOS Software, C870 Software (C870-ADVIPSERVICESK9-M), Version 12.4(24)T1,
RELEASE SOFTWARE (fc3)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2009 by Cisco Systems, Inc.
Compiled Sat 20-Jun-09 02:20 by prod_rel_team
[output suppressed]
Installed image archive
Cisco 871W (MPC8272) processor (revision 0x100) with 118784K/12288K bytes of
memory.
Processor board ID FHK093310A5
MPC8272 CPU Rev: Part Number 0xC, Mask Number 0x10
5 FastEthernet interfaces
1 802.11 Radio
128K bytes of non-volatile configuration memory.
24576K bytes of processor board System flash (Intel Strataflash)

--- System Configuration Dialog ---
Would you like to enter the initial configuration dialog? [yes/no]: no
Press RETURN to get started!
*Mar 1 00:00:05.579: %VPN_HW-6-INFO_LOC: Crypto engine: onboard 0 State changed
to: Initialized
*Mar 1 00:00:05.583: %VPN_HW-6-INFO_LOC: Crypto engine: onboard 0 State changed
to: Enabled
[output suppressed]
Router>
Router>enable
Router#

```

Example 3-20 Sample show version Command for IOS

```

Router# show version
Cisco IOS Software, C870 Software (C870-ADVIPSERVICESK9-M), Version 12.4(24)T1,
RELEASE SOFTWARE (fc3)
[output suppressed]
ROM: System Bootstrap, Version 12.3(8r)YI, RELEASE SOFTWARE
Router uptime is 2 minutes
System returned to ROM by power-on
System image file is "flash:c870-advipservicesk9-mz.124-24.T1.bin"

```

```
[output suppressed]
Cisco 871W (MPC8272) processor (revision 0x100) with 118784K/12288K bytes of
memory.
Processor board ID FHK093310A5
MPC8272 CPU Rev: Part Number 0xC, Mask Number 0x10
5 FastEthernet interfaces
1 802.11 Radio
128K bytes of non-volatile configuration memory.
24576K bytes of processor board System flash (Intel Strataflash)
Configuration register is 0x2102
Router#
```

Note To reduce the output of the **show running config** command, both IOS and ASA adopt the approach of displaying only *nondefault* options and parameters. In the interest of getting access to default values, you should use the **show running-config all** command.

Note The help (?) option, the command completion using the <Tab> key, and the output filters (**begin**, **include**, **exclude**, and so on) discussed for ASA are equally relevant for IOS. One advantage of ASA in this domain is the possibility of displaying specific sectors of the **show running-config** (refer to the last portion of Example 3-5). The main option available in IOS is **show running-config interface**, which limits the output to the specified interface (physical or logical).

Configuring Interfaces on IOS Routers

IOS interface configuration is simpler than that of ASA-based products, and does not define, for example, information such as **nameif** and **security-level**, two concepts that lie at the core of ASA philosophy. Example 3-21 brings two possibilities for IOS interfaces:

- Configuration of logical parameters directly under the physical interface.
- Configuration of 802.1Q subinterfaces, enabling many logical subnets to be tied to one physical interface.

Example 3-21 Basic Interface Configuration Tasks

```
! Configuring logical attributes directly on the physical interface
interface FastEthernet4
description *** DMZ interface ***
ip address 192.168.1.201 255.255.255.0
speed 100
duplex full
```



```

no shutdown
!
! Creating 802.1Q (Dot1Q) subinterfaces
interface FastEthernet4
  speed 100
  duplex full
  no ip address
  no shutdown
!
interface FastEthernet4.100
  description *** DMZ interface (VLAN 100) ***
  encapsulation dot1q 100
  ip address 192.168.1.201 255.255.255.0

```

Note Subsequent chapters show that when IOS operates as a *Zone-Based Policy Firewall*, the concepts of *zone-security* (which resemble ASA *sec-lvl*) come into play.

Remote Management Access to IOS Devices

The examples analyzed in the previous section assumed local access to the console port of the IOS device. This section looks at remote management connections that rely on protocols such as Telnet, SSH, and HTTPS.

Remote Access Using Telnet

IOS uses the concept of *Virtual Terminal (VTY)* lines to receive connections related to protocols such as Telnet and SSH. The settings entered on a VTY line apply to session requests arriving on any of the router interfaces. The typical VTY-level settings follow:

- **Timeout value:** Defines the inactivity timeout for the terminal lines. The parameters of the `exec-timeout` command are respectively **MINUTES** and **SECONDS**.
- **Password:** The combination of the `password` and `login` commands requires a generic user initiating a Telnet session to the router to inform this password.

Example 3-22 shows the basic parameters concerning VTY configuration and the authentication sequence for a Telnet session. *Privileged mode* access requires an `enable secret` to be configured at the global level.

Example 3-22 VTY Lines for Telnet Access

```

line vty 0 4
  exec-timeout 5 0
  password cisco

```

```

login
!
enable secret 5 $1$k6BB$clMRpv4a6hQ.EmbS0EPJ/
!
! Authentication Experience for a Generic User when using Telnet
User Access Verification
Password:****
R1>enable
Password:*****
R1#
!! Following authentication, the generic user obtains information about the session
R1# show tcp brief

```

TCB	Local Address	Foreign Address	(state)
838F2330	192.168.1.201.23	192.168.1.15.3547	ESTAB

Remote Access Using SSH

The previous section demonstrated the creation of a Telnet session for *generic users*. (Only the password is presented; no user information is entered.) SSH, on the opposite range of the spectrum from Telnet, requires *nongeneric users*, meaning that the *username* is always requested by the device (acting as an SSH server).

Example 3-23 shows a typical sequence of tasks to enable SSH on an IOS device. The preliminary activities of setting and verifying the clock, and verifying the existence of an RSA key-pair, are still recommended and employ exactly the same commands as those analyzed for ASA in Example 3-16. A few points that deserve special attention for IOS follow:

- The `aaa new-model` must be enabled, so that authentication methods for each type of access can be specified. The example shows that users of the VTY lines (Telnet or SSH) are authenticated through the method-list called `TERMINAL-LINES`, which points to the local database.
- Specifying a hostname (distinct from the default name Router) and creating an ip domain-name). These two elements are grouped to generate a name for the RSA key-pair. It is interesting to clearly associate this key-pair to SSH usage as illustrated in the example (`ip ssh keypair-name` command).

Example 3-23 Enabling SSH on IOS

```

! Creating a username in the local database
R1(config)# username user1 password #####
!
! Enabling aaa new-model (allowing an authentication method for each type of access)

```

```

R1(config)# aaa new-model
!
! The authentication method for the VTY lines uses the local database
R1(config)# aaa authentication login TERMINAL-LINES local
R1(config)# line vty 0 4
R1(config-line)# login authentication TERMINAL-LINES
!
! Changing the default host name and creating a domain name
Router(config)# hostname R1
R1(config)# ip domain-name mylab.net
!
! Generating the RSA key pair
R1(config)# crypto key generate rsa
The name for the keys will be: R1.mylab.net
Choose the size of the key modulus in the range of 360 to 2048 for your
General Purpose Keys. Choosing a key modulus greater than 512 may take
a few minutes.
How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]
%SSH-5-ENABLED: SSH 1.99 has been enabled
!
! Determining the key pair to be used for SSH
R1(config)# ip ssh rsa keypair-name R1.mylab.net
!
! Displaying the SSH sessions
R1# show ssh

```

Connection	Version	Mode	Encryption	Hmac	State	Username
0	2.0	IN	aes256-cbc	hmac-sha1	Session started	user1
0	2.0	OUT	aes256-cbc	hmac-sha1	Session started	user1

```

%No SSHv1 server connections running.
!
R1# show tcp brief

```

TCB	Local Address	Foreign Address	(state)
83FC0F0C	192.168.1.201.22	192.168.1.15.3756	ESTAB

Note If the goal is to define the IP addresses entitled to have terminal access to an IOS device, an **access-class** should be applied to the VTY lines. This eliminates the need to configure permissions on the individual router interfaces. The following combination establishes that only source addresses on 192.168.1.0/24 are allowed to access the terminal lines.

```
access-list 1 permit 192.168.1.0 0.0.255
```

```
line vty 0 4
```

```
access-class 1 in
```

Note Taking into consideration that IOS and ASA CLIs are *case-sensitive*, a good practice is to use *uppercase letters* for parameters such as the name of an *aaa method-list*. (Please refer to Example 3-23, in which the name *TERMINAL-LINES* was used for the *login authentication* method list.) This approach clearly helps to differentiate commands from their parameters when displaying the *running-config*.

Remote Access Using HTTP and HTTPS

IOS enables remote access using HTTP and HTTPS; the latter, of course, being preferable. Example 3-24 shows how to enable the HTTP Server and unveils what goes on behind the scenes when the `ip http secure-server` command is issued.

Example 3-25 registers how to control the web connections to the routers both from the user database and source addresses perspectives. (Chapter 14 examines the usage of more sophisticated user databases in a great level of detail.)

Example 3-24 *Enabling HTTP and HTTPS on IOS*

```
! Enabling the HTTP server
R1(config)# ip http server
!
! Enabling the HTTPS Server
R1(config)# ip http secure-server
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]
CRYPTO_PKI: Generating self signed cert TP-self-signed-681151852
CRYPTO_PKI: Creating trustpoint TP-self-signed-681151852
CRYPTO_PKI:Insert Selfsigned Certificate:
[output suppressed]
CRYPTO_PKI: Self signed cert TP-self-signed-681151852 created
%PKI-4-NOAUTOSAVE: Configuration was modified. Issue "write memory" to save new
certificate
!
! Saving the self-signed certificate
R1# write memory
Building configuration...
PKI: Removing old cert file nvram:IOS-Self-Sig#6.cer
crypto_ca_certificate: saved cert to nvram:IOS-Self-Sig#6.cer [OK][OK]
!
! Information about the self-signed certificate
R1# show crypto pki certificates
Router Self-Signed Certificate
  Status: Available
  Certificate Serial Number (hex): 02
  Certificate Usage: General Purpose
```

Issuer:

cn=IOS-Self-Signed-Certificate-681151852

Subject:

Name: IOS-Self-Signed-Certificate-681151852

cn=IOS-Self-Signed-Certificate-681151852

Validity Date:

start date: 00:04:19 UTC Sep 14 2009

end date: 00:00:00 UTC Jan 1 2020

Associated Trustpoints: TP-self-signed-681151852

Storage: nvram:IOS-Self-Sig#6.cer

!

```
R1# show running-config | include crypto|key
crypto pki trustpoint TP-self-signed-681151852
  rsakeypair TP-self-signed-681151852
crypto pki certificate chain TP-self-signed-681151852
  hidekeys
```

Example 3-25 *Controlling HTTP and HTTPS access***! Using the local database for web authentication**

ip http authentication local

!

! Defining allowed source addresses for web access

access-list 1 permit 192.168.1.0 0.0.0.255

ip http access-class 1

Note In strict accord to what was done for SSH, it is a smart choice to set up timing information before beginning the HTTPS configuration. This example employs only the default values. The next section goes beyond the basics for timing setup and complements what has been discussed so far.

Clock Synchronization Using NTP

Ensuring that consistent time information be distributed throughout the network is an important accomplishment. Correct timing not only makes event logs and management data more meaningful, but also brings the possibility of using accounting records for auditing tasks and enabling features (such as *time-based ACLs*) on specific periods.

The clock on a Cisco device might be set up manually using the **clock set exec-level** command. Nonetheless the recommended method for time synchronization is through the use of the NTP.

Because of the potential adverse effects that incorrect time information might have on network availability and manageability, it is advisable that network and security adminis-

trators do whatever they can to guarantee that only authorized synchronization sources are used. Although NTP supports both *plain-text* and *hash-based* (MD5) authentication options, the MD5 method is doubtlessly the preferred one.

Figure 3-8 depicts the reference topology for the analysis of NTP operation using MD5 Authentication. Example 3-26 assembles the configuration commands of an *IOS NTP Server* that was employed to synchronize a set of *NTP clients*. For a better understanding of this scenario, you should pay attention to the following details:

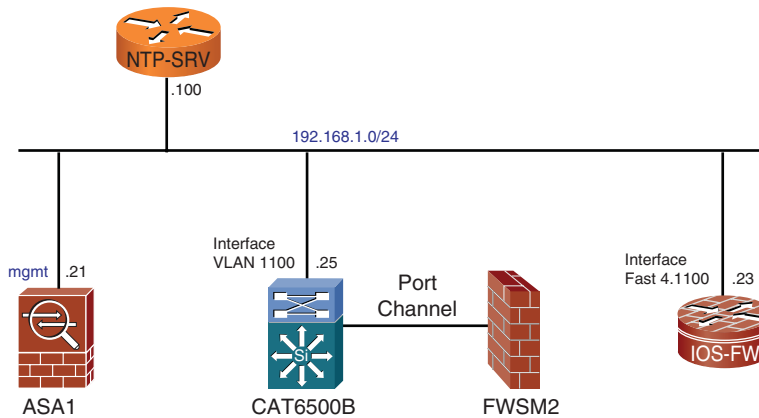


Figure 3-8 Reference Topology for NTP Analysis

- NTP clients authenticate packets received from servers. This is reflected in that the `ntp authenticate` command is configured only on the client side.
- The `ntp trusted-key` command is necessary to activate a key configured with the `ntp authentication-key` command.

Example 3-27 illustrates a classic client-side debug. It also shows how to verify the synchronization status on a certain device and the details associated with the source of clock data. (The commands used in this example are equally valid for ASA and IOS.)

Example 3-26 NTP Baseline Configuration

```
! NTP Server Configuration on IOS
ntp master 4
ntp authentication-key 1 md5 *****
ntp peer 192.168.1.21 key 1
ntp peer 192.168.1.23 key 1
ntp peer 192.168.1.25 key 1
!
! NTP configuration on ASA (server is reachable through logical interface "mgmt")
ASA1# show running-config ntp
```

```

ntp authentication-key 1 md5 *****
ntp authenticate
ntp trusted-key 1
ntp server 192.168.1.100 key 1 source mgmt
!
! NTP Configuration on IOS (including Catalyst 6500)
CAT6500B# show running-config | include ntp
ntp authentication-key 1 md5 *****
ntp authenticate
ntp trusted-key 1
ntp source Vlan1100
ntp server 192.168.1.100 key 1

```

Example 3-27 NTP Operation

```

! A typical debug on the client side (valid for IOS and ASA)
NTP: rcv packet from 192.168.1.100 to 192.168.1.25 on Vlan1100:
 leap 0, mode 4, version 3, stratum 4, ppoll 64
 rtdel 0000 (0.000), rtdsp 0002 (0.031), refid 7F7F0701 (127.127.7.1)
  ref CEAFC40.50C27B03 (11:59:16.315 BRT Thu Nov 19 2009)
  org CEAFC40.7E7F03DF (10:57:20.494 BRT Thu Nov 19 2009)
  rec CEAFCBD2.6C4177F8 (11:59:30.422 BRT Thu Nov 19 2009)
  xmt CEAFCBD2.6C54908E (11:59:30.423 BRT Thu Nov 19 2009)
  inp CEAFC40.7F0D7366 (10:57:20.496 BRT Thu Nov 19 2009)
NTP: syncd to new peer 192.168.1.100
!
! Verifying if the clock is already synchronized (valid for IOS and ASA)
ASA1# show ntp status
Clock is synchronized, stratum 5, reference is 192.168.1.100
nominal freq is 99.9984 Hz, actual freq is 99.9984 Hz, precision is 2**6
reference time is ceafcf15.f87ff739 (11:05:09.970 BRT Thu Nov 19 2009)
clock offset is -31.4958 msec, root delay is 36.30 msec
root dispersion is 1923.63 msec, peer dispersion is 1892.09 msec
!
! Verifying the source of timing information (valid for IOS and ASA)
ASA1# show clock detail
11:05:46.460 BRT Thu Nov 19 2009
Time source is NTP

```

Example 3-28 documents that FWSM receives its timing information from the Catalyst 6500 chassis. If the underlying 6500 is synchronized through NTP, FWSM displays NTP as its time source.

Example 3-28 *Specific Considerations for the FWSM*

```

! Time information obtained from the Catalyst chassis (which uses NTP)
FWSM2# show clock detail
12:42:56.570 BRT Thu Nov 19 2009
Time source is NTP
!
! There is no NTP option on the FWSM CLI
FWSM2# show ntp ?
ERROR: % Unrecognized command
!
FWSM2(config)#ntp ?
ERROR: % Unrecognized command

```

Obtaining an IP Address Through the PPPoE Client

All the IP addresses involved on the baseline configurations considered so far have been assigned statically. This section presents the *PPP over Ethernet (PPPoE)* dynamic method for obtaining an IP address, which is particularly useful on xDSL broadband access environments.

Figure 3-9 portrays one of the typical DSL environments that employ PPPoE to build sessions between client devices and the Broadband Aggregator. The client can be located on the user PC (such as that one behind CPE1) or embedded in the CPE (as exemplified for CPE2).

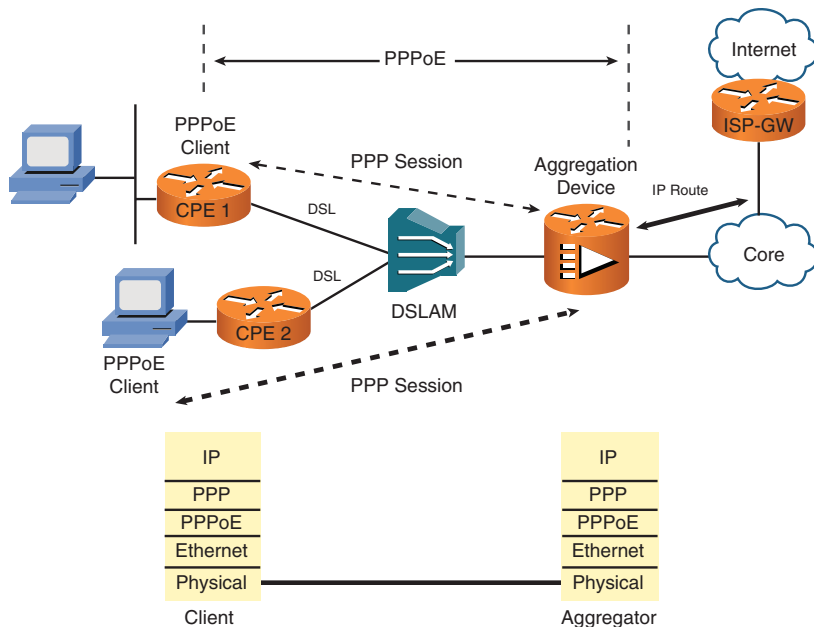


Figure 3-9 *Example of DSL Environment That Employs PPPoE*

Figure 3-10 shows a simplified environment in which the router called *Server* plays the role of the *aggregation device*. The focus of the scenario is on the PPP negotiation (which includes IP address assignment). Example 3-29 relates to this topology and contains the relevant commands for both the server and client sides. The server (emulating the *aggregator*) is an IOS router, and there are two clients: a PIX and another router.

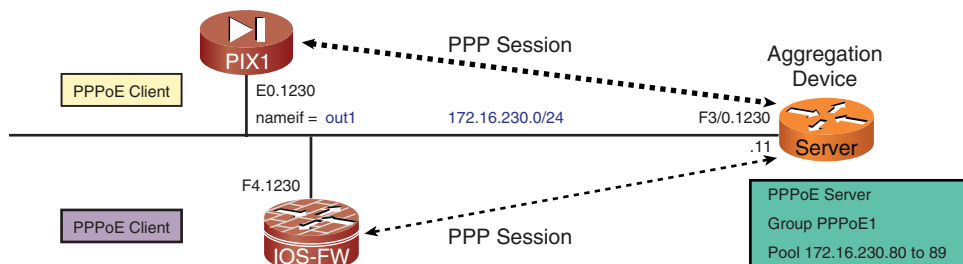


Figure 3-10 Reference Topology for PPPoE Analysis

Example 3-29 Baseline PPPoE Configuration

```
! IOS acting as PPPoE Server (on network 172.16.230.0/24)
username PIX1 password *****
username IOS-FW password *****
!
ip local pool PPPoE1 172.16.230.80 172.16.230.89
!
interface Virtual-Template1
  ip unnumbered FastEthernet3/0.1230
  peer default ip address pool PPPoE1
  ppp authentication chap
!
bba-group pppoe PPPoE1
  virtual-template 1
!
interface FastEthernet3/0.1230
  encapsulation dot1Q 1230
  ip address 172.16.230.11 255.255.255.0
  pppoe enable group PPPoE1
!
! PIX/ASA acting as a PPPoE Client (on interface called out1)
vpdn group PPPoE1 request dialout pppoe
```

```

vpdn group PPPoE1 localname PIX1
vpdn group PPPoE1 ppp authentication chap
vpdn username PIX1 password cisco
!
interface Ethernet0.1230
vlan 1230
nameif out1
security-level 0
pppoe client vpdn group PPPoE1
ip address pppoe setroute
!
! IOS acting as PPPoE client
vpdn enable
!
vpdn-group PPPoE1
request-dialin
protocol pppoe
local name IOS-FW
!
interface Dialer2
ip address negotiated
encapsulation ppp
dialer pool 2
dialer-group 1
no cdp enable
ppp authentication chap
ppp chap password cisco
!
ip route 0.0.0.0 0.0.0.0 Dialer2
!
interface f4.1230
no ip address
pppoe-client dial-pool-number 2
pppoe enable

```

Example 3-30 documents the relevant information concerning PPPoE negotiation on the server side, when the aggregator establishes a session with PIX1 (refer to Figure 3-10). The example also shows PIX1's perspective, characterizing session creation and IP address assignment. You need to observe that the **pppoe setroute** option for the **ip address** command (refer to Example 3-29) results in *PIX1* pointing a default route to the aggregator router (172.16.230.11), which is reachable through interface *out1*.

Example 3-30 *Sample PPP Session Negotiation (Server Side)*

```

ppp1 PPP: Phase is ESTABLISHING, Passive Open
[output suppressed]
ppp1 LCP: MRU 1492 (0x010405D4)
ppp1 LCP: AuthProto CHAP (0x0305C22305)
ppp1 LCP: MagicNumber 0xE80FF183 (0x0506E80FF183)
ppp1 LCP: State is Open
ppp1 PPP: Phase is AUTHENTICATING, by this end
ppp1 CHAP: O CHALLENGE id 1 len 27 from "SERVER"
ppp1 CHAP: I RESPONSE id 1 len 25 from "PIX1"
ppp1 PPP: Phase is FORWARDING, Attempting Forward
ppp1 PPP SSS: Receive SSS-Mgr Connect-Local
ppp1 PPP: Phase is AUTHENTICATING, Unauthenticated User
ppp1 PPP: Phase is FORWARDING, Attempting Forward
ppp1 PPP: Send Message[Connect Local]
ppp1 PPP: Bind to [Virtual-Access1.1]
Vi1.1 PPP: Send Message[Static Bind Response]
Vi1.1 PPP: Phase is AUTHENTICATING, Authenticated User
Vi1.1 CHAP: O SUCCESS id 1 len 4
Vi1.1 PPP: Phase is UP
[output suppressed]
Vi1.1 IPCP: I CONFREQ [ACKrcvd] id 2 len 10
Vi1.1 IPCP: Address 172.16.230.80 (0x0306AC10E650)
Vi1.1 IPCP: O CONFACK [ACKrcvd] id 2 len 10
Vi1.1 IPCP: Address 172.16.230.80 (0x0306AC10E650)
Vi1.1 IPCP: State is Open
Vi1.1 IPCP: Install route to 172.16.230.80#
!
! Session establishment as seen on the client side (PIX/ASA)
%PIX-6-603108: Built PPPoE Tunnel, tunnel_id = 1, remote_peer_ip = 172.16.230.11
ppp_virtual_interface_id = 1, client_dynamic_ip = 172.16.230.80
username = PIX1
!
PIX1# show vpdn pppinterface | exclude MPPE
PPP virtual interface id = 1
PPP authentication protocol is CHAP
Server ip address is 172.16.230.11
Our ip address is 172.16.230.80
Transmitted Pkts: 403, Received Pkts: 404, Error Pkts: 0
!
PIX1# show route out1 | begin Gateway
Gateway of last resort is 172.16.230.11 to network 0.0.0.0
S* 0.0.0.0 0.0.0.0 [1/0] via 172.16.230.11, out1

```

Note The scenario in Figure 3-10 was intentionally simplified so that you can center the discussion on PPP negotiation. In a real-life environment, it would be unlikely for a PPPoE server to have an IP address assigned to the interface in which PPPoE traffic arrives.

Example 3-31 registers the PPPoE sessions from the server (*Aggregator*) standpoint. The server creates /32 routes to each of the clients.

Example 3-32 complements the previous example by documenting an IOS client's point of view of the PPPoE sessions. The *IP Control Protocol (IPCP)* phase is the component of PPP negotiation in charge of IP address assignment (refer to Example 3-30).

Example 3-31 PPPoE Sessions, as Seen on the Server Side

```
! Visualizing PPPoE sessions (server side)
SERVER# show pppoe session
  2 sessions in LOCALLY_TERMINATED (PTA) State
  2 sessions total
```

Uniq ID	PPPoE	RemMAC	Port	Source	VA	State
	SID	LocMAC			VA-st	
2	2	0050.54ff.4c4e	Fa3/0.1230	Vt1	Vi1.1	PTA
		00e0.1e94.7510	VLAN :1230		UP	
3	3	0015.6200.9871	Fa3/0.1230	Vt1	Vi1.2	PTA
		00e0.1e94.7510	VLAN :1230		UP	

```
!
! After IP Address Assignment the Server installs /32 routes to clients
SERVER# show ip route | begin Gateway
Gateway of last resort is not set
  172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks
C    172.16.230.80/32 is directly connected, Virtual-Access1.1
C    172.16.230.81/32 is directly connected, Virtual-Access1.2
C    172.16.250.0/24 is directly connected, FastEthernet3/0.1250
C    172.16.240.0/24 is directly connected, FastEthernet3/0.1240
C    172.16.230.0/24 is directly connected, FastEthernet3/0.1230
```

Example 3-32 PPPoE Sessions, as Seen on the Client Side (IOS-FW)

```
IOS-FW# show pppoe session
  1 client session
```

Uniq ID	PPPoE	RemMAC	Port	Source	VA	State
	SID	LocMAC			VA-st	
N/A	3	00e0.1e94.7510	Fa4.1230	Di2	Vi1	UP
		0015.6200.9871	VLAN :1230		UP	

```
!
! Characterizing that the Dialer2's IP Address was obtained using IPCP
```

```

IOS-FW# show ip interface brief | include Dialer2|Method
Interface                IP-Address      OK? Method Status
Protocol
Dialer2                  172.16.230.81  YES IPCP   up
!
! IP Routes that point to interface Dialer2
IOS-FW# show ip route | include Dialer2|Gateway
Gateway of last resort is 0.0.0.0 to network 0.0.0.0
C       172.16.230.81/32 is directly connected, Dialer2
C       172.16.230.11/32 is directly connected, Dialer2
S*     0.0.0.0/0 is directly connected, Dialer2

```

DHCP Services

Having already studied the static and PPPoE methods of addressing, now look at the services provided by the classic DHCP Protocol. Figure 3-11 portrays a sample topology for the study of DHCP Server and Client functionalities. Example 3-33 shows an IOS router configured as DHCP server while ASA acts as a client (on its *outside* interface). The address assigned to ASA in this case is 172.16.200.41.

Example 3-34 also relates to the topology of Figure 3-11 and teaches how to enable the DHCP server function on ASA. The `dhcpd auto_config` option enables ASA to forward the parameters it receives on a given interface (as client) to another interface where it works as a server. The `show running-config dhcpd` command displays the configuration related to the DHCP daemon on ASA. (Notice that the `auto_config` attributes are shown on the `running-config`.) This example includes the summary information for DHCP services enabled on ASA and the lease information visible on an IOS client.

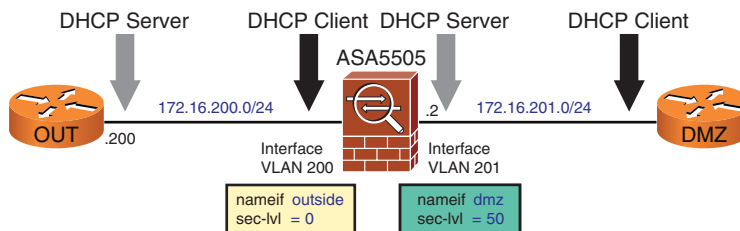


Figure 3-11 Reference Topology for DHCP Server and DHCP Client

Example 3-33 IOS as DHCP Server and ASA as DHCP Client

```

! Router "OUT" acts as DHCP Server for subnet 172.16.200.0/24
interface FastEthernet4.200
 encapsulation dot1Q 200
 ip address 172.16.200.200 255.255.255.0

```

```

!
ip dhcp excluded-address 172.16.200.1 172.16.200.40
ip dhcp excluded-address 172.16.200.50 172.16.200.255
!
ip dhcp pool OUT1
    network 172.16.200.0 255.255.255.0
    default-router 172.16.200.200
    dns-server 172.16.250.250
    domain-name outside.net
!
! ASA configured as a DHCP client on interface outside
ASA5505(config)# interface vlan 200
ASA5505(config-if)# ip address dhcp setroute
%ASA-6-302015: Built outbound UDP connection 46 for outside:255.255.255.255/67
(255.255.255.255/67) to identity:0.0.0.0/68 (0.0.0.0/68)
%ASA-6-604101: DHCP client interface outside: Allocated ip = 172.16.200.41, mask =
255.255.255.0, gw = 172.16.200.200
%ASA-6-302016: Teardown UDP connection 46 for outside:255.255.255.255/67 to
identity:0.0.0.0/68 duration 0:02:03 bytes 1096
!
! The DHCP-learned default route becomes visible on ASA's routing table
ASA5505# show route outside | begin Gateway
Gateway of last resort is 172.16.200.200 to network 0.0.0.0
C    172.16.200.0 255.255.255.0 is directly connected, outside
d*   0.0.0.0 0.0.0.0 [1/0] via 172.16.200.200, outside
!
ASA5505# show interface ip brief | include DHCP|Method
Interface                IP-Address      OK? Method Status
Protocol
Vlan200                  172.16.200.41  YES DHCP    up
!
! Viewing information about the DHCP Server function
OUT# show dhcp server
DHCP server: ANY (255.255.255.255)
Leases:      2
Offers:      1      Requests: 1      Acks : 1      Naks: 0
Declines: 0      Releases: 3      Query: 0      Bad: 0
DNS0:        172.16.250.250,  DNS1:  0.0.0.0
Subnet: 255.255.255.0  DNS Domain: outside.net

```

Example 3-34 ASA as DHCP Server and IOS as DHCP Client

```

! Displaying dhcpd configuration on ASA
ASA5505# show running-config dhcpd

```

```

dhcpcd auto_config outside
**auto_config from interface 'outside'
**auto_config dns 172.16.250.250
**auto_config domain outside.net
!
dhcpcd address 172.16.201.60-172.16.201.69 dmz
dhcpcd enable dmz
!
! Summary information about DHCP Services enabled on ASA
ASA5505# show dhcpcd state
Context    Configured as DHCP Server
Interface mgmt, Not Configured for DHCP
Interface dmz, Configured for DHCP SERVER
Interface outside, Configured for DHCP CLIENT
!
! Displaying information about the DHCP lease on the IOS client
DMZ# show dhcp lease
Temp IP addr: 172.16.201.60 for peer on Interface: FastEthernet4.201
Temp sub net mask: 255.255.255.0
    DHCP Lease server: 172.16.201.2, state: 5 Bound
    DHCP transaction id: 1E88
    Lease: 3600 secs, Renewal: 1800 secs, Rebind: 3150 secs
Temp default-gateway addr: 172.16.201.2
    Next timer fires after: 00:17:52
    Retry count: 0 Client-ID: cisco-0014.f2e3.7df6-Fa4.201
    Client-ID hex dump: 636973636F2D303031342E6632653332E
                        376466362D4661342E323031
    Hostname: DMZ
!
! The default route learned through DHCP is visible on the IOS routing table
DMZ# show ip route | begin Gateway
Gateway of last resort is 172.16.201.2 to network 0.0.0.0
    172.16.0.0/24 is subnetted, 1 subnets
C        172.16.201.0 is directly connected, FastEthernet4.201
S* 0.0.0.0/0 [254/0] via 172.16.201.2

```

Figure 3-12 represents a sample topology used for the investigation of the DHCP Relay feature. When acting as a DHCP Relay, a Layer 3 device (a router or a network firewall, for instance) converts broadcast packets from clients into unicast packets destined to a DHCP server located on a different subnet. The Relay receives replies from the servers and forwards them back to the originating client.

Example 3-35 refers to the internetwork of Figure 3-12, where ASA relays DHCP packets from clients that reside on interface *dmz* (subnet 172.16.201.0/24) to the server 172.16.200.200, reachable through the *outside* interface. It is interesting that there is a


```

! DHCP Relay messages on ASA
DHCPD: Relay msg received, fip=ANY, fport=0 on dmz interface
DHCPD: setting giaddr to 172.16.201.2.
dhcpd_forward_request: request from
0063.6973.636f.2d30.3031.342e.6632.6533.2e37.6466.362d.4661.342e.3230.31 forwarded
to 172.16.200.200.
DHCPD/RA: Punt 172.16.200.200/17152 -> 172.16.201.2/17152 to CP
DHCPD: Relay msg received, fip=ANY, fport=0 on outside interface
DHCPRA: forwarding reply to client
0063.6973.636f.2d30.3031.342e.6632.6533.2e37.6466.362d.4661.342e.3230.31.
DHCPD: Relay msg received, fip=ANY, fport=0 on dmz interface
DHCPD: setting giaddr to 172.16.201.2.
!
! Summary information about DHCP Relay function on ASA
ASA5505# show dhcprelay state
Context Configured as DHCP Relay
Interface mgmt, Not Configured for DHCP
Interface dmz, Configured for DHCP RELAY SERVER
Interface outside, Configured for DHCP RELAY

```

Summary

This chapter described the configuration fundamentals for IOS and ASA-based firewalls, frequently highlighting the similarities between the product families. After being introduced to the main CLI features (such as the help (?) command, completion using the <Tab> key, and the output filters), you learned interface configuration activities and how to enable the remote management protocols (Telnet, SSH, and HTTPS).

The following services were covered:

- Clock synchronization using NTP.
- Dynamic Address Assignment using the PPPoE client on Cisco firewalls, which is well suited for xDSL broadband environments.
- Dynamic Address Assignment using the server, client, and relay features of DHCP.

This chapter examined basic ways to connect firewalls to the network and the essential tasks associated with their management. If more complex ways to insert firewalls in the network topology are required, consider Chapter 5, “Firewalls in the Network Topology.”

Further Reading

Regular Expressions

http://www.cisco.com/en/US/docs/ios/12_2/termserv/configuration/guide/tcfaapre.pdf

This page intentionally left blank

Learn the Tools. Know the Firewall

This chapter covers the following topics:

- Using Access Control Lists beyond packet filtering
- Event logging
- **debug** commands
- Flow accounting and other usages of Netflow
- Performance monitoring using ASDM
- Correlation between graphical interfaces and CLI
- Packet Tracer on ASA
- Packet capture

“I hear and I forget. I see and I remember. I do and I understand.”—Chinese proverb

Chapter 3, “Configuration Fundamentals,” introduced the fundamental concepts and activities involved in the initial setup and management of Cisco network firewalls. This chapter describes how a group of tools, frequently dedicated only to *troubleshooting* purposes, can be consistently employed to create linkages between the theory of operations and the practical behavior of the myriad of firewall features.

Drawing on the axiom *“the more you know it, the better you can use it,”* the knowledge you acquire following this approach can enable you to produce better security designs, which is the underlying goal of this book.

With the concept of *design* as the target, IOS-based tools are useful not only when the router is working as a firewall but also when it is providing the WAN connectivity to a dedicated firewall.

Because of the extensive use of this toolbox throughout subsequent chapters, you should invest the time and energy to become acquainted with the topics discussed. This chapter

is by no means an extensive reference on each topic; the main purposes are to raise awareness and motivate reflection and research.

Using Access Control Lists Beyond Packet Filtering

Access Control Lists (ACL) enable the specification of several parameters that are the base for *packet filtering* activities, regardless if the filtering relies on *stateless* or *stateful* methods.

Although this classic usage of ACLs is the subject of detailed analyses in several chapters of this book, this chapter focuses on alternative manners to take advantage of this flexible tool.

Example 4-1 presents a way to use IOS Access Lists to determine the incidence of fragmented packets within the traffic flows crossing the router. In a similar fashion, Example 4-2 provides a view of the distribution of packets among the 08 values (0 to 7) of the *IP Precedence* parameter. This example also shows how to filter the output of the **show access-list** command to display only the *Access Control Entries* with non-null *bit counts* (as learned in Chapter 3).

Note The examples in this section use the reference topology in Figure 4-2.

Example 4-1 Characterizing Occurrence of Fragmented Packets

```
IOS-FW# show access-list 101
Extended IP access list 101
 10 permit tcp any any fragments (1081 matches)
 20 permit tcp any any (1082 matches)
 30 permit udp any any fragments log-input (360 matches)
 40 permit udp any any (361 matches)
 50 permit icmp any any fragments
 60 permit icmp any any
 70 permit ip any any fragments
 80 permit ip any any
```

Example 4-2 Distribution of Packets According to IP Precedence Value

```
IOS-FW# show access-list 100
Extended IP access list 100
 10 permit ip any any precedence network
 20 permit ip any any precedence internet
 30 permit ip any any precedence critical (47216 matches)
 40 permit ip any any precedence flash-override (787 matches)
 50 permit ip any any precedence flash
 60 permit ip any any precedence immediate
 70 permit ip any any precedence priority
 80 permit ip any any precedence routine
```

```

!
! Displaying only the access control entries (ACL lines) with non-null hit count
IOS-FW# show access-list 100 | include matches
    30 permit ip any any precedence critical (61071 matches)
    40 permit ip any any precedence flash-override (1018 matches)

```

Note IOS ACLs become active only when they are associated with an interface by means of an **ip access-group** command that includes the ACL name (or number) and a direction of an application (**in** or **out**). The **in** parameter specifies that incoming packets (those entering the interface) are matched. On the contrary, the **out** keyword establishes that outgoing packets (those leaving the interface) are matched.

Example 4-3 shows how an IOS ACL can be useful to unveil the distribution of TCP flags within the packet stream. Line 70 represents the case in which only the SYN flag is marked. This relatively large number of hit counts with the SYN bit set indicates a moment of high connection setup rate, be it the result of a normal demand or of a *denial of service* attempt. To determine what is actually going on, other tools might be necessary.

Example 4-3 Incidence of Main TCP Flags Combinations

```

IOS-FW# show access-list TCPFLAGS
Extended IP access list TCPFLAGS
    10 permit tcp any any match-all +ack +psh (2294 matches)
    20 permit tcp any any match-all +ack +syn (530 matches)
    30 permit tcp any any match-all +fin +psh +urg
    40 permit tcp any any match-all +fin log (88 matches)
    50 permit tcp any any match-all +rst
    60 permit tcp any any match-all +urg
    70 permit tcp any any match-all -ack -fin -psh -rst +syn -urg (53494 matches)
    80 permit tcp any any (88 matches)
    90 permit udp any any (882 matches)
   100 permit icmp any any
   110 permit ip any any

```

Note Considering that access-lists become active after being bound to interfaces through the **ip access-group** command, one convenient strategy is to have ACLs such as those just discussed preconfigured on the routers and employ them as temporary resources when the circumstances require them.

Example 4-4 shows the effect of the **log** and **log-input** options for individual *Access Control Entries* (ACE), the second being meaningful for *outbound* ACLs. Although the **log** option simply shows the specific combination of source (IP/port) and destination (IP/port) that produced a match, an ACE configured with the **log-input** parameter generates a message that determines the input interface of the packet (and even the source

MAC address when Ethernet is the L2 medium). The example goes a bit deeper by showing how the MAC knowledge might be useful to determine the L3 next-hop address used to reach the source IP of the packet, which is useful for *traceback* purposes.

Example 4-4 *Determining the Input Interface of a Packet*

```
! Log generated by an Access Control Entry configured with the 'log' option
%SEC-6-IPACCESSLOGP: list TCPFLAGS permitted tcp 172.16.250.98(8498) ->
172.16.252.10(443), 1 packet
!
! Log generated by an Access Control Entry configured with the 'log-input' option
%SEC-6-IPACCESSLOGP: list 101 permitted udp 172.18.200.201(0) (FastEthernet4.1250
000f.3461.9620) -> 172.16.251.10(0), 1 packet
!
! Searching for the MAC Address seen with the 'log-input' option
IOS-FW# show arp | include 000f.3461.9620
Internet 172.16.250.10      243  000f.3461.9620  ARPA   FastEthernet4.1250
!
! Another way of determining the next-hop to the source address logged by the ACL
IOS-FW# show ip cef 172.18.200.201
172.18.0.0/16
    nexthop 172.16.250.10 FastEthernet4.1250
!
! Checking the MAC Address of the next-hop from the IP Address (obtained via CEF
table)
IOS-FW# show ip arp 172.16.250.10
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  172.16.250.10      239       000f.3461.9620  ARPA   FastEthernet4.1250
```

Event Logging

The *System Log Messages* generated by Cisco Firewalls provide the device administrators with rich monitoring and troubleshooting information. This capability of registering both correct and incorrect operating behavior is key to determine and understand circumstances in which errors happen, measuring the level of capacity utilization and for helping on the decision process when undesirable conditions come into play.

As they are generated, System Log Messages can be displayed on the equipment’s console, copied to an internal buffer, or sent to an external server using some sort of message logging protocol, among which Syslog is probably the most well-known and widespread option.

This section examines some Syslog features that can render this classic protocol even more helpful.

Example 4-5 brings a variety of common IOS Syslog messages, ranging from environmental conditions to interface and routing events. Examples 4-6 and 4-7 display classic log messages for the Adaptive Security Appliance and FWSM, respectively.

Example 4-5 *Sample IOS Syslog Messages*

```

! Sample event related to environmental conditions
%ENVMON-3-FAN_FAILED: Fan 2 not rotating
!
! Reload event
%SYS-5-RELOAD: Reload requested by console. Reload Reason: Reload Command.
!
! Sample Interface-related events
%LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback2000, changed state to up
%LINK-5-CHANGED: Interface Loopback2000, changed state to administratively down
%LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback2000, changed state to
down
%CLEAR-5-COUNTERS: Clear counter on all interfaces by console
!
! Sample Routing Events
%OSPF-6-AREACHG: 172.30.30.0/24 changed from area 1 to area 0
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 300: Neighbor 172.16.200.2 (FastEthernet4.200) is
down: Interface Goodbye received

```

Example 4-6 *Sample ASA Syslog Messages*

```

! Event related to Software License Key validity
%ASA-4-444005: Temporary license key 0xcd2759fb 0x899e752f 0x1d3821b3 0xcc479787
0x432eef94 will expire in 260 days.
!
! Sample events related to flow creation and teardown
%ASA-6-302013: Built inbound TCPconnection 4144 for external:72.163.96.81/1278
(72.163.96.81/1278) to mgmt:192.168.1.114/443 (10.97.25.114/443)
%ASA-6-302014: Teardown TCP connection 4144 for external:72.163.96.81/1278 to
mgmt:192.168.1.114/443 duration 0:01:24 bytes 1227990 TCP FINs
%ASA-6-302015: Built outbound UDP connection 4211 for external:171.70.168.183/53
(171.70.168.183/53) to mgmt:192.168.1.185/1032 (10.97.25.185/1032)
%ASA-6-302016: Teardown UDP connection 4211 for external:171.70.168.183/53 to
mgmt:192.168.1.185/1032 duration 0:00:00 bytes 526
%ASA-6-302020: Built outbound ICMP connection for faddr 192.168.2.22/0 gaddr
192.168.1.185/512 laddr 192.168.1.185/512
%ASA-6-302021: Teardown ICMP connection for faddr 192.168.2.22/0 gaddr
192.168.1.185/512 laddr 192.168.1.185/512
!
! Sample packet denial events
%ASA-3-710003: TCP access denied by ACL from 192.168.2.23/48975 to fw-mgmt:
192.168.2.12/23
%ASA-4-106023: Deny udp src mgmt:192.168.1.119/1073 dst vpn1:172.29.67.1/161
by access-group "MGMT"

```



```

!
! Sample events related to Network Address Translation
%ASA-6-305009: Built dynamic translation from dmz:172.16.201.201 to out-
side:172.16.200.12
%ASA-6-305009: Built static translation from dmz:172.16.201.0 to out-
side:172.16.200.0
%ASA-3-305005: No translation group found for icmp src dmz:172.16.201.201 dst out-
side:172.16.200.50 (type 8, code 0)
!
! Sample AAA event (AAA = Authentication/Authorization/Accounting)
%ASA-6-113004: AAA user authentication Successful : server = 172.21.21.250 : user
= user1

```

Example 4-7 Sample FWSM Syslog Messages

```

! Events associated with basic CLI tasks
%FWSM-5-111007: Begin configuration: 127.0.0.51 reading from terminal
%FWSM-5-111008: User 'enable_15' executed the 'configure terminal' command.
%FWSM-5-111005: 127.0.0.51 end configuration: OK
%FWSM-5-111001: Begin configuration: 127.0.0.51 writing to memory
!
! ICMP denial events (packets directed to the Firewall itself)
%FWSM-3-313001: Denied ICMP type=8, code=0 from 172.16.240.10 on interface out1

```

A careful look at the previous examples reveals that ASA and FWSM messages include an identifier number. This information may be used to disable a specific log message, as shown in Example 4-8.

Example 4-8 Disabling a Specific Log Message on ASA and FWSM

```

! Disabling the message numbered 111008
no logging message 111008
!
! Verifying the disabled log messages
ASA2# show logging message all | include disabled
syslog 111008: default-level notifications (disabled)

```

Example 4-9 registers the IOS commands used to direct messages (*traps*) of a given level (0 to 7) to a Syslog server (host). It goes a bit further by showing the supported logging destinations, which follow:

- **Console:** Messages are directed to the serial console.
- **Monitor:** To display messages while using terminal access (telnet and SSH).
- **Buffered:** Internal buffer; Messages display using the **show logging** command.

Example 4-10 is the ASA counterpart of Example 4-9, with identical meanings for the log destinations and levels. The ASA family enables messages to be sent to the Adaptive Security Device Manager (ASDM). When this option is specified, messages are available through the *Real Time Log Viewer*, as shown in Figure 4-1.

Example 4-9 Syslog Destinations and Logging levels on IOS

! Defining the logging level on IOS and directing messages (traps) to a Syslog server

```
logging on
logging source-interface FastEthernet4.1102
logging 192.168.1.114
!
```

! Other Logging destinations on IOS

```
logging console notifications
logging monitor warnings
logging buffered informational
```

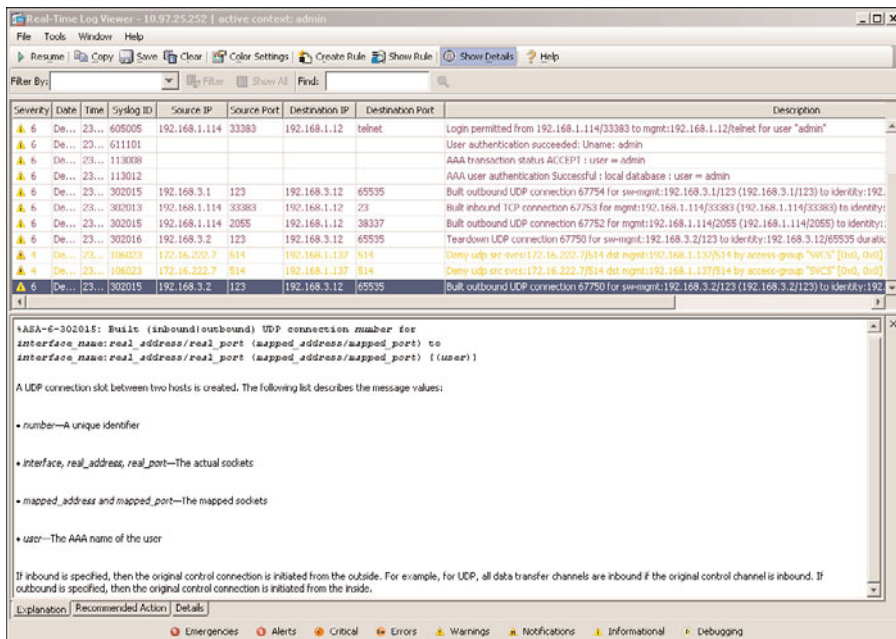


Figure 4-1 ASDM Real Time Log Viewer

Example 4-10 *Syslog Destinations and Logging Levels on ASA and FWSM*

```
! Defining the logging level on ASA and directing messages (traps) to a Syslog
server
logging enable
logging host mgmt 192.168.1.114
logging trap informational
!
! Other Logging destinations and levels on ASA
logging console notifications
logging monitor warnings
logging buffered informational
logging asdm informational
```

Note The console access to the FWSM Module (from the Catalyst 6500 console) is actually a Telnet through the switch backplane. As a result, to control logging levels in this case, you need to use the **monitor** keyword (instead of console) and issue the **terminal monitor** command, whenever a new session is started.

Each ASA Syslog message has a unique number that enables the construction of more complex logging policies. For instance, if only messages up to level 5 (notifications) are to be logged to the physical console, undesirable level 5 (or lower) messages might be moved to a higher level and therefore not sent to this particular destination.

In Example 4-11, the message numbered 111001 (originally at level 5) was moved to the level 7 (debugging).

Example 4-11 *Changing the Level of a Syslog Message on ASA or FWSM*

```
! Changing the default logging level of a Syslog Message
logging message 111001 level debugging
!
! Searching for messages with non-default logging levels
ASA5505# show logging message | include current
syslog 111001: default-level notifications, current-level debugging (enabled)
```

Note The default port for the Syslog server, on both IOS and ASA, is UDP/514.

Debug Commands

In case a *free association* exercise is promoted, the word *debug* would most likely be paired with trouble. **Debug** commands are recognized as a handy resource for troubleshooting purposes. However, they are even more valuable during the deployment, planning, and testing stages, significantly minimizing the probability of problems arising during the actual implementation.

Example 4-12 shows a reduced sample of the numerous **debug** options available on IOS. Each of the first-level options typically unfolds on many other suboptions, which you can preview with the '?' help resource.

Example 4-12 Viewing a Summary of IOS debug Options

```
! Displaying debug options on IOS (always from EXEC mode)
R1# debug ?
    aaa                AAA Authentication, Authorization and Accounting
[ output suppressed ]
    interface          Interface Descriptor Block
    ip                  IP information
    iphc                IPHC information
    ipv6                IPv6 information
[ output suppressed ]
    policy-firewall     Debug policy-firewall
    policy-manager      Policy Manager
    policy-map          Debug policy-map
    ppp                 PPP (Point to Point Protocol) information
[ output suppressed ]
    zone                Zone security events
!
! Viewing some options for the 'debug ip' command
R1# debug ip ?
    access-list         IP access-list operations
    address              IP address activity
    admission            Network admission control debug
    auth-proxy           Authentication proxy debug
```

Following are some important guidelines about **debug** commands:

- Avoid those that produce large outputs on production environments. Commands such as **debug ip packet** and **debug all** should not be used because the amount of data they produce can heavily affect the device operation and cause network disruption.

- Filter the output by knowing the options and suboptions and the associated results. Know what to expect. It is much better to use the **debug** commands to learn the behavior of each feature instead of using them when the problem appears. Be prepared!
- The more specific, the better. It is indispensable to become acquainted with these commands in the lab before using them on live networks.
- To start being displayed, **debug** commands on IOS need the debugging parameter to be specified on the **logging** command (corresponding to 7, the highest logging level). A logging destination must always be specified, as discussed on Examples 4-9 and 4-10.
- ASA's **debug** and **logging** commands are independent of each other, meaning that, by default, ASA does not convey **debug** output in syslog messages. If you need to change this behavior, use the **logging debug-trace** command.
- On IOS, **debug** commands are executed from the EXEC mode, whereas on ASA, they can also be started from **config** mode.

Note Many **debug** commands are used throughout the book to exemplify the operation of features and promote a better understanding of firewall functionality. Investigate the options available (using the Help tool) and register those that seem more promising for the information they provide.

Flow Accounting and Other Usages of Netflow

Within the context of computer networks, a *flow* is defined as a unidirectional sequence of packets between two network endpoints, one of which acts as the *source* and the other as the *destination*. Historically, the seven key fields that have been used to univocally identify a *flow* follow:

- Source IP Address
- Destination IP Address
- Layer 3 Protocol Type
- Source Port Number
- Destination Port Number
- Type of Service (ToS)
- Input Logical Interface

Netflow is a powerful Cisco IOS instrumentation conceived to produce awareness about using network resources. It can capture important instantaneous and statistical data about *flow distribution* on an IOS L3 device, furnishing information that can readily be

applied on the domains of Capacity Planning, Network Application Monitoring, Accounting and Security Analysis, just to name a few.

The flow data profiled through Netflow is accessible using the device CLI and might be exported to *Netflow Collectors*, which can combine the received flows and produce traffic reports that help to identify deviations from what has been defined as the normal behavior for the environment under consideration.

Example 4-13 portrays the classic Netflow record, emphasizing the difference between *key fields* (the set of parameters used to identify packets belonging to the same flow) and *nonkey fields* (additional packet attributes not part of the flow definition). Key fields are defined with a *match* statement, whereas the nonkey fields are specified with a *collect* statement.

Example 4-13 Traditional Netflow Record

```
IOS-FW# show flow record netflow-original
flow record netflow-original:
  Description:      Traditional IPv4 input NetFlow with origin ASs
  No. of users:     0
  Total field space: 53 bytes
  Fields:
    match ipv4 tos
    match ipv4 protocol
    match ipv4 source address
    match ipv4 destination address
    match transport source-port
    match transport destination-port
    match interface input
    match flow sampler
    collect routing source as
    collect routing destination as
    collect routing next-hop address ipv4
    collect ipv4 source mask
    collect ipv4 destination mask
    collect transport tcp flags
    collect interface output
    collect counter bytes
    collect counter packets
    collect timestamp sys-uptime first
    collect timestamp sys-uptime last
```

Note Within the context of traditional Netflow, if two given packets differ in at least one of the seven *key fields*, they are treated as part of different flows.

At this point, it is worth mentioning that many companies have specialized in the analysis of Netflow records and applying the insight it produces to detect the class of attacks known as *Distributed Denial of Service (DDoS)*.

The first part of this section is devoted to IOS Flow accounting, and the second part characterizes how the ASA *Netflow Security Event Logging (NSEL)* feature has proven to be more effective than Syslog on environments that require high-performance logging capabilities.

Enabling Flow Collection on IOS

Figure 4-2 portrays the reference topology used for the ensuing analysis of both traditional and Flexible Netflow.

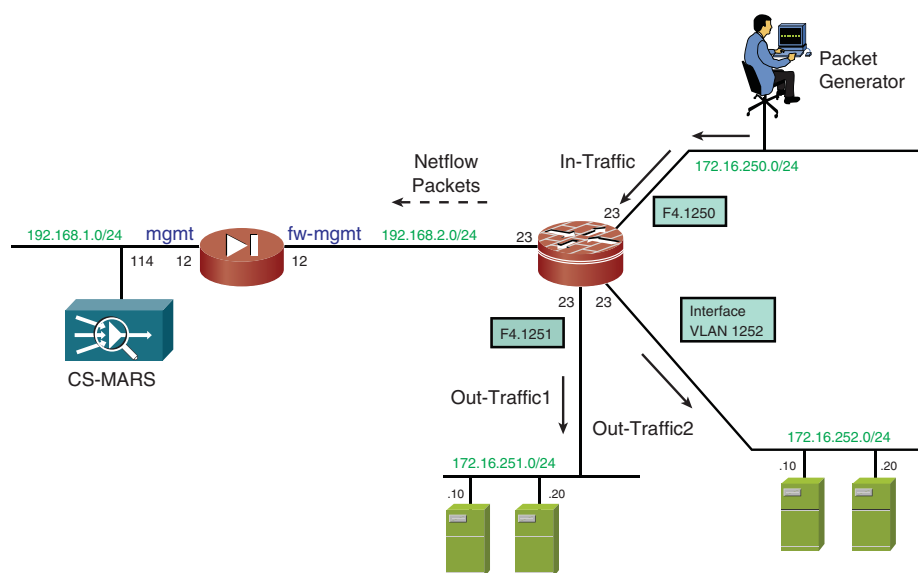


Figure 4-2 Topology for Analysis of Netflow Operation

Traditional Netflow

This section deals with traditional Netflow, which employs a predetermined *record format* and has been available for quite a while on IOS devices.

NetFlow does not involve any connection setup protocol, and it does not require changes to the packets whose flow information is being collected. It is totally transparent to the underlying network (does not capture nor block any packet) and is enabled independently on each router.

Example 4-14 registers the command to enable Netflow for incoming packets (**ip flow ingress**) and shows how to define the destination and source of exported flow data.

Example 4-14 *Enabling Traditional Netflow*

```
! Enabling Netflow for incoming packets
interface FastEthernet4.1250
 encapsulation dot1Q 1250
 ip address 172.16.250.23 255.255.255.0
 ip flow ingress
!
! Enabling Flow Export
ip flow-export source FastEthernet4.1102
ip flow-export destination 192.168.1.114 2055
```

Note The **ip flow egress** command enables the collection of flow data for outgoing packets.

Example 4-15 displays the contents of a sample flow cache. Some relevant information that can be extracted from the output of the **show ip cache flow** command follows:

- Packet size distribution
- Number of active and inactive flows and corresponding timeouts
- Summary of flows on a per L4-port basis
- **Individual active flows:** Although the ToS key-field is not present in the output, the destination interface (DstIf), an important nonkey field, is visible. For instance, although in the example some flows are destined to interface Fa4.1251, many others use Vlan 1252 as their output interface.

Example 4-15 *Displaying the Flow Cache*

```
! Viewing the entire flow cache
IOS-FW# show ip cache flow
IP packet size distribution (18379 total packets):
 1-32   64   96  128  160  192  224  256  288  320  352  384  416  448  480
.000 .000 .000 .106 .000 .000 .000 .000 .080 .000 .000 .699 .000 .000 .113

   512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
.000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 278544 bytes
 406 active, 3690 inactive, 16768 added
```


291733 ager polls, 0 flow alloc failures							
Active flows timeout in 30 minutes							
Inactive flows timeout in 15 seconds							
IP Sub Flow Cache, 34056 bytes							
421 active, 603 inactive, 16783 added, 16783 added to flow							
0 alloc failures, 0 force free							
1 chunk, 1 chunk added							
last clearing of statistics never							
Protocol	Total	Flows	Packets	Bytes	Packets	Active(Sec)	Idle(Sec)
-----	Flows	/Sec	/Flow	/Pkt	/Sec	/Flow	/Flow
TCP-WWW	12554	2.2	1	360	2.2	0.0	15.4
TCP-SMTP	720	0.1	1	260	0.1	0.0	15.5
TCP-Frag	47	0.0	2	260	0.0	12.5	15.5
TCP-other	2054	0.3	1	460	0.3	0.0	14.0
UDP-DNS	958	0.1	1	128	0.1	0.0	15.5
UDP-Frag	34	0.0	1	128	0.0	0.0	15.2
Total:	16367	2.9	1	353	2.9	0.0	15.3
SrcIf	SrcIPAddress	DstIf	DstIPAddress	Pr	SrcP	DstPPkts	
Fa4.1250	172.16.250.52	Fa4.1251	172.16.251.10	11	1740	0035	1
Fa4.1250	172.16.250.58	Fa4.1251	172.16.251.10	11	171E	0035	1
Fa4.1250	172.16.250.54	Fa4.1251	172.16.251.10	11	171A	0035	1
Fa4.1250	172.16.250.50	Fa4.1251	172.16.251.10	11	1716	0035	1
Fa4.1250	172.16.250.52	Fa4.1251	172.16.251.10	11	1718	0035	1
Fa4.1250	172.16.250.65	Fa4.1251	172.16.251.20	06	0000	0000	65
Fa4.1250	172.16.250.67	Fa4.1251	172.16.251.20	06	0000	0000	65
[ouput suppressed]							
Fa4.1250	172.16.250.81	Vl1252	172.16.252.10	06	26D5	01BB	1
Fa4.1250	172.16.250.80	Vl1252	172.16.252.10	06	26D4	01BB	1
Fa4.1250	172.16.250.95	Vl1252	172.16.252.10	06	26CF	01BB	1

Example 4-16 initially shows how to filter the output of the `show ip cache flow` command so that only the flows related to a particular source IP display. It also suggests a way to determine the L3 next hop to reach this source IP using the source interface (SrcIf) information. Compare this traceback method with the one proposed earlier in Example 4-4.

Example 4-16 *Using Netflow Source Interface for Source IP Traceback*

! Filtering the output of the 'show ip cache flow' command							
IOS-FW# show ip cache flow include 172.19.100.100 Src							
SrcIf	SrcIPAddress	DstIf	DstIPAddress	Pr	SrcP	DstP	Pkts
Fa4.1250	172.19.100.100	Vl1252	172.16.252.20	06	2E43	0050	1
Fa4.1250	172.19.100.100	Vl1252	172.16.252.20	06	2E42	0050	1

```

Fa4.1250      172.19.100.100  V11252      172.16.252.20  06 2E41 0050  1
!
! Searching for the source interface in the CEF Table
IOS-FW# show ip cef f4.1250
172.16.250.0/24
    attached to FastEthernet4.1250
172.16.250.10/32
    attached to FastEthernet4.1250
172.19.0.0/16
    nexthop 172.16.250.10 FastEthernet4.1250

```

Example 4-17 focuses on the data associated with the **show ip flow export** command. One noteworthy item in this example is the number of UDP datagrams employed to carry a much higher amount of flows. Figure 4-3 portrays a graphic produced by the Netflow collector subsystem embedded in Cisco Security Monitoring, Analysis and Response System (MARS) that might be used to point out deviations from normal traffic volumes. The graphic clearly shows that something *wrong* (or at least *different*) took place around the 4-to-5 PM interval on Monday.

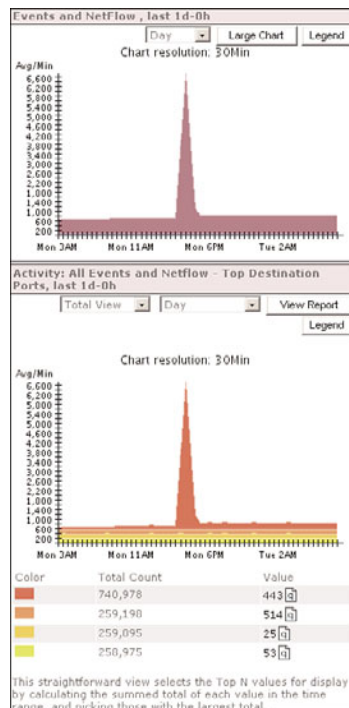


Figure 4-3 Using Netflow Information to Detect Traffic Volume Deviations

Example 4-17 Information About Flow Export

```
IOS-FW# show ip flow export
Flow export v1 is enabled for main cache
Export source and destination details :
VRF ID : Default
Source(1)      192.168.2.23 (FastEthernet4.1102)
Destination(1) 192.168.1.114 (2055)
Version 1 flow records
64131 flows exported in 2677 udp datagrams
0 flows failed due to lack of export packet
0 export packets were sent up to process level
0 export packets were dropped due to no fib
0 export packets were dropped due to adjacency issues
0 export packets were dropped due to fragmentation failures
0 export packets were dropped due to encapsulation fixup failures
```

Figure 4-4 and Figure 4-5 show two detailed graphics built by Cisco MARS (after collecting and processing Netflow information sent by an IOS device). Figure 4-4 shows that the distribution of flows per L4 port during regular hours was quite balanced. Figure 4-5 makes it clear that an increase of approximately 30:1 happened during the peak hour on a specific port. Such a sudden increase of traffic typically signals a DoS or DDoS attack.

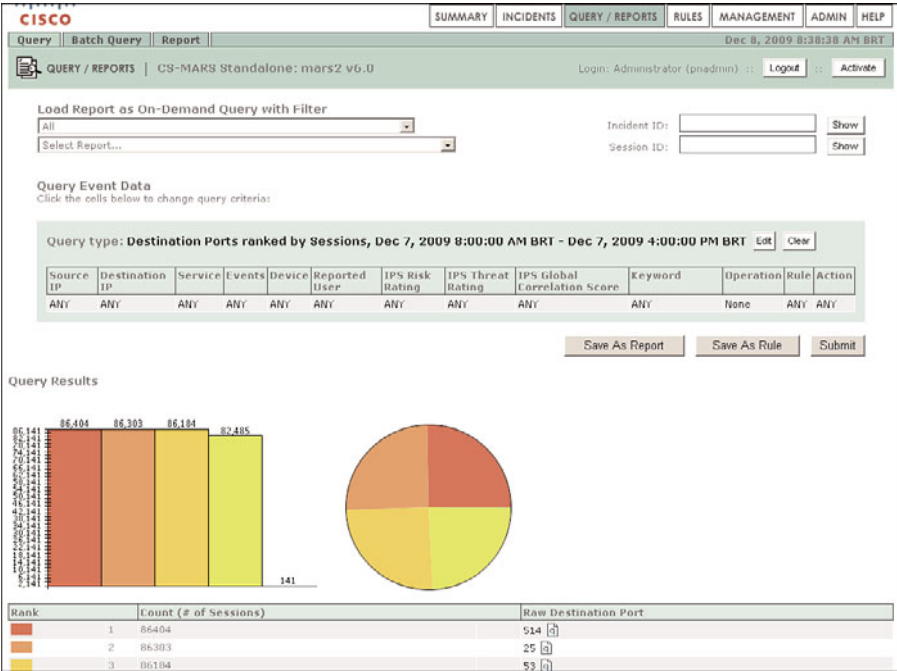


Figure 4-4 Sample Traffic Distribution During Regular Hours

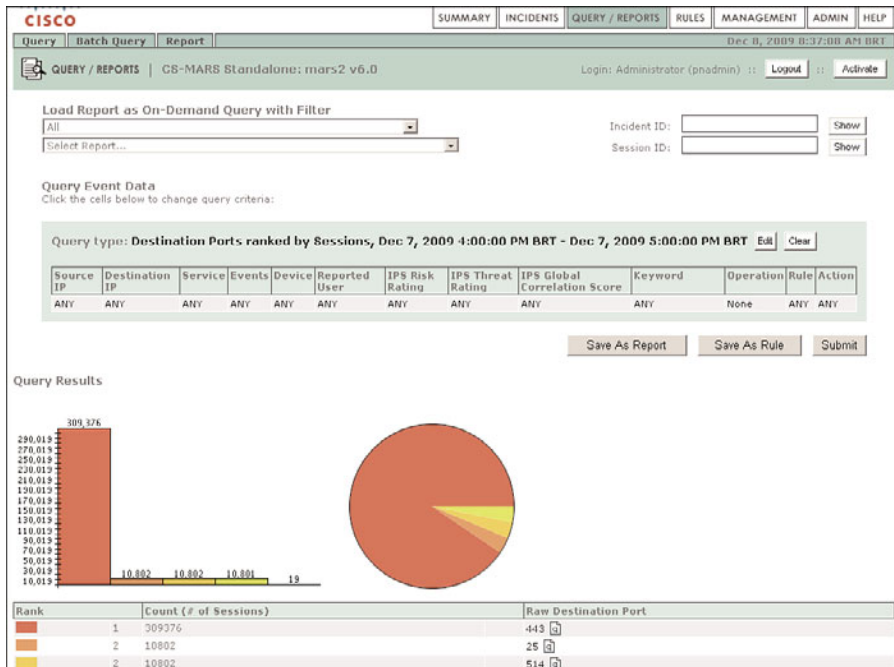


Figure 4-5 Sample Traffic Distribution During the Hour of Sudden Increase of Volume

Note Cisco MARS has a system event that considers the **Sudden Increase of Traffic to Port** an *incident*. (A set of events that after correlation is more meaningful and worth further analysis than any of the component events.)

Netflow v9 and Flexible Netflow

Several versions of Netflow were made available through the years and culminated on v9, which has the distinguishing feature of being *template-based*. Templates bring flexibility to the *flow record* format, therefore enabling future enhancements to NetFlow services without requiring changes to the base Netflow v9 packet structure (Figure 4-6). Some important terms pertaining to the v9 universe deserve a brief definition:

- **Template FlowSet:** A collection of one or more template records that have been grouped in an export packet.
- **Template record:** Used to describe the format of subsequent data records that may be received within export packets. A template record contained in an export packet does not necessarily define the format of data records within that same packet. A Netflow collector application must store any template records received and process the data records it receives according to the appropriate template within its cache.

- **Template ID:** A unique identifier for each template record sent by a given export device. Because uniqueness of the ID is not guaranteed across export devices, the collector should cache the address of the device that produced the template ID.
- **Data FlowSet:** A collection of one or more data records grouped in an export packet.
- **Data record:** Information about an IP flow that exists on the device that originated an export packet. Each data flow set references a predefined template ID.

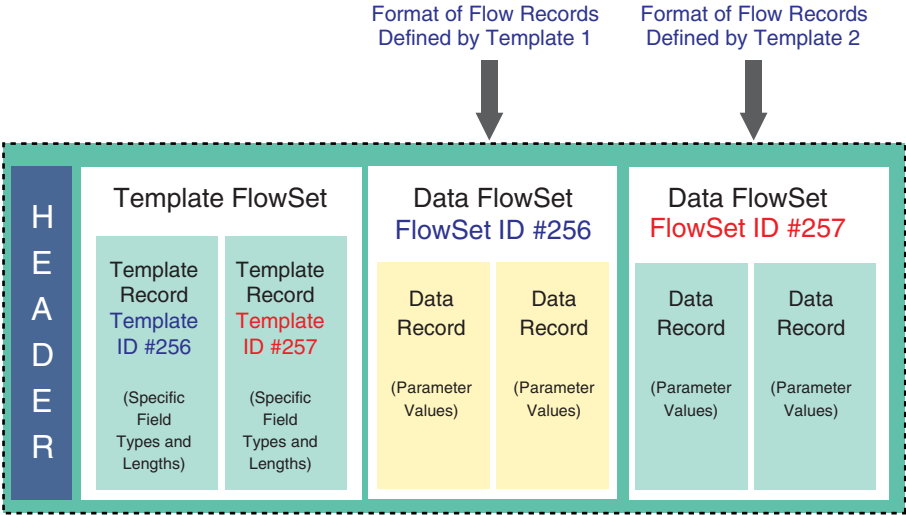


Figure 4-6 *Structure of the Netflow v9 Packet*

Note Reserved flow sets such as Template flow sets and Options flow sets use Template IDs in the range 0–255. Template IDs associated with Data flow sets are numbered from 256 to 65535. For more details about Netflow v9, refer to RFC 3954.

Flexible NetFlow is a more modern IOS utility that totally dissociates the flow collection and export processes. Some of its main features follow:

- Supports the selection of more than 100 fields (IPv4, IPv6, datalink, counters, time-stamp, and so on) as part of the flow record definition, when using the Netflow v9 format.
- Enables the capture of complete packet headers.

- Supports the Stream Control Transmission Protocol (SCTP) as a replacement for UDP, in case reliable transport is necessary.
- Different instances of flow collection can be applied simultaneously to a given interface, providing distinct perspectives on the data gathered (according to the flow record format used in each case). Following the collection phase, the various models of flow sets captured could be sent to different analysis tools (maintained by more than one team, such as Security and Capacity Planning).

Note Although Flexible Netflow was conceived to be used with the Netflow v9 format, for compatibility (and migration) concerns, it might employ the v5 format to export the traditional Netflow records (shown in Example 4-13).

Example 4-18 assembles a typical configuration using Flexible Netflow:

- The **flow exporter** command defines source and destination addresses of the exported packets.
- The **flow record** command defines the key and nonkey fields to be collected. A good exercise is to compare the fields in this example with those that are part of the traditional record (refer to Example 4-13). Extra details about the template record just defined might be seen in the packet captured with a network sniffer (Figure 4-7).

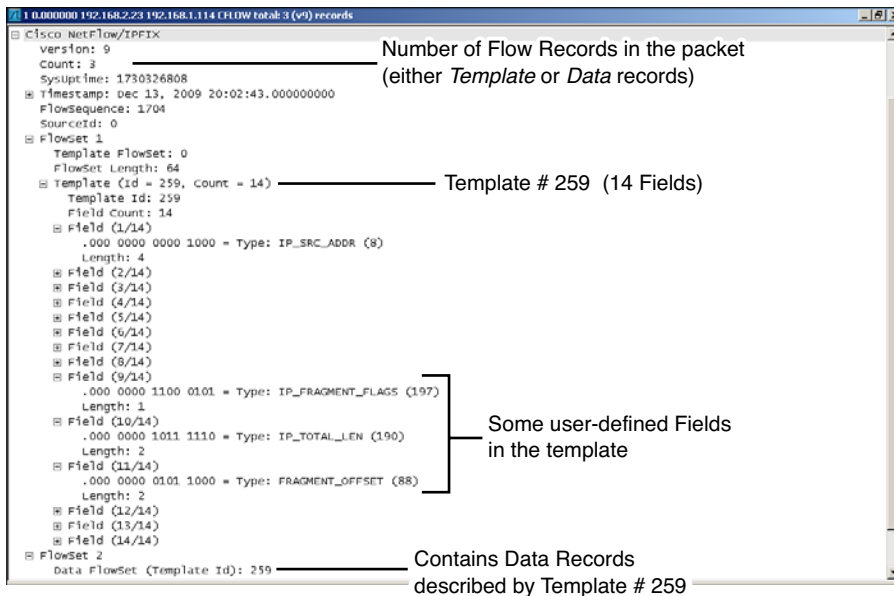


Figure 4-7 Details About the New Flexible-Netflow Template

- The flow monitor combines the flow exporter and flow record definitions and is later associated with an interface, either in the inbound or outbound direction.

This example also registers some useful commands to verify Flexible Netflow configuration and to clear flow statistics.

Example 4-18 Sample Configuration for Flexible Netflow

```

! Defining Flow Export details
flow exporter FLEXNETFLOW1
  description *** Exporting to Cisco MARS
  destination 192.168.1.114
  source FastEthernet4.1102
  transport udp 2055
!
! Defining parameters for a new flow record called FLEXRECORD1
flow record FLEXRECORD1
  match ipv4 precedence
  match ipv4 protocol
  match ipv4 source address
  match ipv4 destination address
  match transport source-port
  match transport destination-port
  match interface input
  collect ipv4 total-length
  collect ipv4 fragmentation flags
  collect ipv4 fragmentation offset
  collect transport tcp flags
  collect interface output
  collect counter bytes
  collect counter packets
!
! Defining the Flow Monitor
flow monitor FLEX1
  record FLEXRECORD1
  exporter FLEXNETFLOW1
!
! Applying the Flow Monitor to the interface (ingress direction)
interface FastEthernet4.1250
  ip flow monitor FLEX1 input
  encapsulation dot1Q 1250
  ip address 172.16.250.23 255.255.255.0
!
! Useful commands to verify Flexible Netflow configuration
IOS-FW#show flow export

```

```

IOS-FW#show flow record FLEXRECORD1
IOS-FW#show flow monitor FLEX1
!
! Clearing Flow Statistics
IOS-FW#clear flow export statistics
IOS-FW#clear flow monitor FLEX1 statistics

```

Figure 4-8 shows an example of a *data flow set* that uses the *template record* of Figure 4-7 (and Example 4-18). This particular data flow set contains five Flow records within a single Netflow v9 packet.

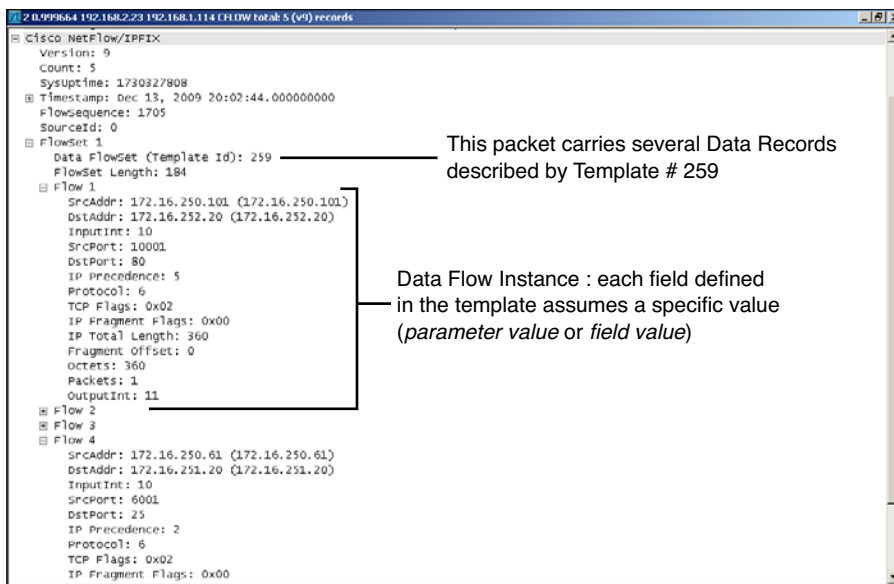


Figure 4-8 Sample Flexible-Netflow Template Applied to Exported Flows

Example 4-19 displays instances of the FLEXRECORD1 defined in Example 4-18. The custom parameters appear highlighted for easier identification.

Example 4-20 builds upon Examples 4-18 and 4-19 by illustrating basic ways to aggregate the collected flow data around the user-defined attributes. At this point revisit the methods discussed in Examples 4-1 through 4-3 to compare with Example 4-20.

Note Similarly to what happens between ACLs and access-groups, the **Flow Monitor** concept becomes active only after association with an interface. As a result, a special **flow monitor** structure (possibly containing custom flow records) can be strategically defined in advance and applied only when the situation requires it.

Example 4-19 *Viewing Flexible-Netflow Records*

```
IOS-FW# show flow monitor FLEX1 cache sort counter packets format record
```

```
Processed 76 flows
```

```
Aggregated to 76 flows
```

```
Showing the top 20 flows
```

```
IPV4 SOURCE ADDRESS:      172.16.250.102
```

```
IPV4 DESTINATION ADDRESS: 172.16.252.20
```

```
TRNS SOURCE PORT:        10182
```

```
TRNS DESTINATION PORT:    80
```

```
INTERFACE INPUT:         Fa4.1250
```

```
IP PRECEDENCE:         5
```

```
IP PROTOCOL:              6
```

```
ipv4 total length:     360
```

```
tcp flags:             0x02
```

```
interface output:         V11252
```

```
counter bytes:            360
```

```
counter packets:          1
```

```
ip fragmentation offset: 0
```

```
ip fragmentation flags: 0x00
```

```
IPV4 SOURCE ADDRESS:      172.16.250.51
```

```
IPV4 DESTINATION ADDRESS: 172.16.251.10
```

```
TRNS SOURCE PORT:         0
```

```
TRNS DESTINATION PORT:    0
```

```
INTERFACE INPUT:         Fa4.1250
```

```
IP PRECEDENCE:         3
```

```
IP PROTOCOL:              17
```

```
ipv4 total length:     128
```

```
tcp flags:             0x00
```

```
interface output:         Fa4.1251
```

```
counter bytes:            128
```

```
counter packets:          1
```

```
ip fragmentation offset: 1480
```

```
ip fragmentation flags: 0x01
```

Example 4-20 *Aggregating Flexible-Netflow Records*

```
! Aggregating around IP Precedence information
```

```
IOS-FW# show flow monitor FLEX1 cache aggregate ipv4 destination address ipv4  
precedence
```

```
Processed 79 flows
```

```
Aggregated to 4 flows
```

```
IPV4 DST ADDR   IP PREC      flows      bytes      pkts
```

```
=====
172.16.252.20      5      32      11520      32
172.16.252.10     4      16      7360      16
172.16.251.20     2      16      4160      16
172.16.251.10     3      15      1920      15
!
```

! Distribution of flows on a per L4-port basis

```
IOS-FW# show flow monitor FLEX1 cache aggregate ipv4 destination address ipv4 pro-
protocol transport destination-port
```

Processed 80 flows

Aggregated to 5 flows

IPV4 DST ADDR	TRNS DST PORT	IP PROT	flows	bytes	pkts
=====	=====	=====	=====	=====	=====
172.16.252.20	80	6	32	11520	32
172.16.252.10	443	6	16	7360	16
172.16.251.20	25	6	16	4160	16
172.16.251.10	53	17	8	1024	8
172.16.251.10	0	17	8	1024	8

!

! Focusing on information about fragmented packets

```
IOS-FW# show flow monitor FLEX1 cache aggregate ipv4 fragmentation flags ipv4 pro-
protocol transport destination-port
```

Processed 75 flows

Aggregated to 5 flows

TRNS DST PORT	IP PROT	IP FRAG FLAGS	flows	bytes	pkts
=====	=====	=====	=====	=====	=====
80	6	0x00	30	10800	30
443	6	0x00	15	6900	15
25	6	0x00	15	3900	15
0	17	0x01	8	1024	8
53	17	0x00	7	896	7

!

! Searching Information about TCP Flags (0x02 = SYN packet)

```
IOS-FW# show flow monitor FLEX1 cache aggregate interface output transport tcp
flags
```

Processed 971 flows

Aggregated to 9 flows

TCP FLAGS	INTF OUTPUT	flows	bytes	pkts
=====	=====	=====	=====	=====
0x12	Fa4.1251	6	1560	6
0x02	Vl1252	926	333560	926
0x18	Vl1252	8	3680	8
0x00	Fa4.1251	15	1920	15
0x18	Fa4.1251	5	1300	5
0x02	Fa4.1251	5	1300	5

0x01	V11252	2	920	2
0x10	V11252	2	920	2
0x12	V11252	2	920	2

Enabling NSEL on an ASA Appliance

The need for a protocol that can perform better than Syslog on environments with high demands of message logging, motivated the adaptation of Netflow v9 to ASA devices. This alternative resource was named *Netflow Security Event Logging* (NSEL) because it focuses on only three types of events, which are naturally the most frequent for firewalls: *flow creation*, *flow teardown*, and *flow denial by ACLs*. Some important facts about the NSEL relations with Netflow and Syslog follow:

- Syslog is text-based and limited to a single event per packet, whereas Netflow is binary and can carry multiple events per packet (refer to Figures 4-7 and 4-8).
- NSEL is **bidirectional**: A connection through a firewall generates two flows within IOS, whereas NSEL treats it as single flow.
- NSEL Records are generated based on the three flow status events (creation, tear-down, and denial) rather than on activity timers. (See Active and Inactive flow counters for traditional Netflow on Example 4-15.)
- When NSEL is enabled, the Syslog messages that describe the three flow status events become redundant and therefore should be disabled to avoid performance impact. These messages, many of which already appeared on Examples 4-6 and 4-7, are further documented in Example 4-21.

Example 4-21 also assembles the basic commands to start sending NSEL packets to the destination host 192.168.1.114 on UDP port 2055, using the logical ASA interface called *mgmt*. (UDP is the only transport protocol initially supported by NSEL.)

Example 4-21 Configuring NSEL Export

```
! Enabling NSEL
policy-map global_policy
class class-default
    flow-export event-type all destination 192.168.1.114
!
flow-export destination mgmt 192.168.1.114 2055
!
! Syslog messages whose information is captured via NSEL
ASA5505# show logging flow-export-syslogs
Syslog ID      Type                Status
302013         Flow Created        Enabled
302015         Flow Created        Enabled
302017         Flow Created        Enabled
```

302020	Flow Created	Enabled
302014	Flow Deleted	Enabled
302016	Flow Deleted	Enabled
302018	Flow Deleted	Enabled
302021	Flow Deleted	Enabled
106015	Flow Denied	Enabled
106023	Flow Denied	Enabled
313001	Flow Denied	Enabled
313008	Flow Denied	Enabled
710003	Flow Denied	Enabled
106100	Flow Created/Denied	Enabled

Figure 4-9 portrays a sample NSEL packet, which clearly builds on the Netflow v9 format. You can find the correspondence between the input and output *interface numbers* sent in the packet and their logical names using the method of Example 4-22.

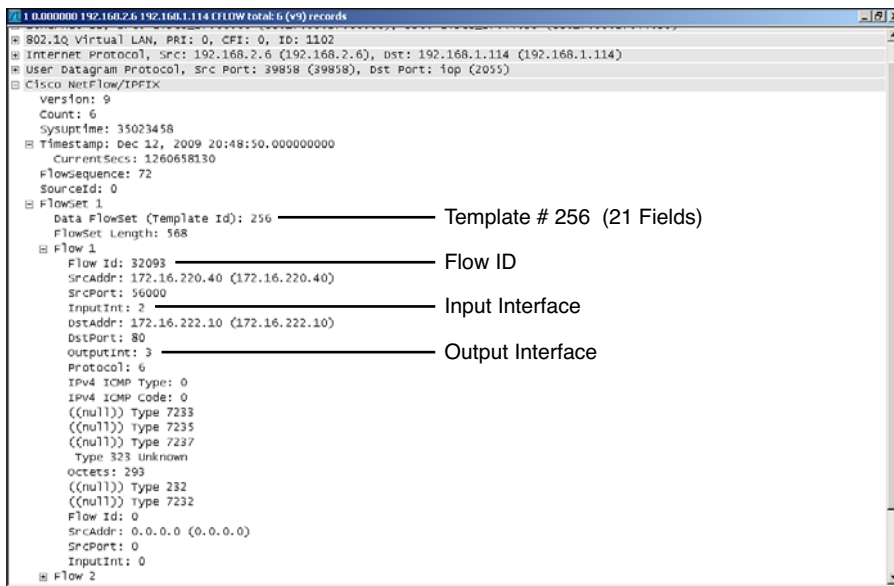


Figure 4-9 Sample NSEL Packet

Figure 4-10 depicts NSEL events as seen on CS-MARS. (MARS receives NSEL packets but displays the individual events.)

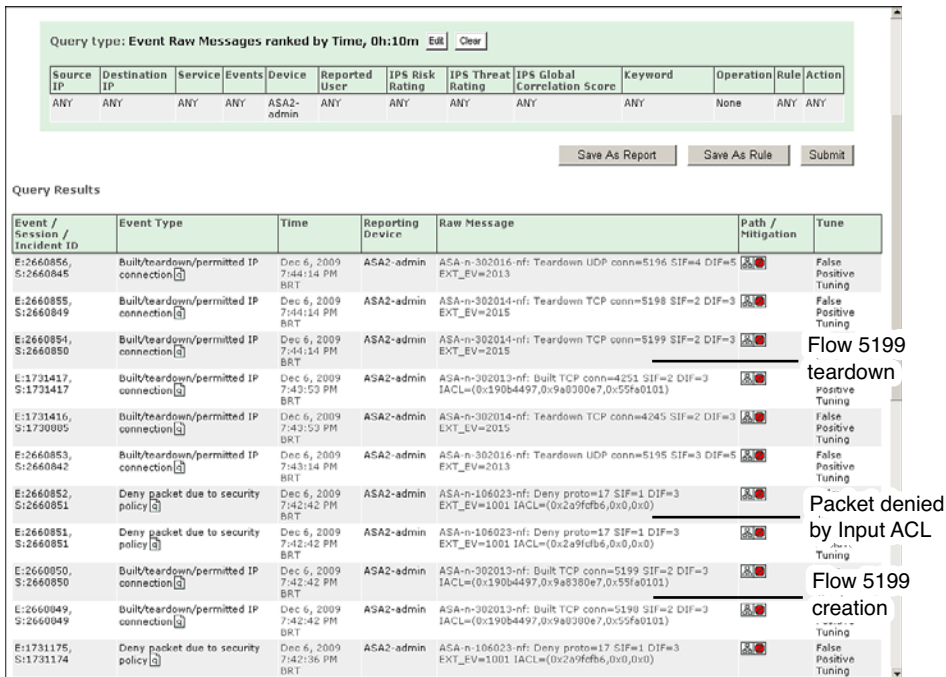


Figure 4-10 Sample NSEL Logging on CS-MARS

Example 4-22 Determining the Number of a Logical Interface

```
ASA2/LAB6# show interface detail | include protocol|number
Interface Management0/0.1102 "mgmt", is up, line protocol is up
Interface number is 1
Interface GigabitEthernet0/1.1220 "out1", is up, line protocol is up
Interface number is 2
Interface GigabitEthernet0/1.1222 "dmz1", is up, line protocol is up
Interface number is 3
```

Note NSEL is supported for IOS-XE on the ASR1000 series routers.

Performance Monitoring Using ASDM

ASDM provides an easy way to build real-time graphs that gather information about interface traffic, connection setup rates, CPU and memory usage, and so on. This is simple to use, therefore deserving to be registered here.

Figure 4-11 brings a collection of 04 sample graphs built simultaneously by ASDM. The paths followed to set up these graphs were

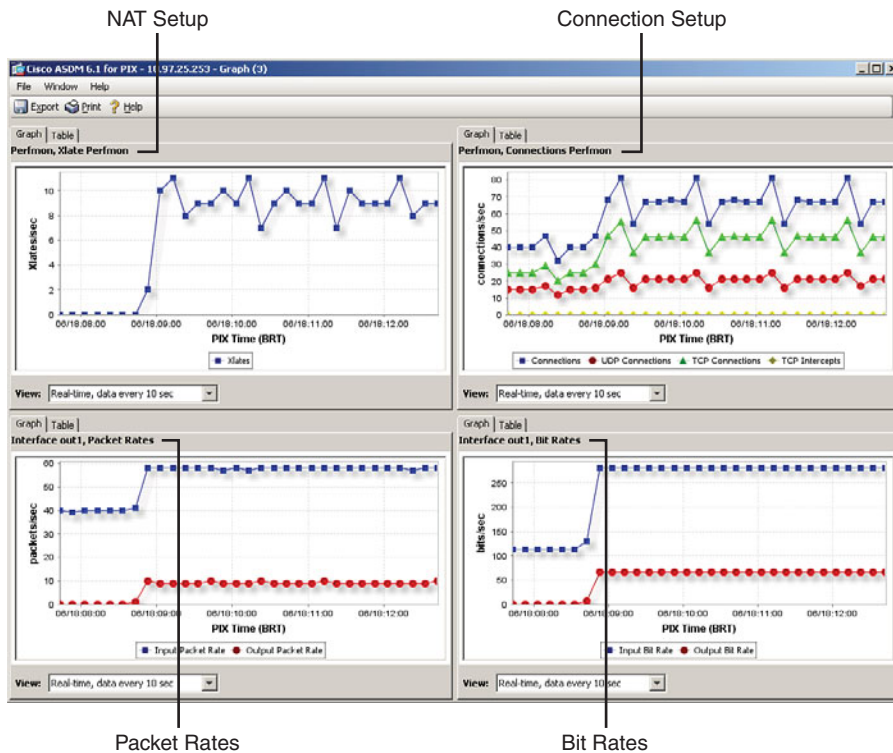


Figure 4-11 Sample ASDM Monitoring Graphics

- **Monitoring > Properties > Connection Graphs > Perfmon:** The Xlate Perfmon and Connections Perfmon options were selected to build the two graphs at the top. It is interesting to observe that the connection setup graph separates the UDP and TCP components that sum up to produce the active connections.
- **Monitoring > Interfaces > Interface Graphs > out1:** The Packet Rates and Bit Rates options were selected to produce a view of traffic volume on the logical interface called out1. These are the two graphs at the bottom.

Many other graphs are available, and the best way to become familiar with them is to start playing with ASDM options.

Correlation Between Graphical Interfaces and CLI

When firewall management is under discussion, it is common to hear administrators express in prose and verse their praise for user-friendly graphical interfaces. Nevertheless, those that need to configure many devices at a time, mainly when there are lots of repetitive

tasks, do show preference for an intelligible CLI and do not demonstrate much patience for dealing with GUIs.

Well, *no technology religion* and no value judgment here. The good news is that Cisco products deliver both options in a tightly integrated manner. Following are some sample benefits that derive from this correlation between the two kinds of interface:

- Users that know the main firewall concepts such as filtering and NAT (even from experience with other vendors' products) might find it easier to initially use the GUI. If you are part of this group, keep in mind that for each task accomplished using the graphical abstraction, the correspondent commands are generated (and may be viewed on the GUI before delivering them to the device). Little reflection leads to the conclusion that this is a simple way to learn the commands from the intuitive GUI.
- You can perform a task from the GUI and then visualize the resultant commands with the goal of creating templates for each procedure. The template might then be employed to generate scripts that fit well for recurring activities. After applying the commands to the devices, the configuration can be imported to populate the GUI, which is a convenient resource for maintenance purposes (assuming that after careful planning and initial deployment, few changes are necessary).
- If a configuration already runs on existent devices, it can be imported with the GUI and the graphical option used thereafter to produce changes.
- If you are familiar with the CLI, you can learn the GUI options after importing the settings from the device.
- Cisco Security Manager, an Enterprise class product that offers a graphical interface for administration of several kinds of Cisco devices, creates the same logical abstraction when similar tasks are performed, irrespectively of the type of equipment managed. This enables firewalls that use different sets of commands (ASA-family versus IOS, for instance) to be managed without knowing the details of each platform.
- The abstraction provided by Cisco Security Manager can be employed to import the configuration running on a certain device type (IOS, for instance) and convert the commands to those used by another family (such as ASA). This is convenient in migration scenarios.

Having presented this discussion about the relation between CLI and GUI, it is time to look more closely by means of some practical examples. Figure 4-12 shows the **Tools** menu on ASDM, which provides access to three resources that will receive special attention:

- **Command Line Interface tool:** Enables the execution of CLI commands from the GUI without the need to connect to the device by other means.
- **Packet Tracer:** A simulation utility that is analyzed later in this chapter.
- **Preferences window:** Permits changing many settings of the ASDM GUI.

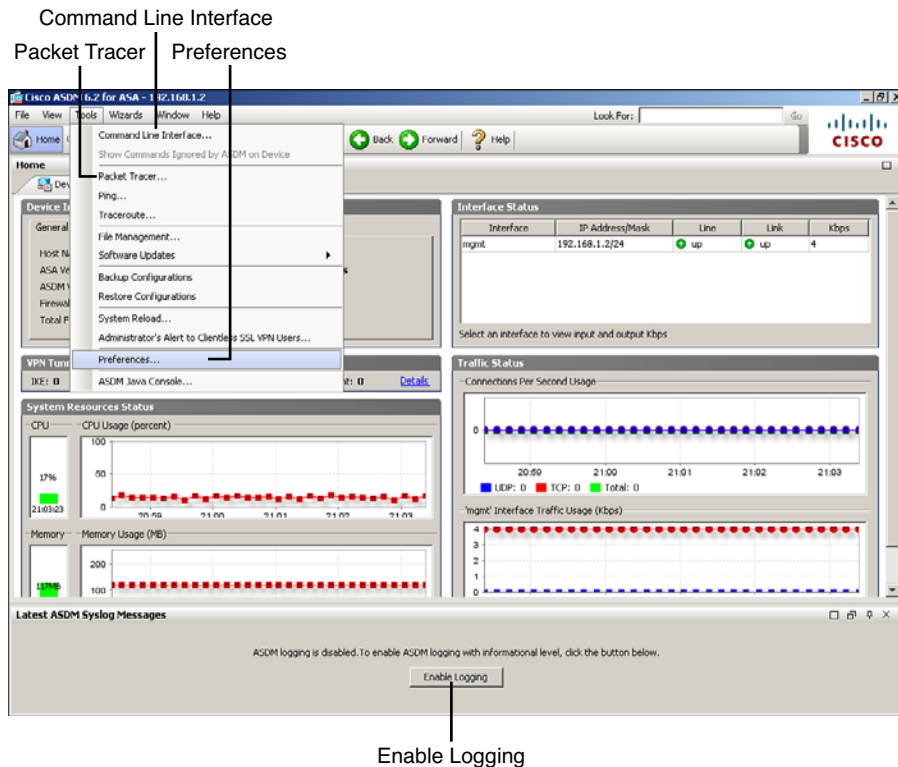


Figure 4-12 ASDM Tools Menu

Figure 4-13 shows the ASDM Preferences window. Among the many options available in the **General** tab, three deserve explicit mention:

- Step 1.** **Warn that configuration in ASDM is out of sync with the configuration on the ASA:** When this box is checked, ASDM detects configuration changes promoted via other means (such as direct CLI access) and generates a warning.
- Step 2.** **Preview commands before sending them to the device:** Checking this box instructs ASDM to display the Preview CLI Commands dialog box before delivering any configuration commands to the device. (For an example refer to Figure 4-15). This is the core action to establish the linkage between GUI and CLI.

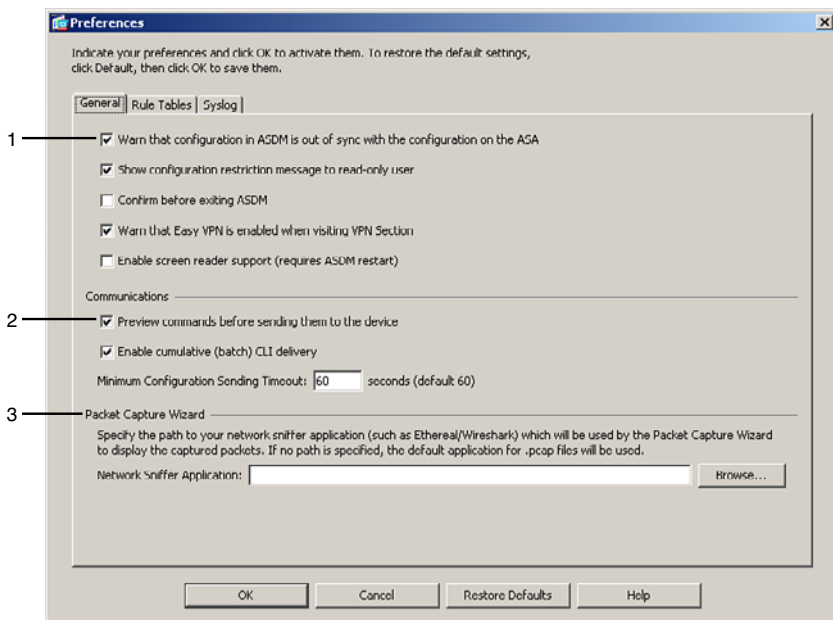


Figure 4-13 *The ASDM Preferences Window*

- Step 3.** **Packet Capture Wizard:** Enables the specification of the Network Sniffer application to be used with the Packet Capture Wizard, accessible from the Wizards menu and analyzed in the last section of this chapter.

Figure 4-14 registers a sample use of ASDM's Command Line Interface tool for the commands `show logging` and `show logging | exclude disabled`. It also points out the sequence of steps to execute any other CLI command.

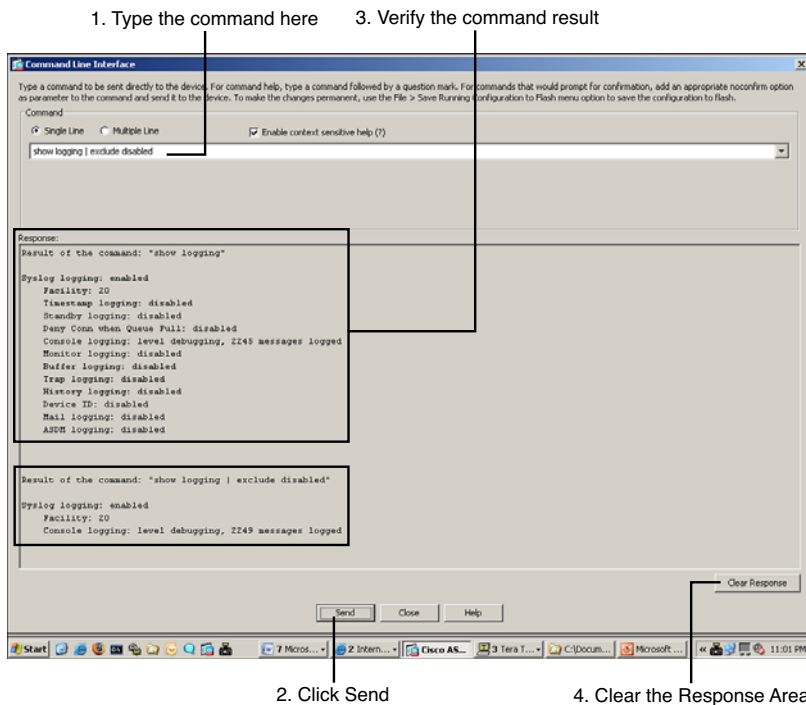


Figure 4-14 Sample Session Using the ASDM Command Line Interface Tool

Packet Tracer on ASA

The *Packet Tracer* application is an invaluable resource to simulate *the life of a packet* as it crosses each software module of the *Adaptive Security Algorithm* on an ASA appliance. Among other usages, it does help to determine whether the running configuration is producing the expected outcome and to identify the cause of eventual packet drops.

As documented earlier on Figure 4-12, the Packet Tracer tool is accessible from the ASDM **Tools** menu. Although Figure 4-15 shows how to start a simulation using the Packet Tracer mechanism, Figures 4-16 and 4-17 respectively show the results of such an activity for an allowed and a dropped packet. Details about the sequence of operations follow:

- Having entered information such as input interface, IP protocol type, Source IP/Port, Destination IP/Port should be provided; the **Start** button was pressed.
- Because the Preview Commands Before Sending Them to the Device box was previously checked on the Preferences window (refer to Figure 4-13), the Preview CLI Commands dialog box appeared. By pressing the **Send** button, the commands are delivered to ASA and the analysis starts.
- Each software module shown on Figures 4-16 and 4-17 can be expanded to learn more details about its operation and how it affects the packets coming through.
- In Figure 4-17, the packet was dropped by the ACCESS-LIST module. The blocking ACL is shown, and a link to locate the specific ACE in the Access Rules Table is provided.

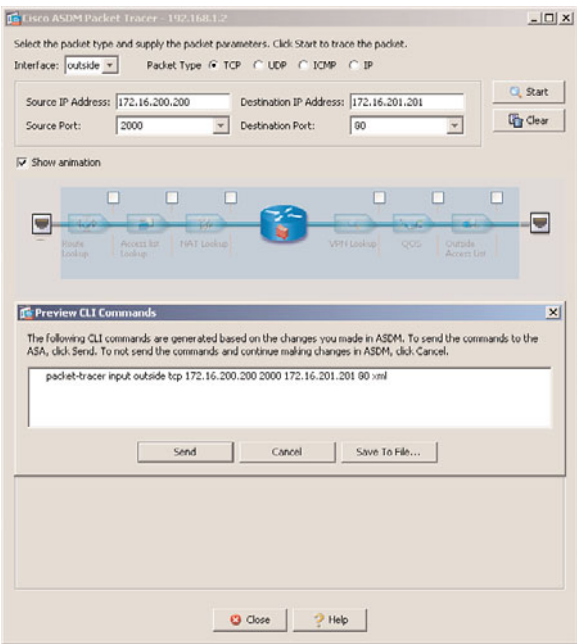


Figure 4-15 Defining a Packet Tracer Task on ASDM

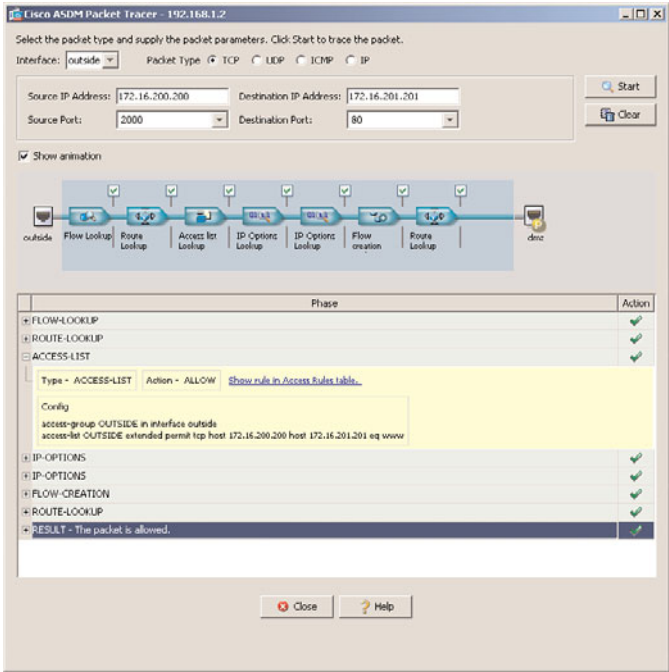


Figure 4-16 Packet Tracer Result for an Allowed Packet

The attractive graphical animations shown in Figures 4-15 through 4-17 are only part of the power inherent to the Packet Tracer feature. It is essentially a CLI resource whose usage is made more intuitive by means of the GUI version. The following examples show the amount of detail provided via the CLI and how useful it can be for documenting the behavior of ASA features.

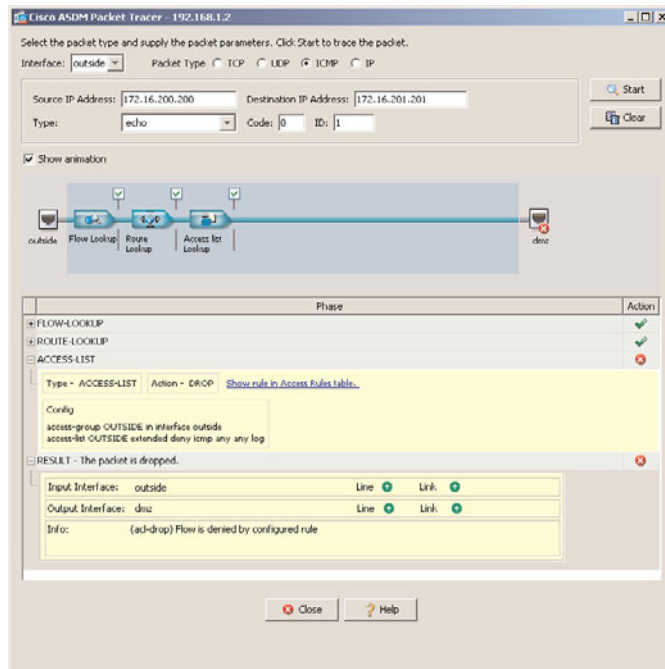


Figure 4-17 Packet Tracer Result for a Denied Packet

Example 4-23 shows the precedence of an existing flow over any other software module. A connection already established was exposed using the **show conn** command, and the pertinent parameters (input interface, Protocol Type, Source IP/Port, and Destination IP/Port) were reused in the **packet tracer** command. The flow was allowed by Phase 1 and the other phases were not even tested.

Example 4-24 shows that connectivity matters! A new flow was created, but because no route to the destination existed, the packet was not allowed to reach subsequent modules.

Example 4-23 Packet Tracer for an Existent Flow

! Viewing existent connections on ASA

```
ASA1# show conn
4 in use, 5 most used
TCP outside 172.16.200.200:23dmz 172.21.21.101:1767, idle 0:00:22, bytes 355,
flags UIO
```

```

! Applying packet tracer for an existent connection (flow)
ASA1# packet-tracer input dmz tcp 172.21.21.101 1767 172.16.200.200 23
Phase: 1
Type: FLOW-LOOKUP
Subtype:
Result: ALLOW
Config:
Additional Information:
Found flow with id 298, using existing flow
Result:
input-interface: dmz
input-status: up
input-line-status: up
Action: allow

```

Example 4-24 Packet Tracer for a Packet That Fails Route Lookup

```

ASA5505# packet-tracer input outside udp 172.16.200.200 6000 172.20.20.20 123
Phase: 1
Type: FLOW-LOOKUP
Subtype:
Result: ALLOW
Config:
Additional Information:
Found no matching flow, creating a new flow
Result:
input-interface: outside
input-status: up
input-line-status: up
Action: drop
Drop-reason: (no-route) No route to host

```

Note This was just a quick visit to the **Packet Tracer** functionality, which hopefully, looked promising. As chapters are developed, many opportunities to use this insightful resource will come into play.

Packet Capture

Packet Capture is a widely known method to make traffic visible to network elements such as RMON probes, IDS sensors, and protocol decoders. Although hubs behave as shared media that automatically replicate traffic to all their ports, methods such as network taps or port mirroring are needed to direct traffic to the analysis equipment on a LAN switched network environment.

Rather than detailing these methods of directing traffic for capture, this section focuses on the embedded capture mechanisms available on Cisco Network Firewalls.

Embedded Packet Capture on an ASA Appliance

The ASDM GUI includes a Packet Capture Wizard, accessible through the Wizards menu (refer to Figure 4-12), greatly facilitating the definition of capture tasks. This wizard is composed of six steps, represented by ASDM windows whose titles are shown in the following:

- Step 1.** **Overview of Packet Capture:** Summary of the six steps.
- Step 2.** **Ingress Traffic Selector:** Specifies details of the type of packets to be captured, including the input (ingress) interface (as shown in Figure 4-18).

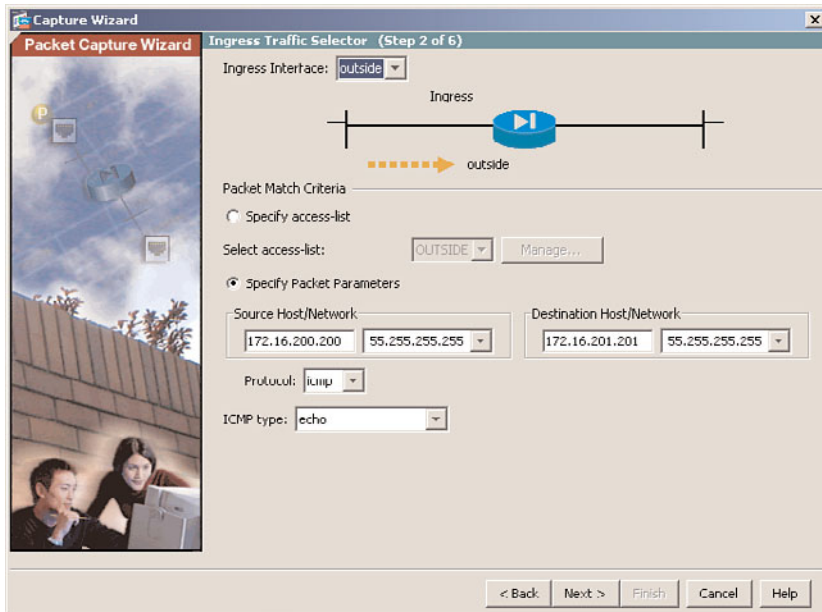


Figure 4-18 *Packet Capture Wizard: Step 2 - Ingress Traffic Selector*

- Step 3.** **Egress Traffic Selector:** Completely analogous to the definitions of Step 2, but focusing on the egress traffic and the output interface.
- Step 4.** **Buffers & Captures:** Parameters such as largest packet size that may be captured and buffer size are configured in this step.
- Step 5.** **Summary:** Previews the resultant **capture** commands that will be delivered to ASA (as registered in Figure 4-19).

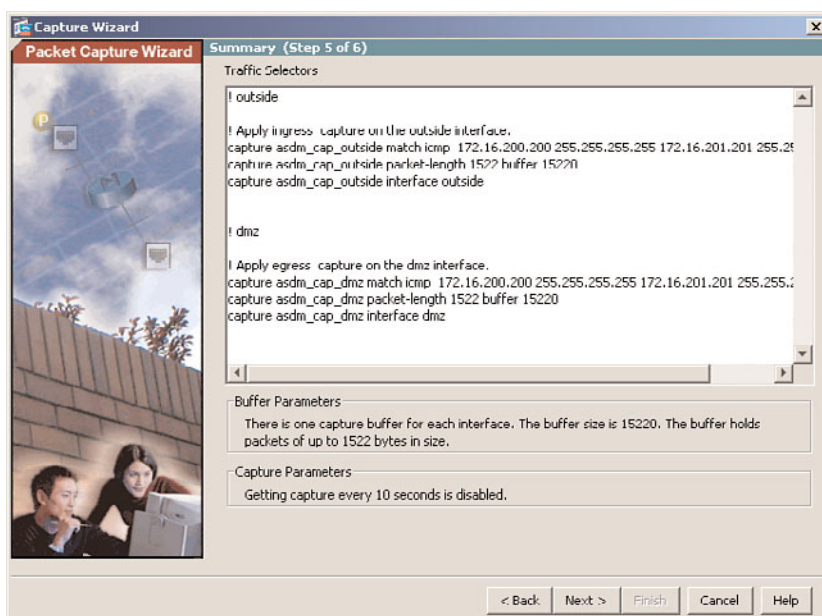


Figure 4-19 Packet Capture Wizard: Step 5 – Summary

Step 6. Run Captures: This step is used to start and stop captures, get the capture buffers from ASA, launch the sniffer application (for ingress or egress traffic), and even save the capture (using *.pcap* or *ascii* formats). This is shown in Figure 4-20.

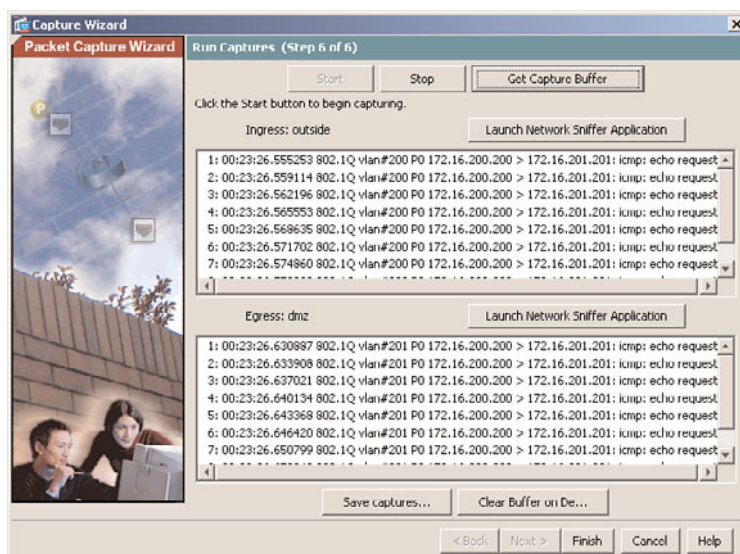


Figure 4-20 Packet Capture Wizard: Step 6 - Run Captures

Note The path to the Network Sniffer Application is specified using the Preferences window, as shown earlier in Figure 4-13. Another interesting option is to save the captures in the .pcap for later analysis with tools such as Wireshark.

Example 4-25 shows that the Packet Capture feature is enabled on the CLI, in accordance with the settings of Figure 4-19. Two ICMP *echo* messages of length 1500 are generated, and the **show capture** command is used to display them.

Example 4-26 demonstrates that a capture task defined including the **trace** keyword brings the possibility of integrating the Packet Tracer and Packet Capture resources.

Example 4-25 *Sample Capture using the ASA CLI*

```

! Capture definition in the CLI
ASA5505# show capture
capture asdm_cap_outside type raw-data buffer 15220 packet-length 1522 interface
outside [Capturing - 0 bytes]
  match icmp host 172.16.200.200 host 172.16.201.201 echo
capture asdm_cap_dmz type raw-data buffer 15220 packet-length 1522 interface dmz
[Capturing - 0 bytes]
  match icmp host 172.16.200.200 host 172.16.201.201 echo
!
! Generating packets to be captured
OUT# ping 172.16.201.201 source 172.16.200.200 size 1500 repeat 2
Success rate is 100 percent (2/2), round-trip min/avg/max = 1/2/4 ms
!
! Capture results on ASA
ASA5505# show capture asdm_cap_outside detail
2 packets captured
  1: 00:32:26.823901 0014.f2e3.8bf8 001e.4a05.2008 0x8100 1518: 802.1Q vlan#200
P0
172.16.200.200 > 172.16.201.201: icmp: echo request (ttl 255, id 34474)
  2: 00:32:26.827792 0014.f2e3.8bf8 001e.4a05.2008 0x8100 1518: 802.1Q vlan#200
P0
172.16.200.200 > 172.16.201.201: icmp: echo request (ttl 255, id 34475)
!
! Clearing the capture buffer between two tests
ASA5505# clear capture asdm_cap_outside

```

Example 4-26 *Integrating Packet Capture and the Packet Tracer*

```

! Including the trace option in the capture definition
ASA5505# show capture
capture asdm_cap_outside type raw-data buffer 15220 packet-length 1522 trace
interface outside [Capturing - 0 bytes]

```



```

!
! Generating Packets for Capture
OUT# ping 172.16.201.201 source 172.16.200.200 size 700 repeat 2
Success rate is 100 percent (2/2), round-trip min/avg/max = 4/4/4 ms
!
! Capture Results
ASA5505# show capture asdm_cap_outside detail
4 packets captured
  1: 00:44:10.859650 0014.f2e3.8bf8 001e.4a05.2008 0x8100 718:
802.1Q vlan#200 P0 172.16.200.200 > 172.16.201.201: icmp: echo request (ttl 255,
id 34481)
  2: 00:44:10.861115 001e.4a05.2008 0014.f2e3.8bf8 0x8100 718:
802.1Q vlan#200 P0 172.16.201.201 > 172.16.200.200: icmp: echo reply (ttl 255, id
34481)
!
! Specifying which captured packet should be traced
ASA5505# show capture asdm_cap_outside packet-number 1 trace
4 packets captured
  1: 00:44:10.859650 802.1Q vlan#200 P0 172.16.200.200 > 172.16.201.201: icmp:
echo request
Phase: 1
Type: FLOW-LOOKUP
Subtype:
Result: ALLOW
Config:
Additional Information:
Found no matching flow, creating a new flow
[output suppressed]
Result:
output-interface: dmz
output-status: up
output-line-status: up
Action: allow
1 packet shown

```

Example 4-27 illustrates another interesting usage of ACLs: They may be employed to specify the packets that are intended to be captured. In the case under analysis, a Telnet session was recorded and later exhibited with the **show capture** command. Diving deeper, one of the packets belonging to the session was shown in detail using the **dump** option.

Example 4-27 *Defining Captures Using ACLs*

```

! Creating an access-list for capturing a Telnet session

```

```

access-list TELNET extended permit tcp host 172.16.200.200 host 172.16.201.201 eq
23
!
ASA5505# show capture
capture asdm_cap_outside type raw-data access-list TELNET buffer 100000 packet-
length 1522 interface outside [Capturing - 4536 bytes]
capture asdm_cap_dmz type raw-data access-list TELNET buffer 100000 packet-length
1522 interface dmz [Capturing - 4536 bytes]
!
ASA5505# show capture asdm_cap_outside
60 packets captured
  1: 01:38:48.846010 802.1Q vlan#200 P0 172.16.200.200.46594 > 172.16.201.201.23:
S 4105900999:4105900999(0) win 4128 <mss 536>
  2: 01:38:48.848710 802.1Q vlan#200 P0 172.16.200.200.46594 > 172.16.201.201.23:
. ack 1394607960 win 4128
  3: 01:38:48.849351 802.1Q vlan#200 P0 172.16.200.200.46594 > 172.16.201.201.23:
P 4105901000:4105901018(18) ack 1394607960 win 4128
!
! Displaying details about a specific packet with the dump option
ASA5505# show capture asdm_cap_outside dump packet-number 2 detail
60 packets captured
  2: 01:38:48.848710 0014.f2e3.8bf8 001e.4a05.2008 0x8100 58: 802.1Q
vlan#200 P0 172.16.200.200.46594 > 172.16.201.201.23: . [tcp sum ok]
4105901000:4105901000(0) ack 1394607960 win 4128 [tos 0xc0] (ttl
255, id 43141)
0x0000 00c8 0800 45c0 0028 a885 0000 ff06 27d7 ....E..(.....'.
0x0010 ac10 c8c8 ac10 c9c9 b602 0017 f4bb 13c8 .....
0x0020 5320 0758 5010 1020 9beb 0000 S.XP.. ....
1 packet shown

```

ASA enables the administrator to capture packets that have been dropped for a particular reason. Example 4-28 displays some of the possible drop causes and registers a situation in which an ACL-related drop has taken place.

Example 4-28 Capturing Packets Dropped by ASA

```

ASA# capture CAP2 type asp-drop ?
acl-drop                               Flow is denied by configured
rule
all                                     All packet drop reasons
async-lock-queue-limit                 Async lock queue limit exceeded
bad-crypto                             Bad crypto return in packet
bad-ipsec-natt                         Bad IPSEC NATT packet
bad-ipsec-prot                         IPSEC not AH or ESP
bad-ipsec-udp                          Bad IPSEC UDP packet
bad-tcp-cksum                          Bad TCP checksum

```

bad-tcp-flags	Bad TCP flags
[output suppressed]	
unsupported-ip-version	Unsupported IP version
vpn-handle-error	VPN Handle Error
vpn-handle-mismatch	VPN Handle Mismatch Error
wccp-redirect-no-route	WCCP redirect packet no route
wccp-return-no-route	WCCP returned packet no route
<cr>	
!	
! Capturing packets dropped by an ACL	
ASA# show capture	
capture CAP2 type asp-drop acl-drop [Capturing - 312 bytes]	
!	
ASA# show capture CAP2	
4 packets captured	
1: 15:09:55.755576 802.1Q vlan#200 P0 172.16.16.200.16386 >	
10.10.10.140.23: S 1501932073:1501932073(0) win 4128 <mss 536> Drop-	
reason: (acl-drop)	Flow is denied by configured rule

Embedded Packet Capture on IOS

The Embedded Packet Capture functionality was introduced in IOS on late 12.4T releases. Some important terminology related with this feature is summarized in the following:

- **Capture Buffer:** An area in memory for storing the packet data. Buffer type, size and name may be defined with the **monitor capture buffer** command. An access-list is used to specify the types of packets that should be captured.
- **Capture Point:** Defines, by means of the **monitor capture point** command, the transit point where packets should be captured and associated with a buffer. The transit points can be CEF (Cisco Express Forwarding) switching path or process switching, for either IPv4 or IPv6.

The **monitor capture point associate** command ties a capture point to a capture buffer.

Example 4-29 assembles the commands to configure a **Capture Point** called *ICMP* that is bound to a circular buffer called *CAPTURE1*. The filter access-list 100 limits the capture to ICMP messages with origin on the 192.168.1.0/24 network and destined to host 192.168.1.200 (which is the IP address of the router interface). The example also teaches how to verify the configuration.

In Example 4-30, an ICMP packet of 160 bytes is generated and captured. Details about the packet may be visualized using the **dump** option with the **show monitor capture buffer** command. Packets might optionally be exported using protocols such as FTP and HTTPS.

Example 4-29 *Defining an IOS Packet Capture and Verifying the Configuration*

```

! Defining interesting traffic for Capture
access-list 100 permit icmp 192.168.1.0 0.0.0.255 host 192.168.1.200
!
! Defining a capture point named ICMP
OUT# monitor capture point ip process-switched ICMP in
%BUFCAP-6-CREATE: Capture Point ICMP created.
!
! Defining a capture buffer called CAPTURE1
OUT# monitor capture buffer CAPTURE1 size 512 max-size 1024 circular
!
! Associating the filter access-list with the capture buffer just created
OUT#monitor capture buffer CAPTURE1 filter access-list 100
Filter Association succeeded
!
! Associating the capture point with the capture buffer
OUT# monitor capture point associate ICMP CAPTURE1
!
! Starting Capture
OUT# monitor capture point start ICMP
%BUFCAP-6-ENABLE: Capture Point ICMP enabled.
!
! Viewing information about the Capture Buffer and the associated Capture Point
OUT# show monitor capture buffer CAPTURE1 parameters
Capture buffer CAPTURE1 (circular buffer)
Buffer Size : 524288 bytes, Max Element Size : 1024 bytes, Packets : 1
Allow-nth-pak : 0, Duration : 0 (seconds), Max packets : 0, pps : 0
Associated Capture Points:
Name : ICMP, Status : Active
Configuration:
monitor capture buffer CAPTURE1 size 512 max-size 1024 circular
monitor capture point associate ICMP CAPTURE1
monitor capture buffer CAPTURE1 filter access-list 100
!
OUT#show monitor capture point ICMP
Status Information for Capture Point ICMP
IPv4 Process
Switch Path: IPv4 Process          , Capture Buffer: CAPTURE1
Status : Active
Configuration:
monitor capture point ip process-switched ICMP in

```

Example 4-30 *Capturing the Packets*

```

! Generating packets to be captured
R1# ping 192.168.1.200 size 160 repeat 1 data EEA
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
!
! Viewing details about a captured packet in IOS
OUT# show monitor capture buffer CAPTURE1 dump
12:37:23.835 UTC Sep 13 2009 : IPv4 Process      : Fa4.100 None
840304B0:          0014F2E3 8BF80014 F2E37DF6      ..rc.x..rc}v
840304C0: 08004500 00A00007 0000FE01 3774C0A8      ..E.. ...~.7t@
840304D0: 01C9C0A8 01C80800 E4A30003 00000000      .I@(.H..d#. ....
840304E0: 0000004B 45A8EEAA EEAEEAA EEAEEAA      ...KE(n*n*n*n*n*
840304F0: EEAEEAA EEAEEAA EEAEEAA EEAEEAA      n*n*n*n*n*n*n*n*
84030500: EEAEEAA EEAEEAA EEAEEAA EEAEEAA      n*n*n*n*n*n*n*n*
84030510: EEAEEAA EEAEEAA EEAEEAA EEAEEAA      n*n*n*n*n*n*n*n*
84030520: EEAEEAA EEAEEAA EEAEEAA EEAEEAA      n*n*n*n*n*n*n*n*
84030530: EEAEEAA EEAEEAA EEAEEAA EEAEEAA      n*n*n*n*n*n*n*n*
84030540: EEAEEAA EEAEEAA EEAEEAA EEAEEAA      n*n*n*n*n*n*n*n*
84030550: EEAEEAA EEAEEAA EEAEEAA EEAEEAA      n*n*n*n*n*n*n*n*
84030560: EEAAB                                n*+

```

Summary

This chapter analyzed the usage of tools that have been associated with troubleshooting, with the alternative perspective of understanding firewall operations. A wise employment of such a toolbox can effectively contribute to better designs and implementations, minimizing the likelihood of reaching the troubleshooting stage.

- You can use ACLs for more than the classic packet-filtering tasks in activities that include measuring packet distribution based on parameters such as IP Precedence, presence of IP fragments, occurrence of a certain TCP Flag combination, just to name a few. They are also useful for traceback purposes and for limiting both the amount of captured packets and the output of **debug** commands.
- Logging provides helpful information about several aspects of device operation. The traditional protocol for event logging is Syslog.

- Several **debug** commands are available on Cisco devices and provide even more detailed information than Syslog. Nevertheless they should be used carefully to avoid disruption on the network environment.
- Netflow and its derivatives are a powerful instrumentation that you can apply to tasks such as flow accounting, traceback, and traffic anomaly detection.
- Netflow Security Event Logging (NSEL), an ASA feature based on Netflow v9, was designed to perform better than Syslog on environments with high logging rates. The use of NSEL makes many Syslog messages redundant.
- The graphical interfaces for Cisco Firewall management are always tightly coupled with the CLI commands that actually materialize the features.
- Packet Tracer is a useful simulation mechanism available on ASA appliances.
- ASA and IOS devices support Embedded Packet Capture functionality, which provides quick access to captures without the need to set up a packet mirroring infrastructure.

This page intentionally left blank

Firewalls in the Network Topology

This chapter covers the following topics:

- Introduction to IP routing and forwarding
- Static routing overview
- Basic concepts of routing protocols
- RIP overview
- EIGRP overview
- OSPF overview
- Configuring authentication for routing protocols
- Bridged operation

“Two roads diverged in a wood, and I took the one less travelled by / And that has made all the difference.” —Robert Frost

In Chapter 3, “Configuration Fundamentals,” you learned how to assign IP addresses to firewall interfaces using either static or dynamic means (Dynamic Host Configuration Protocol [DHCP] or PPP over Ethernet [PPPOE]). That chapter also discussed the basic features of the Command Line Interface (CLI) of each family of Cisco network firewalls. Chapter 4, “Learn the Tools. Know the Firewall,” presented a set of tools that greatly helps you understand the theory of operations of Cisco firewalls.

In this chapter, you turn your attention to the methods of integrating the firewalls into the network topology using a Layer 3 (L3) or Layer 2 (L2) connectivity model. Although, historically, the L3 method has been the most prevalent choice, the L2 option (*Bridge mode*) enables the insertion of firewall devices with a minimal need of topology reconfiguration—an attractive characteristic in some situations.

Your study begins with a brief discussion of IP routing and forwarding mechanisms. Some practical routing scenarios that combine Adaptive Security Appliances (ASA) and

IOS Firewalls are analyzed and, finally, the operation of *Bridge mode* (also known as *Transparent mode*) is examined.

Introduction to IP Routing and Forwarding

IP routing is concerned with the choice of a path over which the packets (IP datagrams), destined to a given host, are sent. Although there might exist some advanced techniques that use additional attributes (the source address, for instance), the classic definition of routing considers the destination address as the only criterion for path selection. With that said, keep in mind that any reference made to IP routing always means *destination-based* routing.

The IP routing function can be divided in four basic activities:

- Step 1. Gathering routing information:** You can do this through the manual definition of static routes or using Dynamic Routing Protocols.
- Step 2. Installing entries in the routing table:** Before installing a path in the routing table, a Cisco Firewall performs two comparisons in the following order: in the event that two equal length network prefixes are available to a given destination, a Cisco Firewall prefers the one with the lowest Administrative Distance. For two equal length prefixes that have the same value for the Administrative Distance parameter, a Cisco Firewall chooses the one that has the lowest cost under the perspective of the particular routing protocol.
- Step 3. Searching for the longest prefix match:** When a packet arrives at the incoming interface, its destination IP address (destIP) is extracted and compared with each entry available in the routing table. The comparison that results in the longest bitwise match for the network mask will be selected. (The last possibility of finding a match is to use a default route, if one is available).
- Step 4. Forwarding the packet on the outgoing interface:** When a match happens in Step 3, it points to an entry in the routing table that has an associated outgoing interface. This last step includes the construction of the appropriate L2 header for this interface.

Note The simplest routing case happens when the *incoming* and *outgoing* interfaces are directly connected to the same firewall (or routing device). In such a situation Steps 1 and 2 are not necessary.

Note When two routes to a given destination point to the same outgoing interface and have equal values for the Administrative Distance and cost parameters, they are both installed in the routing table. In such a situation, load sharing takes place.

Figure 5-1 depicts two examples of generic IP routing tables for Routers R1 and R5. The key terms in the figure are *connected* (networks to which the particular router is directly attached) and *remote* (destinations inserted either manually or dynamically in the routing table and reachable with the help of other routers).

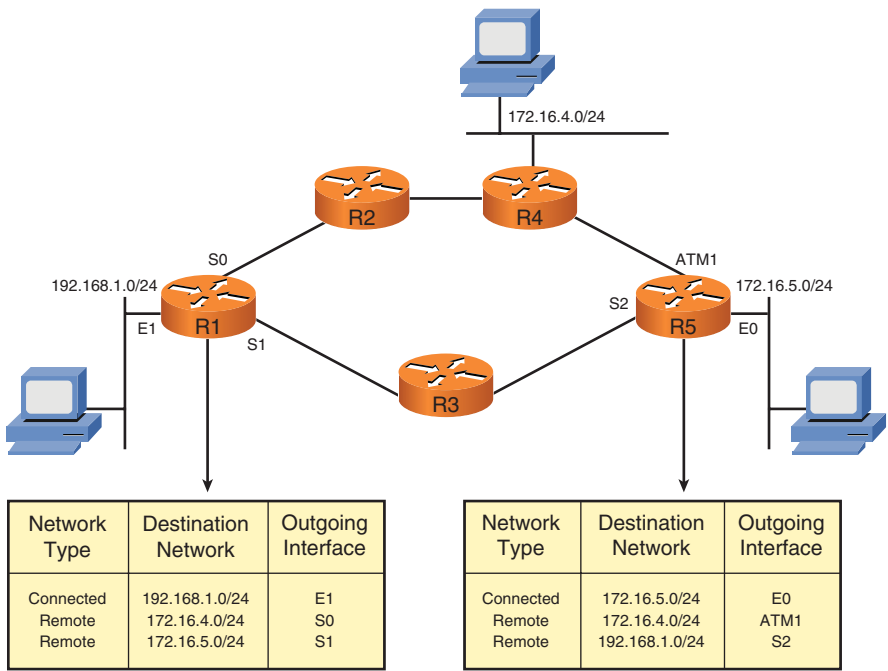


Figure 5-1 Example of Generic IP Routing Tables

Static Routing Overview

If the destination IP address resides in a network that is not directly connected, the firewall needs the contribution of other routing devices to deliver the packet.

When you need to send packets to a remotely connected destination, the first available option is to manually define *static routes*. A typical *static entry* in the routing table is the combination of a *network prefix* and a pointer to the *next-hop router* toward the desired destination.

Static routes are simple to configure and have the advantage of being available in any version of Cisco Firewalls. But when choosing to route packets in this fashion, you need to be aware of some important characteristics:

- Static routing fits well for simple topologies with a small number of destinations. As the number of routing devices and destinations increases, it can be challenging to keep the reachability information up to date.

- When a set of routers is configured to use a Dynamic Routing Protocol, each router automatically shares its view of reachability for the participating destinations. In this way, if a router detects, for instance, that one of its connected networks is unreachable, it automatically propagates this information to the other elements of the set so that they can update their individual perspective of the topology. Static routing, on the other hand, relies on the administrator's view of the topology and has no automatic way to distribute this updated information.

Figure 5-2 displays the topology used for the examples covering the analysis of static routing. Example 5-1 shows the relevant configurations to achieve full connectivity for the networks depicted in Figure 5-2.

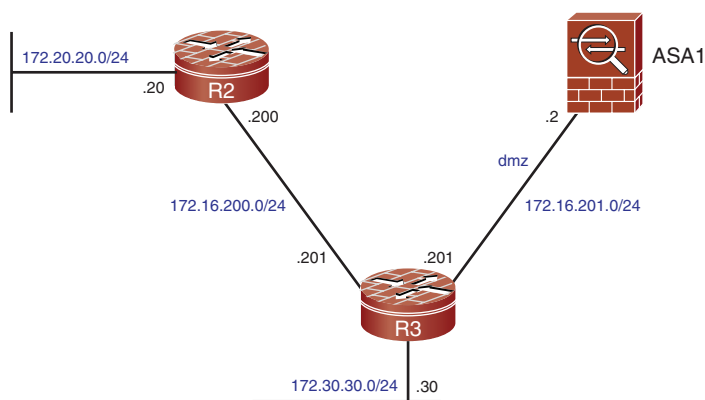


Figure 5-2 *Reference Topology for Static Routing Analysis*

Example 5-1 *Baseline Connectivity Configurations for Static Routing Analysis*

```
! Static Routes on R2
ip route 172.16.201.0 255.255.255.0 172.16.200.201
ip route 172.30.30.0 255.255.255.0 172.16.200.201
!
! Static Routes on R3
ip route 172.20.20.0 255.255.255.0 172.16.200.200
!
! Static Routes on ASA1
route dmz 172.16.200.0 255.255.255.0 172.16.201.201 1
route dmz 172.20.20.0 255.255.255.0 172.16.201.201 1
route dmz 172.30.30.0 255.255.255.0 172.16.201.201 1
```

The command syntaxes for configuring static routes vary slightly between IOS and ASA. The latter does not use the `ip` keyword but needs to explicitly reference the logical name of the interface through which the intended destination is reachable. In Example 5-1, ASA uses the interface called `dmz` to send packets to all the remote networks. Despite these

small differences, the structures of the commands on the two platforms are similar. Both include the destination network/mask pair and the gateway to which the IP datagrams should be forwarded.

Example 5-2 illustrates how to display routes on ASA. It is worth mentioning that the codes appearing on the first output inform the method used to insert a route entry in the routing table. These codes are common to ASA and IOS.

Example 5-2 Visualizing Routing Information on ASA

```
! Displaying the networks reachable through the "dmz" interface on ASA
ASA1# show route dmz
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route
Gateway of last resort is not set
S    172.16.200.0 255.255.255.0 [1/0] via 172.16.201.201, dmz
C    172.16.201.0 255.255.255.0 is directly connected, dmz
S    172.20.20.0 255.255.255.0 [1/0] via 172.16.201.201, dmz
S    172.30.30.0 255.255.255.0 [1/0] via 172.16.201.201, dmz
!
! A form of minimizing the output of the "show route" command
ASA1# show route dmz | begin Gateway
Gateway of last resort is not set
S    172.16.200.0 255.255.255.0 [1/0] via 172.16.201.201, dmz
C    172.16.201.0 255.255.255.0 is directly connected, dmz
S    172.20.20.0 255.255.255.0 [1/0] via 172.16.201.201, dmz
S    172.30.30.0 255.255.255.0 [1/0] via 172.16.201.201, dmz
```

Example 5-3 displays sample outputs of the `show ip route` command on IOS. IOS enables separation of each type of learned route (connected, static, or received through any dynamic routing protocol).

Example 5-3 Visualizing Routing Information on IOS

```
! Displaying directly connected networks on IOS
R2# show ip route connected
    172.16.0.0/24 is subnetted, 2 subnets
C      172.16.200.0 is directly connected, FastEthernet4.200
    172.20.0.0/24 is subnetted, 1 subnets
C      172.20.20.0 is directly connected, Loopback0
!
```

! Displaying static routes on IOS

```
R2# show ip route static
      172.16.0.0/24 is subnetted, 2 subnets
S       172.16.201.0 [1/0] via 172.16.200.201
      172.30.0.0/24 is subnetted, 1 subnets
S       172.30.30.0 [1/0] via 172.16.200.201
!
```

! Displaying details about a specific route on IOS

```
R2# show ip route 172.30.30.0
Routing entry for 172.30.30.0/24
  Known via "static", distance 1, metric 0
  Routing Descriptor Blocks:
    * 172.16.200.201
      Route metric is 0, traffic share count is 1
```

Basic Concepts of Routing Protocols

Network devices running a routing protocol work in a collaborative way to determine how destinations in the internetwork can be reached. The high-level activities involved in the operation of such a protocol follow:

- Each participating routing device tells its directly connected neighbors about the destinations it knows and how far they are located. The information advertised to neighbors can be the routes themselves or link state descriptors, depending on the class of protocol used. The unit of distance is specific to each protocol.
- The receiving device processes the routing information, compares to data it already possesses, performs the pertinent distance calculations and selects the paths that should be installed in the routing table (and later used for actual forwarding).
- After building the tables, the devices need to maintain them. The most important task in this phase is to detect modifications in the topology and propagate this information to neighbors so that they can reflect these changes in their own tables.

Note All destinations that need to be reachable within the scope of an Internetwork must be connected to at least one router. This router is in charge of informing other routing devices about this specific network and acts as a *gateway* for the attached hosts.

One important point to take into account when choosing a routing protocol is to define, in advance, whether it will be used within a single administrative entity or to interconnect distinct entities. An instance of such a routing administration entity is usually denominated as an *autonomous system* (AS).

The suitable choice when routing is confined to one AS is to employ a protocol chosen from the class of *Interior Gateway Protocols* (IGP). Classic examples of IGPs are the

Routing Information Protocol (RIP), Enhanced Interior Gateway Routing Protocol (EIGRP), and Open Shortest Path First (OSPF), which are studied in further detail later in this chapter.

On the opposite range of the spectrum from IGP are the *Exterior Gateway Protocols (EGP)*, designed to interconnect different *autonomous systems*. The only EGP currently used is *Border Gateway Protocol*, version 4 [BGP4]).

Note The concept of AS does not impose the election of a single IGP. A combination of multiple IGPs and static routes can be used within an AS to materialize *internal routing*.

BGP4 focuses on providing flexible routing policies and elaborate filtering capabilities. It has the capability to carry large amounts of routes (absolutely necessary for its application to Internet routing) and privileges scalability, flexibility, and reliability over the fast convergence that is wanted on IGPs.

Note A BGP4 speaker uses the reserved TCP port 179 to establish sessions with its peers, which enables the establishment of sessions with routers that are not directly connected. In cases where a connection to a remote peer is necessary, the IP connectivity needs to be provided by IGPs (or static routes).

Note The analysis of BGP4 is beyond the scope of this book. The purpose of making just a mention in passing was to help differentiate it from IGPs.

A second aspect to be considered when deciding on the routing protocol is the view of the internetwork topology it provides. The protocols currently used belong to two categories, Distance Vector and Link State:

- **Distance Vector:** Independent of the metric used to determine best paths, a DV speaker relies on what its neighbors tell about each destination and does not have a view of individual links within the topology. To calculate the distance to a remote network, they simply add the cost informed by a directly connected router to the cost of reaching the announcing neighbor. This approach, highly influenced by neighbors, is sometimes denominated routing by rumor. Classic members of this class are RIP, Interior Gateway Routing Protocol (IGRP), and EIGRP.
- **Link State:** a link state speaker knows the costs associated to individual links inside a routing domain and does not need to rely on calculations made by its neighbors. In possession of this information, received in the form of Link State Advertisements (LSA), an LS router performs its own computations and derives a complete view of the topology. This approach, where routers in the same domain share the visibility of the overall topology, is also known as routing by propaganda. Two representatives of this family of protocols are OSPF and Intermediate System to Intermediate System (IS-IS).

Figure 5-3 portrays the internetwork topology as seen from a DV protocol standpoint. Figure 5-4 shows the perspective of an LS routing protocol. LS speakers take advantage of the data contained in LSAs to build a detailed view of the topology.

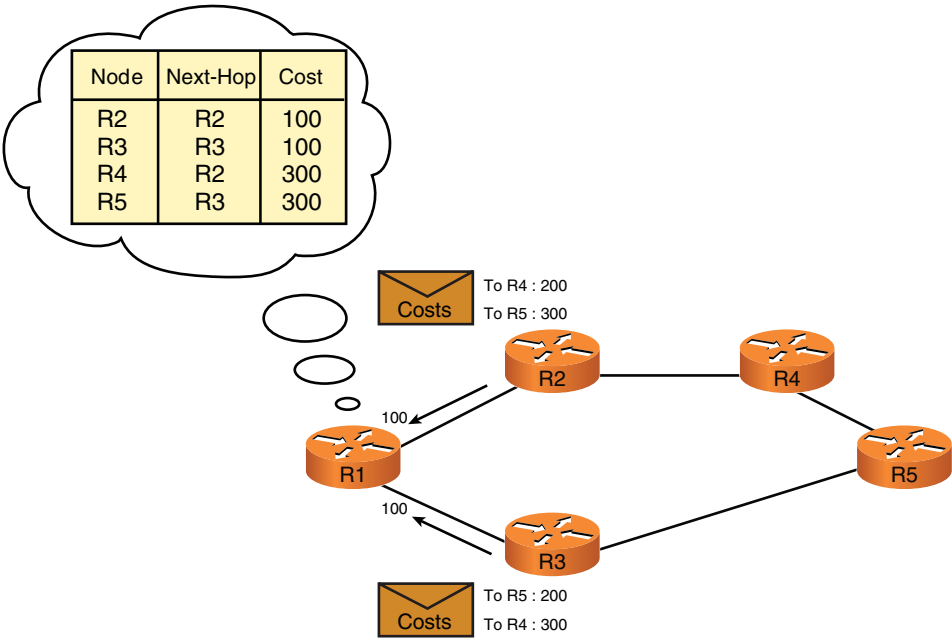


Figure 5-3 Distance Vector Perspective

RIP Overview

The Routing Information Protocol (RIP), an ancient inhabitant of the Internet World, is based on algorithms that were the subject of academic research as early as 1957.

Routers that run this Distance Vector IGP use *hop-count* as the single metric parameter and periodically advertise the entire routing table, not just the changed data. Additionally, RIP timers take too long to declare a route unreachable and remove it from the routing table, therefore greatly impacting convergence. This resource-consuming update method and the slow convergence make the protocol inadequate for large or more complex deployments. On the other hand, its simplicity and widespread availability still provide it with a good share of practical deployments.

IETF RFC 1058 documents the first version of the IP RIP standard. Subsequent RFCs have been published to update some aspects of the original implementation and culminated on RFC 2453, the current standard for RIP version 2. Although one of the key design goals of this second version was to ensure backward compatibility with RIPv1, some relevant improvements were made and deserve explicit mention:

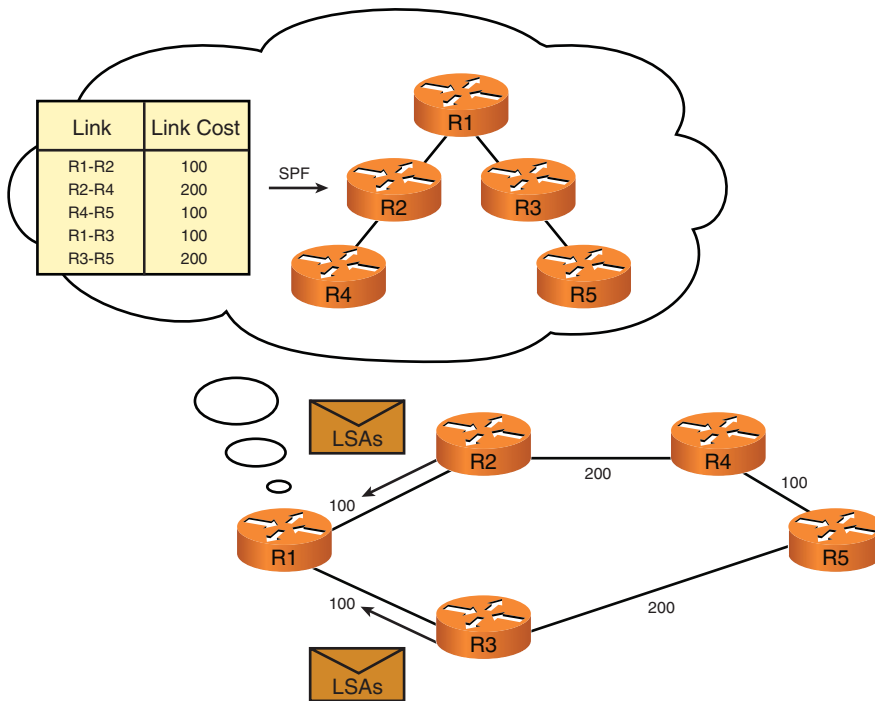


Figure 5-4 *Link State Perspective*

- RIPv2 sends updates to the reserved multicast group 224.0.0.9, instead of using broadcast messages, which excludes routers that are not concerned with RIP from processing these packets. This behavior is even more important on LAN segments that contain many hosts and routers but just a few RIP speakers.
- RIPv2 includes a Subnet Mask field in its packets, thus enabling the use of Variable Length Subnet Masking (VLSM). RIPv1 does not include such a field and enables only routing for subnets in the particular case where all of them use exactly the same mask.
- RIPv2 is subnet mask-aware and, as such, enables routing for discontinuous subnets of the same major (classful) network.
- RIPv2 supports Authentication using clear text or MD5 methods. MD5 is naturally the recommended option.

The default behavior of Cisco routers is to receive both versions of RIP and send only v1 updates on an interface that participates in the routing process. The routers enable explicit configuration of the RIP version used to send or receive updates on a per-interface basis. In case the equipment is instructed to use version 2, it discards any v1 update.

Configuring and Monitoring RIP

The topics presented in this brief introduction and some other important aspects of RIP operation are analyzed by means of some examples.

Figure 5-5 shows the baseline topology for the scenarios discussed in this section. Example 5-4 displays the relevant configurations to enable RIPv1 on the link connecting R3 and ASA1.

Example 5-5 registers how a v1 update is sent and received, clearly showing the use of the broadcast address (255.255.255.255) and reminding you that there is no reference to subnet mask in RIPv1 messages.

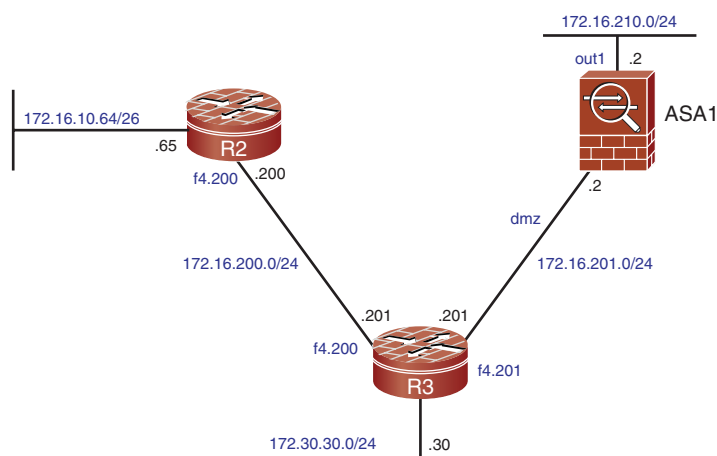


Figure 5-5 *Baseline Topology for RIP Examples*

The RIP Routing Tables (for R3 and ASA1) and Database (for R3) resulting from the combination of Examples 5-4 and 5-5 are registered on Example 5-6. The values between brackets that accompany an entry in the routing table are the Administrative Distance of the protocol (120 in the case of RIP) and the metric (hop count).

Example 5-4 Enabling RIP on R3 and ASA1

! Relevant configurations on R3

```
interface FastEthernet4.200
 encapsulation dot1Q 200
 ip address 172.16.200.201 255.255.255.0
!
interface FastEthernet4.201
```

```

encapsulation dot1Q 201
ip address 172.16.201.201 255.255.255.0
!
router rip
  network 172.16.0.0
  no auto-summary
!
! Relevant configurations on ASA1
interface Vlan210
  nameif out1
  security-level 1
  ip address 172.16.210.2 255.255.255.0
!
interface Vlan201
  nameif dmz
  security-level 50
  ip address 172.16.201.2 255.255.255.0
!
router rip
  network 172.16.0.0
  no auto-summary

```

Example 5-5 Sample RIPv1 Update

```

! R3 sends a v1 update to the broadcast address
RIP: sending v1 update to 255.255.255.255 via FastEthernet4.201 (172.16.201.201)
RIP: build update entries
      subnet 172.16.200.0 metric 1
!
! ASA1 receives the v1 announcement from R3 and updates its RIP database
RIP: received v1 update from 172.16.201.201 on dmz
      172.16.200.0 in 1 hops
RIP-DB: network_update with 172.16.200.0 255.255.0.0 succeeds
RIP-DB: adding 172.16.200.0 0.0.0.0 (metric 1) via 172.16.201.201 on Vlan201 to
RIP database

```

Example 5-6 RIP Routing Table and RIP Database

```

! RIP routes reachable through the "dmz" interface on ASA1
ASA1# show route dmz | begin Gateway
Gateway of last resort is not set
R    172.16.200.0 255.255.255.0 [120/1] via 172.16.201.201, 0:00:22, dmz
C    172.16.201.0 255.255.255.0 is directly connected, dmz
!

```

```

! RIP Routing Table and RIP Database on R3
R3# show ip route rip
      172.16.0.0/24 is subnetted, 3 subnets
R       172.16.210.0 [120/1] via 172.16.201.2, 00:00:18, FastEthernet4.201
!
R3# show ip rip database
172.16.0.0/16      auto-summary
172.16.200.0/24    directly connected, FastEthernet4.200
172.16.201.0/24    directly connected, FastEthernet4.201
172.16.210.0/24    [1] via 172.16.201.2, 00:00:15, FastEthernet4.201

```

Example 5-7 focuses on registering the default timers used by RIP (v1 and v2), making you aware of how the protocol behaves for version control. When RIP is enabled on a Cisco routing device, it accepts both v1 and v2 updates but sends only v1.

The default update timer is 30 seconds, meaning that the entire routing table (with the exception of routes removed by the split-horizon rule) will be sent every half minute. The *invalidation or expiration timer* (180 seconds) defines how long a RIP speaker maintains an entry that was not updated in its routing table. In the event that information about the entry is not present on six consecutive updates (180 seconds), the corresponding hop count value is changed to 16, a technique used by RIP to declare the route unreachable. The other RIP timers are discussed in Example 5-14.

Example 5-7 Default RIP Timers and Version Control

```

R3# show ip protocols
Routing Protocol is "rip"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 8 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: rip
  Default version control: send version 1, receive any version
    Interface          Send Recv Triggered RIP Key-chain
    FastEthernet4.200    1     1  2
    FastEthernet4.201    1     1  2
[output suppressed]
!
! Default Behavior : receives any version (and joins RIPv2 reserved multicast group)
R3# show ip interface f4.200 | include Multicast
Multicast reserved groups joined: 224.0.0.9

```

RIPv1 is a classful protocol and its packet header does not include a subnet mask field. However, it can route for subnets of the same major network when all the subnets have equal masks, as shown in Example 5-6.

Examples 5-8 and 5-9 illustrate two limitations of RIPv1, namely its inability to support *VLSM* and discontinuous subnets.

Example 5-8 shows that the 172.16.10.64/26 subnet cannot be announced over the /24 link that connects R2 and R3. This is because a RIPv1 speaker has no way to define the frontier between subnet and host bits and assumes that the mask to be used should be a copy of that configured on its interfaces.

R2 and R3, as shown in Figure 5-5, are connected through a subnet of the major network 172.16.0.0/16. This topology also shows that the subnet 172.30.30.0/24 is attached to R3. The addition of another subnet of 172.30.0.0/16 on R2, as shown in Example 5-9, makes this *class B* network become discontinuous. This happens because R2 considers itself a border element between 172.30.0.0/16 and 172.16.0.0/16, and advertises only the major network 172.30.0.0/16 to R3. Because there is no subnet information on the update from R2, R3 will ignore it.

Example 5-8 *VLSM Is Not Supported by RIPv1*

```
! Enabling RIP on R2. Notice the /26 subnet of major network 172.16.0.0/16
interface Loopback10
 ip address 172.16.10.65 255.255.255.192
!
interface FastEthernet4.200
 encapsulation dot1Q 200
 ip address 172.16.200.200 255.255.255.0
!
router rip
 passive-interface default
 no passive-interface FastEthernet4.200
 network 172.16.0.0
 no auto-summary
!
! Announcing a new major network (172.30.0.0/16) on R3
interface Loopback0
 ip address 172.30.30.30 255.255.255.0
!
router rip
 network 172.30.0.0
!
!! R2 receives RIP routes from R3 (172.16.200.201) but does not send any
RIP: received v1 update from 172.16.200.201 on FastEthernet4.200
    172.16.201.0 in 1 hops
    172.16.210.0 in 2 hops
```

```

    172.30.0.0 in 1 hops
!
R2# show ip route rip
    172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
R    172.16.210.0/24 [120/2] via 172.16.200.201, 00:00:05, FastEthernet4.200
R    172.16.201.0/24 [120/1] via 172.16.200.201, 00:00:05, FastEthernet4.200
R    172.30.0.0/16 [120/1] via 172.16.200.201, 00:00:05, FastEthernet4.200
!
! The /26 subnet on R2 cannot be announced using RIPv1 on a /24 link (f4.200)
RIP: sending v1 update to 255.255.255.255 via FastEthernet4.200 (172.16.200.200)
RIP: build update entries - suppressing null update
!
! R3 does not receive any update from R2 (only from ASA1)
R3# show ip route rip
    172.16.0.0/24 is subnetted, 3 subnets
R    172.16.210.0 [120/1] via 172.16.201.2, 00:00:09, FastEthernet4.201

```

Example 5-9 RIPv1 Does Not Support Discontinuous Subnets

```

! Adding a new /24 subnet on R2 (making 172.30.0.0/16 to become discontinuous)
interface Loopback30
 ip address 172.30.2.2 255.255.255.0
!
router rip
 network 172.30.0.0
!
! R2 starts announcing the major network 172.30.0.0/16
RIP: sending v1 update to 255.255.255.255 via FastEthernet4.200 (172.16.200.200)
RIP: build update entries
    network 172.30.0.0 metric 1
!
! R3 ignores the update, for it is not a subnet (just a major network it
already knows)
RIP: received v1 update from 172.16.200.200 on FastEthernet4.200
    172.30.0.0 in 1 hops
!
R3# show ip route rip
    172.16.0.0/24 is subnetted, 3 subnets
R    172.16.210.0 [120/1] via 172.16.201.2, 00:00:13, FastEthernet4.201

```

Example 5-7 taught that the RIP-enabled interfaces of a Cisco routing device are, by default, prepared to receive v1 and v2 updates but to send only v1.

In Example 5-10, R2 is configured to send v2 updates out of the f4.200 interface. You can see that the updates now include the subnet mask and are directed to the reserved

Multicast group 224.0.0.9. The 172.30.2.0/24 subnet becomes visible on R3 routing table, solving the issue raised in Example 5-9.

When RIPv2 is enabled inside the routing process (using the **version 2** command), a Cisco device starts considering any received v1 update as illegal. This behavior is shown in Example 5-11.

Example 5-10 *Enabling v2 Updates on an Interface*

```
! Enabling R2 to send v2 updates on f4.200 (R3 receives all versions by default)
interface FastEthernet4.200
ip rip send version 2
!
! The v2 update includes the subnet mask and is sent to the multicast group
224.0.0.9
RIP: sending v2 update to 224.0.0.9 via FastEthernet4.200 (172.16.200.200)
RIP: build update entries
      172.16.10.64/26 via 0.0.0.0, metric 1, tag 0
      172.30.2.0/24 via 0.0.0.0, metric 1, tag 0
!
! R3 receives the RIPv2 update and includes in its routing table
RIP: received v2 update from 172.16.200.200 on FastEthernet4.200
      172.16.10.64/26 via 0.0.0.0 in 1 hops
      172.30.2.0/24 via 0.0.0.0 in 1 hops
!
R3# show ip route rip
      172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
R       172.16.210.0/24 [120/1] via 172.16.201.2, 00:00:03, FastEthernet4.201
R       172.16.10.64/26 [120/1] via 172.16.200.200, 00:00:13, FastEthernet4.200
      172.30.0.0/16 is variably subnetted, 4 subnets, 2 masks
R       172.30.2.0/24 [120/1] via 172.16.200.200, 00:00:13, FastEthernet4.200
```

Example 5-11 *Instructing a Router to Use Only RIP Version 2*

```
! Instructing R2 to use only RIPv2. Received V1 updates are ignored.
router rip
  version 2
!
RIP: ignored v1 packet from 172.16.200.201 (illegal version)
!
R2# show ip protocols
Routing Protocol is "rip"
! [output suppressed]
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface          Send Recv Triggered RIP Key-chain
```

```
FastEthernet4.200      2      2
[output suppressed]
```

Example 5-12 shows a method for injecting a default route into the RIP process. Example 5-13 brings a sample configuration of a *summary address* on a RIP speaker (R3) and documents the corresponding effects on ASA1.

Example 5-12 *Originating the Default Route on RIP*

```
! Originating the default route on ASA1 (all routers already configured for v2
only)
router rip
  default-information originate
!
ASA1# show rip database
0.0.0.0 0.0.0.0      auto-summary
0.0.0.0 0.0.0.0      redistributed      [0] via 0.0.0.0,
[output suppressed]
!
! Default route and gateway of last resort information on R3

R3# show ip route rip | include 0.0.0.0
R*  0.0.0.0/0 [120/1] via 172.16.201.2, 00:00:03, FastEthernet4.201
!
R3#show ip route | include Gateway
Gateway of last resort is 172.16.201.2 to network 0.0.0.0
```

Example 5-13 *Generating a Summary Route out of an Interface*

```
! Generating a summary on R3 towards ASA1
interface FastEthernet4.201
  ip summary-address rip 172.30.0.0 255.255.0.0
!
! Immediate effect of this new configuration on R3
RIP: sending v2 flash update to 224.0.0.9 via FastEthernet4.201 (172.16.201.201)
RIP: build flash update entries
      172.30.0.0/16 via 0.0.0.0, metric 1, tag 0
!
! What is seen on R3 a bit later
RIP: sending v2 update to 224.0.0.9 via FastEthernet4.201 (172.16.201.201)
RIP: build update entries
      172.16.10.64/26 via 0.0.0.0, metric 2, tag 0
      172.16.200.0/24 via 0.0.0.0, metric 1, tag 0
      172.30.0.0/16 via 0.0.0.0, metric 1, tag 0
```

```

!
R3# show ip protocols
Routing Protocol is "rip"
[output suppressed]
Automatic network summarization is not in effect
Address Summarization:
  172.30.0.0/16 for FastEthernet4.201
[output suppressed]
!
! New routing table on ASA1
ASA1# show route dmz | begin Gateway
Gateway of last resort is not set
R   172.16.200.0 255.255.255.0 [120/1] via 172.16.201.201, 0:00:28, dmz
C   172.16.201.0 255.255.255.0 is directly connected, dmz
R   172.16.10.64 255.255.255.192 [120/2] via 172.16.201.201, 0:00:28, dmz
R   172.30.0.0 255.255.0.0 [120/1] via 172.16.201.201, 0:00:28, dmz

```

Example 5-14 illustrates how a **distribute-list** applied in the *incoming* direction can be used to limit the routes installed in the routing table. ASA1 receives three updates from R3 but installs only the one that obeys the rules stated by the **distribute-list**. This example complements Example 5-7 by illustrating the sequence of timers used by RIP to withdraw a route from the routing table.

When the invalidation timer expires, the route enters into a *holddown* state and is advertised as unreachable. While in the holddown state, the route keeps being used for packet forwarding and no updates for better paths are accepted. The *flush* (or *garbage collection*) timer, is the amount of time that must elapse before the route is removed from the routing table.

Example 5-14 shows that the route to 172.16.10.64/26 enters the holddown state, after being invalidated. A bit later, the garbage collection timer expires, and the route is finally removed from the routing table.

Example 5-14 Defining and Applying a distribute-list

```

! Distribute-list on ASA1 permitting only network 172.30.0.0/16 and its subnets
access-list RIP-IN standard permit 172.30.0.0 255.255.0.0
!
router rip
distribute-list RIP-IN in
!
! ASA1 receives 03 routes from R3 but installs only one
RIP: received v2 update from 172.16.201.201 on dmz
      172.16.10.64 255.255.255.192 via 0.0.0.0 in 2 hops
      172.16.200.0 255.255.255.0 via 0.0.0.0 in 1 hops

```



```

172.30.0.0 255.255.0.0 via 0.0.0.0 in 1 hops
RIP-DB: network_update with 172.30.0.0 255.255.0.0 succeeds
RIP-DB: adding 172.30.0.0 255.255.0.0 (metric 1) via 172.16.201.201 on Vlan201 to
RIP database

RIP-DB: invalidated route of 172.16.10.64 255.255.255.192 via 172.16.201.201
delete route to 172.16.10.64 via 172.16.201.201, rip metric [120/2]no routes to
64.10.16.172, entering holddown
RIP-DB: rip_destroy_rdb Remove 172.16.10.64 255.255.255.192, (metric 4294967295)
via 172.16.201.201, Vlan201
[output suppressed ]
!
ASA1# show route dmz | begin Gateway
Gateway of last resort is not set
R    172.16.200.0 255.255.255.0 is possibly down, routing via 172.16.201.201, dmz
C    172.16.201.0 255.255.255.0 is directly connected, dmz
R    172.16.10.64 255.255.255.192 is possibly down, routing via 172.16.201.201, dmz
R    172.30.0.0 255.255.0.0 [120/1] via 172.16.201.201, 0:00:24, dmz
!
! Subsequent effects on ASA1 (subnets not allowed by the distribute-list are
removed)
RIP-DB: garbage collect 172.16.10.64 255.255.255.192
RIP-DB: rip_destroy_ndb destroy 172.16.10.64 255.255.255.192, (best metric
4294967295)
delete subnet route to 172.16.10.64 255.255.255.192
!
ASA1# show route dmz | begin Gateway
Gateway of last resort is not set
C    172.16.201.0 255.255.255.0 is directly connected, dmz
R    172.30.0.0 255.255.0.0 [120/1] via 172.16.201.201, 0:00:02, dmz

```

EIGRP Overview

For almost everyone that first got in touch with internetworking in the 21st century, any reference to a protocol stack other than TCP/IP might sound strange and even old fashioned. However, up to the early 1990s, it was common to find Enterprise networks based on IPX/SPX or AppleTalk, for example. Any company that had a mix of protocols in its infrastructure was supposed to solve the routing problem multiple times, one for each routed protocol family.

By that time, Cisco was already famous and respected as a vendor of multiprotocol routers and decided to create an IGP to simultaneously provide routing services for IP, IPX and AppleTalk. IPX and AppleTalk implementations have almost disappeared, and the challenge in today's networks is practically restricted to finding IGPs that can support IPv4 and IPv6. And, for those that own Cisco equipment, EIGRP remain a good option to build scalable and reliable IP internetworks.

EIGRP bases its operation on an algorithm named Diffused Update Algorithm (DUAL) and is classified as an Advanced Distance Vector Protocol. EIGRP updates provide routers with the knowledge of a *vector metric* containing six parameters (bandwidth, delay, reliability, load, MTU, and hop count). In possession of the vector metric for a given destination, an EIGRP router computes the associated distance and calls it the *composite metric*.

Although EIGRP still falls on the distance-vector group, the knowledge of this set of parameters helps to produce a more accurate view of the internetwork than solely relying on hop-count information.

The value of each vector metric component is informed separately on routing updates, and the receiving router uses a mathematical formula to compute a composite metric that expresses how close the destination is to the router. In a situation in which two or more routes to the same destination exist, that with the lowest composite metric will be selected.

Note Even though an EIGRP speaker receives the six parameters of the vector metric on updates, it, by default, uses only bandwidth and delay on the computation of the composite metric (also known as distance). The composite metric is calculated as a scaled sum of the *lowest value of bandwidth* and the *cumulative delay* along the path to the destination. (The scaling factor for EIGRP is 256).

In addition to the routing table, each EIGRP router maintains a *topology table* containing all routes announced by neighbors. For every destination it learns about, the EIGRP device moves the path with the lowest composite metric (best path) to the routing table. The existence of the topology table enables a quick selection of an alternative path, in case the original best path becomes unavailable, and contributes to the fast convergence characteristic of EIGRP.

When a router that runs a conventional distance-vector protocol loses a route to a remote destination, it counts on its neighbors to search for alternative paths and share these new findings. EIGRP speakers do not adopt this approach of mere spectators and actively send *Queries* to neighbors when a destination, to which no precomputed backup path exists (in the topology table), disappears from the routing table.

EIGRP is effective for the updates it sends. Some notable features follow:

- **Nonperiodic updates:** Routing updates are associated with topology or metric changes and are not sent on predefined intervals.
- **Partial updates:** Instead of sending the entire routing table, an EIGRP speaker advertises only routes that were subject to change.

Note The reader should not confuse the EIGRP topology table with the *Link State Database* (LSDB) used by protocols such as OSPF and IS-IS. Link state protocols use the

knowledge of attributes related to each link in a routing domain to compute the best path. The EIGRP Topology Table has only visibility of complete paths (not of individual links) and still reflects the perspective of directed attached neighbors.

Configuring and Monitoring EIGRP

EIGRP is a relatively easy to configure protocol. You need to become familiar with a few commands, but beware that this might create a somewhat careless design. The EIGRP scenarios covered here serve as reference for real-life implementation tasks and, hopefully, can contribute to consolidate the most relevant concepts.

The network topology in Figure 5-6 describes the fundamentals of EIGRP operation and exemplifies the composite metric calculations. In this figure, NBW and NDLY are both abbreviations for Normalized Bandwidth and Normalized Delay.

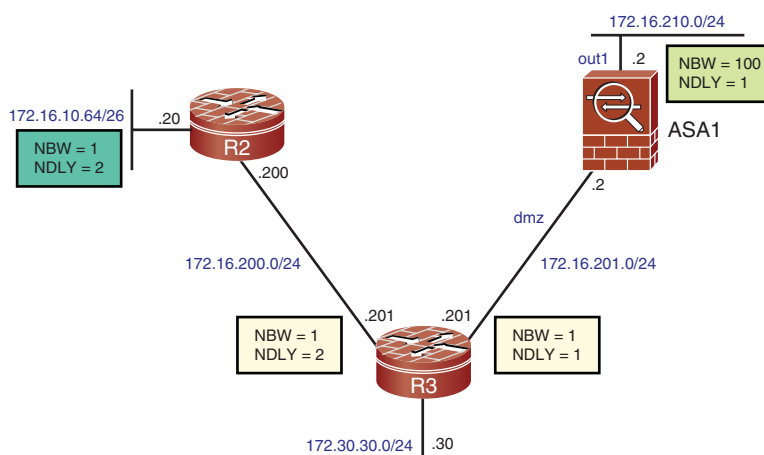


Figure 5-6 Reference Topology for the Analysis of EIGRP Operation

EIGRP Configuration Fundamentals

Example 5-15 documents the typical commands for enabling EIGRP on a practical usage scenario. The first requirement is to choose a number for the EIGRP process (*AS number*) and assign the same number to the intended neighboring routers.

The **network** command lies at the core of EIGRP configuration and, as such, deserves a more detailed explanation:

- The first usage of a network statement is to tell the EIGRP process that the particular network might be used for establishing an adjacency and, naturally, for later exchanging routing packets. The adjacency between R3 and R2, for instance, is established over the 172.16.200.0/24 link.

- The second usage of a network statement is to announce a specific segment over all other EIGRP-enabled interfaces. Network 172.16.210.0/24, for example, is connected to ASA1 as a stub segment (no other routers attached to it) and is advertised to R3 over 172.16.201.0/24.
- At this point, you should revisit the RIP scenarios of the previous section and observe that the network command defined under the RIP process does not bring a mask option. Whenever configured, it includes the major network and its associated subnets. EIGRP, on the contrary, enables for the definition of a subnet and mask pair, thus providing more granular control of routing.
- Cisco routers use a wildcard mask whereas the ASA family uses a regular network mask.
- A **passive-interface** statement is directly related to the **network** command and instructs the EIGRP process to act in listening mode on a particular segment. It receives updates through it but does not advertise any route.

EIGRP requires each router to define a 32-bit identifier called *router ID* at process start-up time. The **eigrp router-id** command is recommended because it takes precedence over any default procedure that a Cisco routing device might use for selecting this ID. In spite of being mandatory, this ID seldom appears on EIGRP **show** commands. The notable situation in which the router ID is directly used is to identify the EIGRP speaker that injects external routes (via redistribution) into the routing process. The **OSPF Overview** section shows that the Router ID plays a much more important role in that protocol than it does in EIGRP.

Best practice is to reserve the IP address of a loopback interface (because of its *always-on* nature) to the router ID of any routing protocols that support it. The same is true for management and control protocols such as SNMP, TACACS+ and NTP.

Example 5-15 also shows how to visualize the EIGRP neighbors table and the reserved IP Multicast Group, 224.0.0.10.

Example 5-15 *Baseline EIGRP Configuration*

```
! R2 baseline EIGRP configuration
router eigrp 300
no auto-summary
eigrp router-id 172.20.20.20
network 172.16.200.0 0.0.0.255
network 172.20.20.0 0.0.0.255
!
! R3 baseline EIGRP configuration
router eigrp 300
no auto-summary
eigrp router-id 172.30.30.30
network 172.16.200.0 0.0.0.255
```

```

network 172.16.201.0 0.0.0.255
network 172.30.30.0 0.0.0.255
!
! ASA1 baseline EIGRP configuration (uses network mask instead of wildcard mask)
router eigrp 300
no auto-summary
eigrp router-id 172.16.201.2
network 172.16.201.0 255.255.255.0
network 172.16.210.0 255.255.255.0
passive-interface out1
!
! EIGRP neighbors of R3's
R3# show ip eigrp neighbor
IP-EIGRP neighbors for process 300

```

H	Address	Interface	Hold (sec)	Uptime	SRTT (ms)	RTO	Q	Seq Cnt	Num
1	172.16.200.200	Fa4.200	11	03:03:44	1	200	0	30	
0	172.16.201.2	Fa4.201	14	03:03:52	1	200	0	58	

```

!
! Reserved IP Multicast Group used by EIGRP
R3#show ip interface f4.200 | include Multicast
Multicast reserved groups joined: 224.0.0.10

```

Understanding EIGRP Metrics

Example 5-16 registers the baseline configurations used in EIGRP metric computation for the topology represented in Figure 5-6. The two relevant parameters are bandwidth and delay, which have specific default values for each type of interface present on a Cisco routing device. Considering that EIGRP composite metrics tend to be large, convenient values have been chosen for the examples of this section.

For better understanding of the scenarios that follow, note these particularities:

- The unit of the delay parameter for interface configuration is tens of microseconds and will be referred to as Normalized Delay (NDLY). It means that, when the command `delay 2` is issued, the actual value is $2 \times (10 \text{ usec}) = 20 \text{ usec}$. The real value may be visualized with the `show interface` command, as illustrated in this example.
- The unit of the bandwidth parameter is Kbps. The Normalized Bandwidth (NBW) is calculated dividing 10 Gbps (reference bandwidth) by the interface bandwidth. The argument 10000000 for the `bandwidth` command (configured, for instance, on R2's loopback 0 interface) means 10000000 Kbps and results in a value of 1 for the NBW.
- The composite metric derives from a scaled sum of the minimum available bandwidth and the somatory of delays all over the path from the router to the destination. The scaling factor is 256, suggesting that, whenever a large metric is seen, the first step to understand how the numbers were generated is to divide it by 256.

- In EIGRP terminology, the lowest calculated metric to a certain destination is called its Feasible Distance (FD). The FD for a directly connected route is simply $256 \times [\text{NBW} + \text{NDLY}]$. A sample calculation is shown in Example 5-16 for the attached network 172.20.20.0/24 on R2. In this case, $\text{FD} = 256 \times [1 + 2] = 768$.
- For every destination it advertises to its neighbors, an EIGRP speaker inserts a distance value. This value is always the smallest one that the routers knows about and is named the Reported Distance (RD) to the destination. The RD for an attached route equals the FD. In Example 5-17, R2 reports to R3 a distance $\text{RD} = 768 = [256 + 512] = 256 \times [1 + 2]$ for network 172.20.20.0/24.

Note Care should be taken when dealing with the Bandwidth (BW) component for metric calculation. The BW information is on the denominator of a fraction that has as numerator the reference bandwidth (10 Gbps). This implies that the lowest bandwidth (BW) generates the highest Normalized Bandwidth (NBW).

When an EIGRP router receives an advertisement (and the associated RD), it computes the new vector metric parameters and, later, the correspondent composite metric. A summary of these calculations follows:

- $\text{New NBW} = \text{Max} (\text{Reported NBW}, \text{Interface NBW})$
- $\text{New NDLY} = \text{Reported NDLY} + \text{Interface NDLY}$
- $\text{FD} = 256 \times [\text{New NBW} + \text{New NDLY}]$

Considering the topology of Figure 5-6 and the output of Example 5-17, the FD (Feasible Distance) to 172.20.20.0/24, under R3's perspective, is calculated as follows:

$\text{New NBW} = \text{Max} (1, 1) = 1$

$\text{New NDLY} = 2 + 2 = 4$

$\text{FD} = 256 \times [1 + 4] = 1280$

Another sample calculation is presented for destination 172.16.210.0/24, announced by ASA1 to R3. ASA1 reports a distance $\text{RD} = 256 \times (100 + 1) = 25856$. The interface parameters for R3 to reach ASA1 are $\text{NBW} = 1$ and $\text{NDLY} = 1$, as documented in Figure 5-6.

$\text{New NBW} = \text{Max} (100, 1) = 100$

$\text{New NDLY} = 1 + 1 = 2$

$\text{FD} = 256 [100 + 2] = 26112$

Note In the topology represented in Figure 5-6, ASA1 is referred to as a *downstream router* (from R3's point of view) for subnet 172.16.210.0/24 because it is closer to this destination than R3. Yet from R3's standpoint, R2 is an *upstream router* (for 172.16.210.0/24) because it uses R3 to forward packets to this reference network.

Example 5-17 also shows the EIGRP topology and routing tables on R3. The numbers between brackets that accompany an entry in the routing table are respectively the Administrative Distance of the protocol (90 for EIGRP) and the metric (*composite metric*). In the topology table, the numbers between brackets are FD/RD.

Note Although it may sound counterintuitive, a *Passive* state for an EIGRP destination means that the correspondent route is valid, converged and ready to be used. A move to the *Active* state denotes that EIGRP no longer knows a loop-free path to the network and therefore needs to *actively search* for one.

Example 5-16 Visualizing Local EIGRP Metric Parameters (DLY and BW)

```
! Configuring and verifying local EIGRP metrics on R2
interface Loopback0
  bandwidth 10000000
  ip address 172.20.20.20 255.255.255.0
  delay 2
!
R2# show interface loopback 0 | include BW
    MTU 1514 bytes, BW 10000000 Kbit/sec, DLY 20 usec,
!
! Feasible distance for one of R2's directly connected networks, 172.20.20.0/24
R2# show ip eigrp topology
IP-EIGRP Topology Table for AS(300)/ID(172.20.20.20)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
r - reply Status, s - sia Status
P 172.20.20.0/24, 1 successors, FD is 768
    via Connected, Loopback0
!
! Examples of Local Metrics on R3
interface FastEthernet4.200
  bandwidth 10000000
  encapsulation dot1Q 200
  ip address 172.16.200.201 255.255.255.0
  delay 2
!
R3# show interfaces FastEthernet 4.200 | include BW
    MTU 1500 bytes, BW 10000000 Kbit/sec, DLY 20 usec,
!
R3# show interfaces FastEthernet 4.201 | include BW
    MTU 1500 bytes, BW 10000000 Kbit/sec, DLY 10 usec,
!
! Example of Local Metrics on ASA1
interface Vlan210
```

```

nameif out1
security-level 1
ip address 172.16.210.2 255.255.255.0
delay 1
!
ASA1# show interface vlan 210 | include BW
    Hardware is EtherSVI, BW 100 Mbps, DLY 10 usec

```

Example 5-17 *Calculating Feasible Distance from the Reported Distance*

```

! R2 announces 172.20.20.0/24 to R3 with RD = 768 = (256 + 512)
IP-EIGRP(Default-IP-Routing-Table:300): 172.20.20.0/24 - do advertise out
FastEthernet4.200
IP-EIGRP(Default-IP-Routing-Table:300): Int 172.20.20.0/24 metric 768 - 256 512
!
! R3 receives the route from R2 and includes its distance to R2 to derive the FD
IP-EIGRP(Default-IP-Routing-Table:300): Processing incoming UPDATE packet
IP-EIGRP(Default-IP-Routing-Table:300): Int 172.20.20.0/24 M 1280 - 256 1024 SM
768 - 256 512
IP-EIGRP(Default-IP-Routing-Table:300): route installed for 172.20.20.0  ()
IP-EIGRP(Default-IP-Routing-Table:300): Int 172.20.20.0/24 metric 1280 - 256 1024
!
! R3 receives update regarding 172.16.210.0/24 from ASA1 and calculates the FD
IP-EIGRP(Default-IP-Routing-Table:300): Processing incoming UPDATE packet
IP-EIGRP(Default-IP-Routing-Table:300): Int 172.16.210.0/24 M 26112 - 25600 512
SM 25856 - 25600 256
!
! Visualizing EIGRP Routes learned by R3
R3# show ip route eigrp
    172.16.0.0/24 is subnetted, 3 subnets
D    172.16.210.0 [90/26112] via 172.16.201.2, 00:32:20, FastEthernet4.201
    172.20.0.0/24 is subnetted, 1 subnets
D    172.20.20.0 [90/1280] via 172.16.200.200, 00:20:14, FastEthernet4.200
!
! Visualizing the EIGRP Topology on R3
R3# show ip eigrp topology
IP-EIGRP Topology Table for AS(300)/ID(172.30.30.30)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status
P 172.16.210.0/24, 1 successors, FD is 26112
    via 172.16.201.2 (26112/25856), FastEthernet4.201
P 172.20.20.0/24, 1 successors, FD is 1280
    via 172.16.200.200 (1280/768), FastEthernet4.200
!

```


! Visualizing the details for a specific route in the EIGRP Topology Table

```

R3# show ip eigrp topology 172.16.210.0/24
IP-EIGRP (AS 300): Topology entry for 172.16.210.0/24
State is Passive, Query origin flag is 1, 1 Successor(s), FD is 26112
Routing Descriptor Blocks:
  172.16.201.2 (FastEthernet4.201), from 172.16.201.2, Send flag is 0x0
    Composite metric is (26112/25856), Route is Internal
      Vector metric:
        Minimum bandwidth is 100000 Kbit
        Total delay is 20 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1

```

Note The term *successor* is used to refer to the neighbor that the current router uses as its next-hop toward a certain destination. In the topology table of Example 5-17, 172.16.201.2 (ASA1) is the successor of R3 for destination 172.16.210.0/24.

Note A neighbor meets the *Feasibility Condition* if its Reported Distance (RD) to a destination is lower than the FD.

A neighbor that satisfies the Feasibility Condition is called a *Feasible Successor*.

In the event that an EIGRP speaker receives an increased metric (for a destination) from its successor, it looks for (precomputed) feasible successors in its topology table. If it finds one, the new route is immediately selected and updates are sent to the neighbors to register the topology modification.

These terms may be applied to Example 5-17, always considering R3's perspective. The Feasible Distance to 172.20.20.0/24 is 1024. R2 is currently the successor for this destination. Any neighbor of R3's that reports a distance lower than 1024 ($RD < 1024$) to this network becomes a feasible successor.

Note Feasible Successors are always *downstream* routers. This concept is important to avoid routing loops because a router does not choose a path that includes itself. (Such a path naturally has a distance larger than its FD to the destination under analysis).

Redistributing Routes into EIGRP

After analyzing the basics of EIGRP configuration and metric calculation, now consider the important topic of route redistribution. One distinctive characteristic is that it uses a different Administrative Distance for internal (90) and external (170) routes.

Figure 5-7 shows the reference topology for EIGRP redistribution analysis, highlighting the metric parameters associated to each network. The baseline configuration for the devices involved (before any redistribution is performed) are shown in Example 5-15.

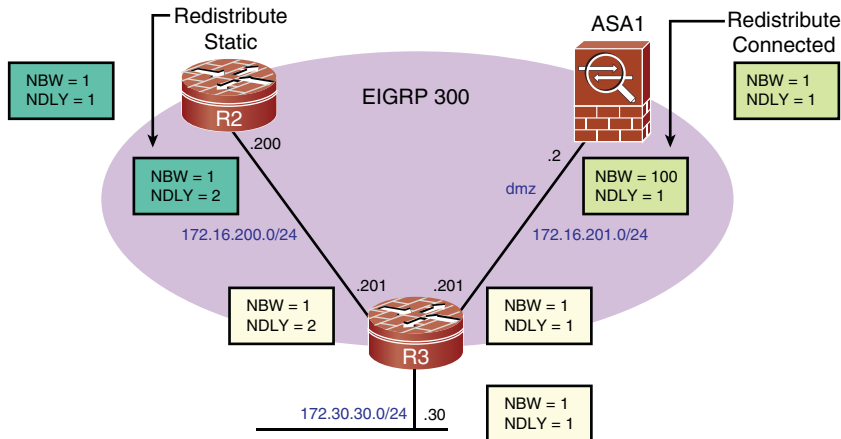


Figure 5-7 Topology for Analysis of EIGRP Redistribution

Example 5-18 shows the required configuration for redistributing connected routes into EIGRP and how the prefix 192.168.1.0/24 displays on ASA1's topology table. Some important data deserve special comments:

- A route-map controls which connected routes should be redistributed.
- Values for five composite metric attributes, in the order Bandwidth, Delay, Reliability, Load, MTU, must be included when performing redistribution. In this example, BW = 10000000 Kbps (yielding a value NBW = 1), DLY = 10 usec (NDLY = 1), Reliability = 255, Load = 1, and MTU = 1500. These last three values were carefully chosen to instruct EIGRP to ignore the contribution of parameters other than BW and DLY on the metric calculation for redistributed routes.

R3's routing table on Example 5-18 illustrates that 192.168.1.0/24 is seen as an external route (D EX). More details about the route are provided in the topology table.

Example 5-18 Redistribution of Connected Routes into EIGRP

```
! Additional configuration on ASA1 for redistribution of connected routes into EIGRP
access-list 1 standard permit 192.168.1.0 255.255.255.0
!
route-map CONNECTED1 permit 10
  match ip address 1
!
router eigrp 300
```

```

redistribute connected metric 10000000 1 255 1 1500 route-map CONNECTED1
!
ASA1# show eigrp topology
EIGRP-IPv4 Topology Table for AS(300)/ID(172.16.201.2)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply, r - reply
Status, s - sia Status
P 192.168.1.0 255.255.255.0, 1 successors, FD is 512
    via Rconnected (512/0)
!
! External route from ASA1 is displayed on R3's routing table with the code "D EX"
R3# show ip route eigrp
    172.16.0.0/24 is subnetted, 3 subnets
D       172.16.210.0 [90/26112] via 172.16.201.2, 01:06:49, FastEthernet4.201
    172.20.0.0/24 is subnetted, 1 subnets
D       172.20.20.0 [90/1280] via 172.16.200.200, 01:27:48, FastEthernet4.200
D EX 192.168.1.0/24 [170/768] via 172.16.201.2, 00:38:51, FastEthernet4.201
!
! Details about the external route on R3's Topology Table
R3# show ip eigrp topology 192.168.1.0/24
IP-EIGRP (AS 300): Topology entry for 192.168.1.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 768
  Routing Descriptor Blocks:
    172.16.201.2 (FastEthernet4.201), from 172.16.201.2, Send flag is 0x0
      Composite metric is (768/512), Route is External
      Vector metric:
        Minimum bandwidth is 10000000 Kbit
        Total delay is 20 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
  External data:
    Originating router is 172.16.201.2
    AS number of route is 0
    External protocol is Connected, external metric is 0
    Administrator tag is 0 (0x00000000)

```

Redistribution of static routes into EIGRP, as shown in Example 5-19, is completely analogous to that of connected routes. As in the previous case, the control of redistributed prefixes with a **route-map** is highly recommended.

Example 5-19 *Redistribution of Static Routes into EIGRP*

```
! Defining static routes on R2
```

```

ip route 172.20.2.0 255.255.255.0 Null0 tag 2222
ip route 172.20.3.0 255.255.255.0 Null0 tag 2222
!
! This route-map specifies that only routes with tag 2222 are to be redistributed
route-map STATIC1 permit 10
  match tag 2222
!
! Redistributing static routes into EIGRP (NBW = 1 ; NDLY = 1) on router R2
router eigrp 300
  redistribute static metric 10000000 1 255 1 1500 route-map STATIC1
!
! Resultant EIGRP Routing Table on R3
R3# show ip route eigrp
      172.20.0.0/24 is subnetted, 3 subnets
D       172.20.20.0 [90/1280] via 172.16.200.200, 02:12:52, FastEthernet4.200
D EX    172.20.2.0 [170/1024] via 172.16.200.200, 00:10:03, FastEthernet4.200
D EX    172.20.3.0 [170/1024] via 172.16.200.200, 00:10:03, FastEthernet4.200
D EX 192.168.1.0/24 [170/768] via 172.16.201.2, 00:30:37, FastEthernet4.201

```

Generating a Summary EIGRP Route

The generation of an EIGRP Summary Route is done at the interface level, as shown in Example 5-20. The configuration is similar to RIP's (see Example 5-13).

Example 5-20 EIGRP Summary Route

```

! Configuring a /23 summary-address on R2
interface FastEthernet4.200
ip summary-address eigrp 300 172.20.2.0 255.255.254.0 5

! Redistributed /24 static routes and correspondent /23 summary on R2's Topology Table
R2# show ip eigrp topology
IP-EIGRP Topology Table for AS(300)/ID(172.20.20.20)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply, r - reply
Status, s - sia Status
P 172.20.2.0/23, 1 successors, FD is 512
    via Summary (512/0), Null0
P 172.20.2.0/24, 1 successors, FD is 512, tag is 2222
    via Rstatic (512/0)
P 172.20.3.0/24, 1 successors, FD is 512, tag is 2222
    via Rstatic (512/0)
!
! The summary address displays as an EIGRP route with a Null0 next-hop
R2# show ip route | include Null

```

```

S      172.20.2.0/24 is directly connected, Null0
D      172.20.2.0/23 is a summary, 00:02:33, Null0
S      172.20.3.0/24 is directly connected, Null0
!
! R3 includes the summary received from R2 in its routing table
R3# show ip route eigrp
      172.20.0.0/16 is variably subnetted, 2 subnets, 2 masks
D      172.20.20.0/24 [90/1280] via 172.16.200.200, 02:19:26, FastEthernet4.200
D      172.20.2.0/23 [90/1024] via 172.16.200.200, 00:03:38, FastEthernet4.200
D EX 192.168.1.0/24 [170/768] via 172.16.201.2, 00:37:11, FastEthernet4.201

```

Limiting Incoming Updates with a Distribute-List

Example 5-21 portrays the definition and usage of an incoming *distribute-list*, according to the scenario depicted on Figure 5-7. This mechanism enables a Cisco routing device (in this case, ASA1) to limit the routes installed in its routing table.

Example 5-21 EIGRP distribute-list Configuration

```

! Applying a Distribute-list in the inbound direction on ASA1 ("dmz" interface)

access-list EIGRP300-IN standard permit 172.20.0.0 255.255.0.0
!
router eigrp 300
distribute-list EIGRP300-IN in interface dmz
!
! EIGRP routes received by ASA1
ASA1# show route dmz | b Gateway
Gateway of last resort is not set
C      172.16.201.0 255.255.255.0 is directly connected, dmz
D      172.20.20.0 255.255.255.0 [90/26880] via 172.16.201.201, 0:02:58, dmz
D      172.20.2.0 255.255.254.0 [90/26624] via 172.16.201.201, 0:02:58, dmz
!
ASA1# show access-list EIGRP300-IN
access-list EIGRP300-IN; 1 elements; name hash: 0x9cbbf8f8
access-list EIGRP300-IN line 1 standard permit 172.20.0.0 255.255.0.0 (hitcnt=6)
0x683e5368

```

EIGRP QUERY and REPLY Messages

In the event that an EIGRP speaker loses a route to a destination but has at least one pre-computed Feasible Successor (FS), it performs a *Local Computation*, selects the alternative path that has the best metric and moves it to the routing table. In situations in which it has no predetermined Feasible Successor, the route loss triggers a *Diffusing Computation* and the state of the destination is changed to *Active*.

As part of the diffusing computation, QUERY messages including the new best *vector metric* are sent to neighbors, and the correspondent REPLY messages are awaited. If the neighbor has an FS, it sends a REPLY, containing its minimum locally computed distance for the destination, back to the originating router. In case the inquired neighbor does not have an FS, it also changes the destination's state to Active and begins another diffusing computation.

Example 5-22 refers to the topology shown in Figure 5-8, in which R2 (172.16.200.200) is an FS for R3 toward destination 172.16.210.0/24. In this example, ASA1 loses its route to 172.16.210.0/24, and because it has no FS, it sends a QUERY to its neighbor, R3, which does not need to send a QUERY (because it has an FS) and performs only a local computation. It also sends a REPLY to ASA1's QUERY, reporting a distance RD = 26368, via R2. After receipt of the REPLY, ASA1 installs the new route.

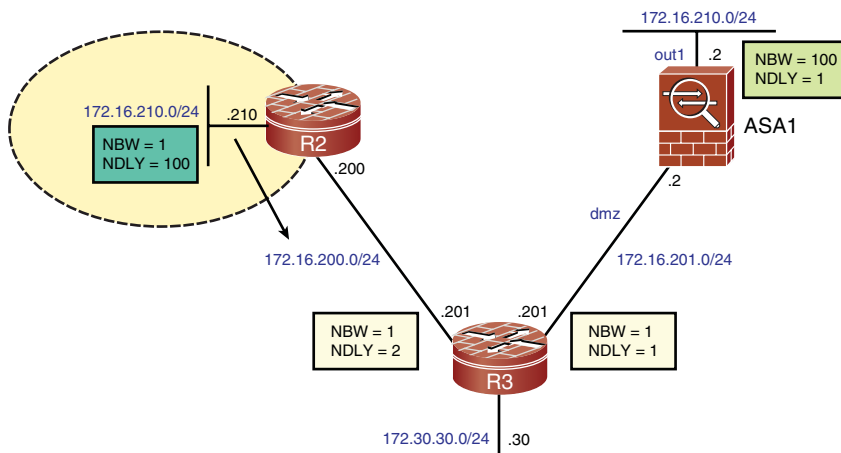


Figure 5-8 EIGRP QUERY and REPLY Messages

In the same topology, if R3 had lost its route to 172.30.30.0/24 (no FS to it), it would have started a diffusing computation, changing the destination to the *Active* state and sending QUERY messages to all its neighbors in search of a new path. It would then wait for the REPLY messages to come.

Example 5-22 EIGRP QUERY and REPLY Messages

```
! 172.16.200.200 (R2) is a Feasible Successor for R3 towards 172.16.210.0/24
R3# sh ip eigrp topology
IP-EIGRP Topology Table for AS(300)/ID(172.30.30.30)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status
P 172.16.210.0/24, 1 successors, FD is 26112
    via 172.16.201.2 (26112/25856), FastEthernet4.201
```

```

    via 172.16.200.200 (26368/25856), FastEthernet4.200
!
! Only the best path (in the Topology Table) is moved to the Routing Table
R3# show ip route eigrp
    172.16.0.0/24 is subnetted, 3 subnets
D    172.16.210.0 [90/26112] via 172.16.201.2, 00:19:55, FastEthernet4.201
    172.20.0.0/24 is subnetted, 1 subnets
D    172.20.20.0 [90/1280] via 172.16.200.200, 00:44:00, FastEthernet4.200
!
! ASA1 loses its route to 172.16.210.0/24 and places destination in Active state
DUAL: Dest 172.16.210.0 255.255.255.0 entering active state for topoid 0. DUAL: Set
reply-status table. Count is 1.
DUAL: Not doing split horizon
EIGRP: Sending QUERY on Vlan201 nbr 172.16.201.201 topoid 0
    AS 19661042, Flags 0x0, Seq 55/63 interfaceQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/1 serno 51-51
EIGRP: Received REPLY on Vlan201 nbr 172.16.201.201
    AS 19661058, Flags 0x0, Seq 64/55 interfaceQ 0/0 iidbQ un/rely 0/0 peerQ
un/rely 0/0
DUAL: dest(172.16.210.0 255.255.255.0) active
    [ output suppressed ]
DUAL: RT installed 172.16.210.0 255.255.255.0 via 172.16.201.201
!
! R3 has a Feasible Successor for 172.16.210.0/24 and performs only a Local
Computation
EIGRP: Received QUERY on FastEthernet4.201 nbr 172.16.201.2
    AS 300, Flags 0x0, Seq 55/63 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/0
DUAL: rcvquery: 172.16.210.0/24 via 172.16.201.2 metric 4294967295/4294967295, RD
is 26112
DUAL: Find FS for dest 172.16.210.0/24. FD is 26112, RD is 26112
DUAL:    172.16.201.2 metric 4294967295/4294967295
DUAL:    172.16.200.200 metric 26368/25856 found Dmin is 26368
DUAL: send REPLY(r1/n1) about 172.16.210.0/24 to 172.16.201.2
DUAL: RT installed 172.16.210.0/24 via 172.16.200.200

```

EIGRP Stub Operation

Example 5-22 briefly examined the use of EIGRP QUERY and REPLY messages over a simple network. In the associated topology (refer to Figure 5-8), a QUERY about a lost route for which R3 did not have a Feasible Successor (FS) would have been forwarded to all its neighbors.

If many other remote routers were added to this scenario to form a large *hub-and-spoke* EIGRP topology, a lost route for which R3 did not have an FS would have been forwarded to all the spokes by the hub, in search for an alternative path.

The diffusing computation rules state that a router should receive replies (within the *Active timer* of 3 minutes) from all the neighbors to which it sent QUERY messages, before being allowed to select the new best route. Certain adverse circumstances can lead EIGRP speakers to fail on their responses to queries; these include the following:

- Reload of a neighbor remains undetected
- Congestion or overload in the network links that connect them to neighbors
- Other types of events, such as hardware or software failures

When one of the previous events takes place, the originator of the diffusing computation is prevented from completing it. If the expected REPLY messages do not arrive before the *Active timer* expires, the EIGRP speaker declares the route *Stuck-in-Active (SIA)*, aborts the diffusing computation for the destination, and clears the adjacency with the nonrespondent neighbors. This loss of adjacency typically leads to more lost routes, additional QUERY messages, and probably to more SIA events in other points of the network.

The SIA condition was conceived as a built-in protection mechanism that establishes a maximum time a diffusing computation can take. Nevertheless, in poorly designed networks (mainly in large ones), the accumulation of SIA events can greatly reduce availability—a certainly undesired effect.

Many design components can come into play to reduce the so-called QUERY Boundaries and minimize the likelihood of network instabilities. These include the following:

- Careful address assignment to allow summarization
- Hierarchical and redundant topology
- Route filtering
- Default routes
- Stub routing

Among these resources, *stub routing* deserves special attention. It was not part of the original EIGRP specifications, but during the last years has proven effective, both as a scalability and stability feature, mainly on hub-and-spoke topologies.

Spoke routers (on remote sites) generally do not have many subnets to advertise and just need to know a few outgoing routes to connect to the rest of the internetwork. Actually, in many situations, such devices have only a default route for outbound traffic. The main motivation for configuring a routing protocol in the spokes relates to the convenience of detecting dynamically that a remote LAN on one of these sites has become unavailable.

The `eigrp stub` command enables stub routing which by default instructs the spoke to send only updates for connected and summary routes to its neighbors. The configuration of this mechanism also stops QUERY messages from being sent to the spokes. This last

feature is interesting because it prevents the spokes from being queried about networks that do not belong to their respective sites. Such a procedure avoids that the reload of a remote router (or loss of the circuit that connects it) come to the cause the SIA events previously discussed.

Example 5-23 shows the usage of the **receive-only** option for the **eigrp stub** command, which blocks the spoke from announcing any route to neighbors. Example 5-24 illustrates the usage of the **redistributed** option for the same command. (The other keywords that may be used with this command are connected, static, and summary). Both examples refer to Figure 5-7 and make clear that the QUERY messages are suppressed.

Example 5-23 EIGRP Stub (receive-only Mode)

```
! Configuring ASA1 to operate as a Stub in receive-only mode
router eigrp 300
 eigrp stub receive-only
!
! The hub , R3, views its neighbor ASA1 as a stub peer
R3#show ip eigrp neighbors detail
IP-EIGRP neighbors for process 300
H   Address                Interface      Hold Uptime    SRTT   RTO   Q   Seq
                               (sec)          (ms)          Cnt Num
1   172.16.201.2            Fa4.201       11 00:02:08    7     200   0   64
Version 0.8/3.0, Retrans: 0, Retries: 0
Receive-Only Peer Advertising ( No ) Routes
Suppressing queries
0   172.16.200.200          Fa4.200       14 01:38:29    3     200   0  120
Version 12.4/1.2, Retrans: 2, Retries: 0, Prefixes: 1
```

Example 5-24 EIGRP Stub Announces Redistributed Routes

```
! Configuring ASA1 to operate as a Stub and advertise only redistributed routes
router eigrp 300
 eigrp stub redistributed
!
! The hub , R3, views its neighbor ASA1 as a stub peer
R3# show ip eigrp neighbors detail
IP-EIGRP neighbors for process 300
H   Address                Interface      Hold Uptime    SRTT   RTO   Q   Seq
                               (sec)          (ms)          Cnt Num
1   172.16.201.2            Fa4.201       14 00:00:23    5     200   0   78
Version 0.8/3.0, Retrans: 0, Retries: 0, Prefixes: 1
Stub Peer Advertising ( REDISTRIBUTED ) Routes
Suppressing queries
0   172.16.200.200          Fa4.200       11 01:42:41    1     200   0  122
Version 12.4/1.2, Retrans: 2, Retries: 0, Prefixes: 1
```

OSPF Overview

Open Shortest Path First (OSPF) is an open standard Link State IGP that uses the Dijkstra's Shortest Path First (SPF) algorithm to build loop-free topologies. Drawing on the *Link State Advertisements* (LSA) they receive from neighbors, OSPF routers construct and maintain a Link State Database (LSDB). Basically, the LSDB describes the topology of the AS. The information contained in the LSDB serves as input to the SPF algorithm, which computes best paths and transfer them to the routing table.

Every link belonging to an OSPF domain has, by default, a cost *inversely proportional* to its bandwidth. To calculate the metric to reach a certain destination, OSPF adds all the costs of individual links in the path and selects the one that results in the lowest sum.

OSPF supports the grouping of networks inside an AS by means of structures called *areas*. Many interesting OSPF features are available because the SPF algorithm is executed separately for each area of a given AS:

- The topology of each area is not visible to the rest of the AS. This saves memory and CPU and protects the area from bad routing data, providing more stability and scalability.
- Link availability changes within each area are quickly detected, enabling for rapid recalculation of a new topology. This results in faster convergence, even for large networks.
- Filtering between areas enables the hiding of routing information that can be used in some cases as a security feature. Hierarchical network topologies are possible, bringing modularity to the AS. This helps with scalability considerations and facilitates maintenance and troubleshooting.

Note To profit from these resources, it is indispensable to invest some time and effort in good IP addressing design. This can enable more efficient filtering and summarization.

A 32-bit ID can identify each area. The ID 0.0.0.0 (or simply 0) is reserved for the *backbone area*, which is in charge of interconnecting the other areas in the AS. Nonbackbone (nonzero) areas make use of *Area 0* to exchange routing packets.

Note The previous section characterized that one key configuration requirement for the establishment of an EIGRP adjacency is that the AS numbers (process number) match on both ends of the link. Although Cisco routing devices enable the definition of an OSPF AS number, this parameter is not part of the RFCs. For an OSPF neighbor relationship to be set up, the main issues are to ensure that both sides reside in the same subnet and are configured with the same *Area ID*, MTU value, and OSPF network type.

According to the areas it connects to and other routing processes it interacts with (through route redistribution), an OSPF router receives a special classification as follows:

- **Internal routers:** Routers whose OSPF-enabled interfaces are all part of the same area. These routing devices have a single LSDB.
- **Area border routers (ABR):** Routers that connect to Area 0 (backbone) and to at least one nonzero area (regular area). Interarea traffic flows through ABRs, which maintain an LSDB for each area it attaches to. ABRs can summarize routes from regular areas toward the backbone, a procedure that help on scaling an OSPF internetwork.
- **Backbone routers:** Routers that have at least one interface residing in the backbone area.
- **Autonomous System Boundary Routers (ASBR):** Routers that redistribute information from other routing processes into OSPF. The external process may be another dynamic routing protocol (eventually, even a second OSPF process), static routes, or connected routes. Any of the first three classes of routers can become an ASBR. (It happens as soon it starts route redistribution.)

Note As per the previous categorization, an *ABR* is always a backbone router, but the converse statement is not necessarily true. A backbone router that is not an ABR falls under the definition of internal routers.

Figure 5-9 illustrates all classes of routers just discussed for a sample OSPF process. In this figure, R1, R2, R3, R5, R6, R7, and R10 are internal routers. The ABRs are R4, R8, and R9. All routers in the sequence R4 through R9 are backbone routers, and the only ASBR is R6.

Note OSPF defines a resource called *virtual link* that enables connectivity to Area 0 through a nonbackbone area. Although this is necessary in some environments, the use of this mechanism as a design practice should be avoided.

The OSPF specification on RFC 2328 (OSPF version 2) also defines five types of *Link State Advertisements* (LSA), each of them carrying a piece of information somehow related to the internetwork topology. The LSDB is formed by the collection of LSAs:

- **Router LSAs (Type 1 LSAs):** Describe the cost and states of the router's interfaces (links) in the area. Generated by all routers and flooded only within the area under consideration. The Link State ID (LS ID) for type 1 LSAs is the Router ID of the advertising router.

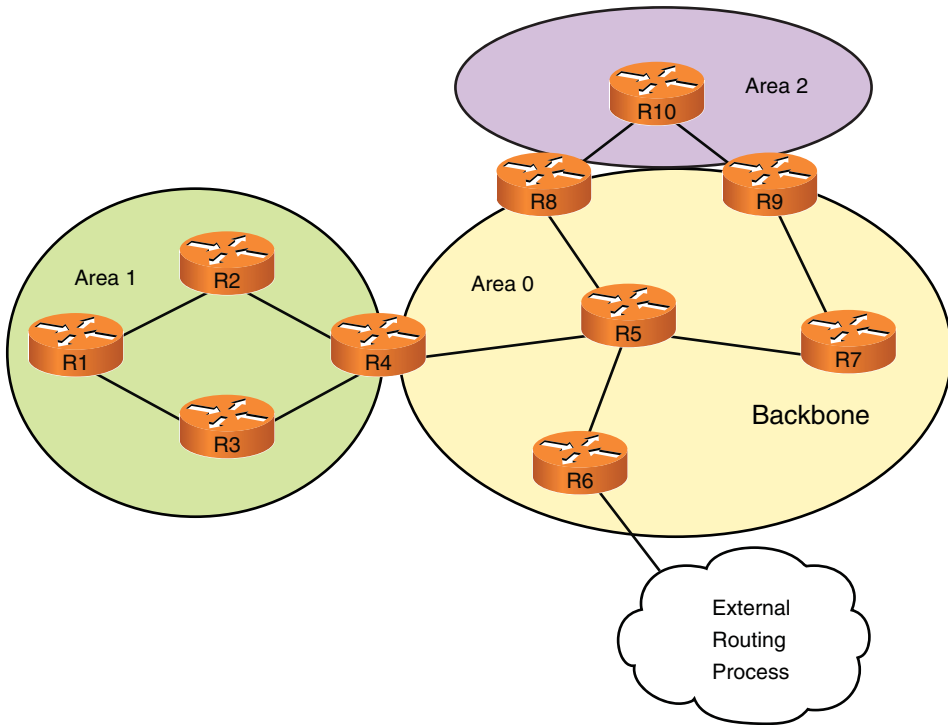


Figure 5-9 *OSPF Router Types*

- **Network LSAs (Type 2 LSAs):** Enumerate the routers connected to an OSPF Broadcast or Non Broadcast Multiple Access (NBMA) segment. This LSA type is generated only by the Designated Router (DR) for the given segment, and its flooding is limited to the area.
- **Summary Network LSAs (Type 3 LSAs):** Used to describe interarea routes for an OSPF process. Generated by ABRs.
- **Summary ASBR LSAs (Type 4 LSAs):** Used to describe routes to reach ASBRs. Also generated by ABRs.
- **AS-external LSAs (Type 5 LSAs):** Describe routes injected into the OSPF process by redistribution. This type of LSA is generated by ASBRs and flooded throughout the AS.

Configuring and Monitoring OSPF

The series of practical examples that follows can certainly help reinforce the main OSPF concepts discussed so far and introduces some other relevant terminology.

OSPF Configuration Fundamentals

Example 5-25 documents the baseline set of commands for enabling OSPF on a practical scenario as the one illustrated in Figure 5-10. OSPF does not mandate the configuration of the same process number (AS number) for an adjacency to be established. Nonetheless, it is good administrative practice to keep numbers consistent across the routing devices.

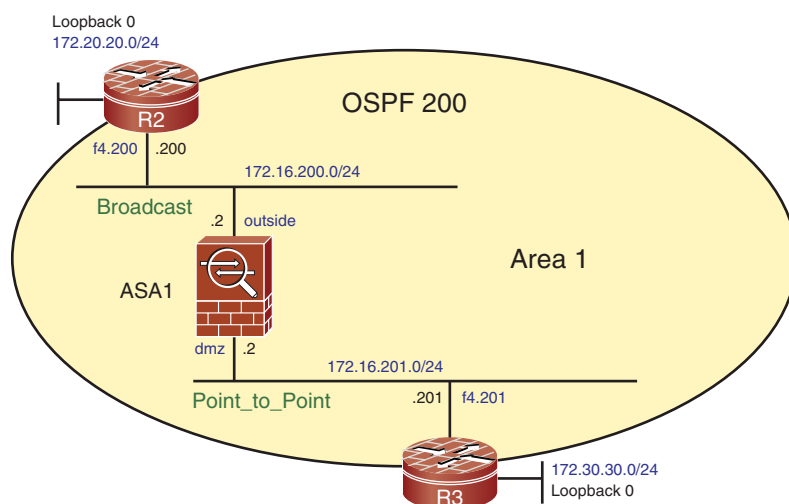


Figure 5-10 *Reference Topology for the Analysis of OSPF Operation*

The first application of a **network** statement is to tell the OSPF process that the particular network may be used for establishing an adjacency over the configured area and for subsequent exchange of LSAs. In the example the adjacency between R3 and ASA1 is built over the 172.16.201.0/24 link and belongs to area 1.

The second usage of a **network** statement is to instruct the OSPF process to advertise a specific segment over the other OSPF-enabled interfaces. Network 172.20.20.0/24, for instance, is connected to R2 as a *stub* segment (no other routers attached to it) and is announced to ASA1 over the 172.16.200.0/24 segment.

When used within an OSPF process, the **network** command is always accompanied by an area number. The configuration of identical area numbers on the intended OSPF neighbors is a must.

Similarly to what was said about EIGRP, Cisco routers running OSPF use a *wildcard mask* when defining a prefix with the **network** command. The ASA family, on the contrary, uses a regular *network mask*.

The Router ID, an IP address sent in *Hello packets* when neighbor negotiation begins, is used to univocally identify an OSPF speaker within a domain.

Example 5-25 *Baseline OSPF Configuration for Single Area Scenario*

```

! R2 baseline OSPF Configuration (single area)
router ospf 200
  router-id 172.20.20.20
  network 172.16.200.0 0.0.0.255 area 1
  network 172.20.20.0 0.0.0.255 area 1
!
! ASA1 baseline OSPF Configuration (single area)
router ospf 200
  router-id 172.16.200.2
  network 172.16.200.0 255.255.255.0 area 1
  network 172.16.201.0 255.255.255.0 area 1
!
! R3 baseline OSPF Configuration (single area)
router ospf 200
  router-id 172.30.30.30
  network 172.16.201.0 0.0.0.255 area 1
  network 172.30.30.0 0.0.0.255 area 1

```

OSPF, being a Layer 3 protocol, can be carried over several types of data link options. According to the Layer 2 technology over which it is being transported (Ethernet, Token Ring, Frame Relay, ATM, and so on), OSPF presents some operational differences for the following topics:

- **Neighbor discovery and maintenance:** The Hello protocol in charge of these tasks behaves slightly differently, depending on the L2 technology.
- **Database synchronization:** How does the reliable flooding process adapt to take advantage of special characteristics of the L2 medium? Which routers become adjacent and how do they synchronize their LSDBs over this L2 network?
- **Logical Representation:** How is router connectivity over this Data Link option represented in the LSDB? Is it possible to create some sort of abstraction to describe L2 technologies that behave in a similar fashion?

The OSPF specification groups the various L2 technologies into four logical *network types*:

- **Point-to-point network:** Although this network type is often associated with serial links, Cisco routing devices enable the configuration of other L2 to behave as if they had the point-to-point nature, thus connecting a single pair of OSPF speakers.
- **Broadcast network:** This Data Link option has intrinsic broadcast capability and might include several OSPF speakers simultaneously. To avoid the need of full mesh LSDB synchronization, OSPF defines a Designated Router (DR) and an associated Backup DR (BDR). These routers register themselves to the reserved multicast group

224.0.0.6 (AllDRouters) and listen to LSA updates from regular speakers. The DR forwards the received update to the reserved group 224.0.0.5 (AllSPFRouters). Classic representatives of this network type are Ethernet, Token Ring, and FDDI.

- **Non Broadcast Multiple Access (NBMA) network:** These L2 technologies enable the simultaneous attachment of more than two speakers but do not offer an inherent broadcast capability. DR and BDR are also elected in this model, which uses unicast OSPF packets, and has ATM, Frame Relay, and X.25 as classic examples.
- **Point-to-multipoint network:** This is just a particular case of the NBMA class in which the multipoint nature derives from the grouping of individual point-to-point links. In this model, the OSPF packets are multicast, and there is no DR/BDR election.

Although Cisco routers support all OSPF network types, the ASA family supports only the first two. This is not a problem because dedicated firewalls typically have only Ethernet-based physical interfaces.

If only two routing devices connect through an Ethernet-based interface, a point-to-point adjacency might sound inviting. It provides a way for a router to accept just one neighbor over a segment that natively uses broadcast.

Examples 5-26 and 5-27 were built using the scenario of Figure 5-10 and show, respectively, some characteristics of broadcast and point-to-point OSPF network types. Setting the **ospf priority** to zero excludes the routing device from the DR/BDR election and results in a state named *DROTHER*.

Example 5-26 OSPF Adjacency over a Broadcast Network

```
! 'ospf priority 0' excludes ASA1 from DR/BDR election on this broadcast segment
interface Vlan200
  nameif outside
  security-level 0
  ip address 172.16.200.2 255.255.255.0
  ospf cost 1
  ospf priority 0
!
! OSPF adjacency between ASA1 and R2 (Broadcast network)
ASA1# show ospf interface outside
outside is up, line protocol is up
  Internet Address 172.16.200.2 mask 255.255.255.0, Area 1
  Process ID 200, Router ID 172.16.200.2, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DROTHER, Priority 0
  Designated Router (ID) 172.20.20.20, Interface address 172.16.200.200
  No backup designated router on this network
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
[ output suppressed ]
  Neighbor Count is 1, Adjacent neighbor count is 1
  Adjacent with neighbor 172.20.20.20 (Designated Router)
```

```

Suppress hello for 0 neighbor(s)
!
! OSPF adjacency between R2 and ASA1 (Broadcast network)
R2# show ip ospf interface f4.200
FastEthernet4.200 is up, line protocol is up
  Internet Address 172.16.200.200/24, Area 1
  Process ID 200, Router ID 172.20.20.20, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 172.20.20.20, Interface address 172.16.200.200
  No backup designated router on this network
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
[output suppressed]
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 172.16.200.2
  Suppress hello for 0 neighbor(s)
!
! DR and BDR on broadcast/NBMA networks join groups 224.0.0.5 and 224.0.0.6
R2# show ip interface f4.200 | include Multicast
  Multicast reserved groups joined: 224.0.0.5 224.0.0.6

```

Example 5-27 OSPF Adjacency over a Point-to-Point Network

```

! R3 establishes an adjacency with ASA1 on 172.16.201.0/24 network (area 1)
interface FastEthernet4.201
  encapsulation dot1Q 201
  ip address 172.16.201.201 255.255.255.0
  ip ospf network point-to-point
!
! Point-to-point on ASA1 interface, requires 'neighbor' command under OSPF process
interface Vlan201
  nameif dmz
  security-level 50
  ip address 172.16.201.2 255.255.255.0
  ospf cost 1
  ospf network point-to-point non-broadcast
!
router ospf 200
  neighbor 172.16.201.201 interface dmz
!
! Point-to-point adjacency between ASA1 and R3
ASA1# show ospf interface dmz
dmz is up, line protocol is up
  Internet Address 172.16.201.2 mask 255.255.255.0, Area 1
  Process ID 200, Router ID 172.16.200.2, Network Type POINT_TO_POINT, Cost: 1

```



```

Transmit Delay is 1 sec, State POINT_TO_POINT,
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
[output suppressed]
Neighbor Count is 1, Adjacent neighbor count is 1
Adjacent with neighbor 172.30.30.30
Suppress hello for 0 neighbor(s)
!
! Only the all OSPF-routers group (224.0.0.5) is joined (Point-to-Point Adjacency)
R3# show ip interface f4.201 | include Multicast
Multicast reserved groups joined: 224.0.0.5
!
ASA1# show ospf neighbor
Neighbor ID      Pri   State           Dead Time   Address        Interface
172.20.20.20     1    FULL/DR         0:00:30     172.16.200.200 outside
172.30.30.30     1    FULL/-          0:00:39     172.16.201.201 dmz
!
ASA1# show ospf neighbor 172.30.30.30 detail
Neighbor 172.30.30.30, interface address 172.16.201.201
In the area 1 via interface dmz
Neighbor priority is 1, State is FULL, 6 state changes
DR is 0.0.0.0 BDR is 0.0.0.0
Options is 0x52
[output suppressed]

```

Example 5-28 displays the routes learned via OSPF for R3 and ASA1 on the topology of Figure 5-10. It also shows the LSDB for ASA1, which contains LSAs of Types 1 and 2.

Example 5-29 details a sample Router LSA advertised by R3 (Adv Router 172.30.30.30), as seen on ASA1. R3 uses two links to describe the point-to-point connection to ASA1:

- The first is for its interface address and the router ID of its neighbor ASA1.
- The second is for the network/mask pair (stub network 172.16.201.0/24).

Example 5-29 also shows how R2 describes its connection to a broadcast segment:

- The first link information describes the DR address and the interface address of R2 in this segment.
- The other relates to the network/mask information (stub network 172.16.200.0/24).

Example 5-28 OSPF Routing Table and Link State Database (LSDB)

```

! Routes learned via OSPF on R3
R3# show ip route ospf
172.16.0.0/24 is subnetted, 2 subnets

```

```

O      172.16.200.0 [110/2] via 172.16.201.2, 00:33:12, FastEthernet4.201
      172.20.0.0/24 is subnetted, 1 subnets
O      172.20.20.0 [110/3] via 172.16.201.2, 00:33:12, FastEthernet4.201
!
ASA1# show route outside | begin Gateway
Gateway of last resort is not set
C      172.16.200.0 255.255.255.0 is directly connected, outside
O      172.20.20.0 255.255.255.0 [110/2] via 172.16.200.200, 0:04:51, outside
!
ASA1# show route dmz | begin Gateway
Gateway of last resort is not set
C      172.16.201.0 255.255.255.0 is directly connected, dmz
O      172.30.30.0 255.255.255.0 [110/2] via 172.16.201.201, 0:04:56, dmz
!
! OSPF Database on ASA1
ASA1# show ospf database
      OSPF Router with ID (172.16.200.2) (Process ID 200)
          Router Link States (Area 1)
Link ID          ADV Router      Age           Seq#           Checksum Link count
172.16.200.2     172.16.200.2     1588          0x80000023    0x1adb 3
172.20.20.20     172.20.20.20     245           0x80000005    0xbb88 3
172.30.30.30     172.30.30.30     1588          0x80000002    0x7249 3

          Net Link States (Area 1)
Link ID          ADV Router      Age           Seq#           Checksum
172.16.200.200   172.20.20.20     748           0x80000002    0x3958

```

Example 5-29 Sample Router LSAs (Type 1)

```

! Type 1 LSA advertised by R3 (as seen on ASA1)

ASA1# show ospf database router 172.30.30.30
      OSPF Router with ID (172.16.200.2) (Process ID 200)
          Router Link States (Area 1)
LS age: 1654
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 172.30.30.30
Advertising Router: 172.30.30.30
LS Seq Number: 80000002
Checksum: 0x7249
Length: 60
Number of Links: 3

```

Link connected to: a Stub Network

(Link ID) Network/subnet number: 172.30.30.0

(Link Data) Network Mask: 255.255.255.0

Number of TOS metrics: 0

TOS 0 Metrics: 1

Link connected to: another Router (point-to-point)(Link ID) **Neighboring Router ID: 172.16.200.2**

(Link Data) Router Interface address: 172.16.201.201

Number of TOS metrics: 0

TOS 0 Metrics: 1

Link connected to: a Stub Network(Link ID) Network/subnet number: **172.16.201.0**(Link Data) Network Mask: **255.255.255.0**

Number of TOS metrics: 0

TOS 0 Metrics: 1

!

! Type 1 LSA advertised by R2 (as seen on ASA1)**ASA1# show ospf database router 172.20.20.20****[output suppressed]****Link connected to: a Transit Network**

(Link ID) Designated Router address: 172.16.200.200

(Link Data) Router Interface address: 172.16.200.200

Number of TOS metrics: 0

TOS 0 Metrics: 1

Link connected to: a Stub Network

(Link ID) Network/subnet number: 172.16.200.0

(Link Data) Network Mask: 255.255.255.0

Number of TOS metrics: 0

TOS 0 Metrics: 1

Example 5-30 details a sample Network LSA, which is generated by the DR (R2) and used to enumerate the routers attached to the broadcast segment (172.16.200.0/24). The reference to the attached routers uses the Router ID (not the interface IP).

Example 5-30 Sample Network LSA (Type 2)**ASA1# show ospf database network 172.16.200.200**

OSPF Router with ID (172.16.200.2) (Process ID 200)

Net Link States (Area 1)

Routing Bit Set on this LSA

```

LS age: 922
Options: (No TOS-capability, DC)
LS Type: Network Links
Link State ID: 172.16.200.200 (address of Designated Router)
Advertising Router: 172.20.20.20
LS Seq Number: 80000002
Checksum: 0x3958
Length: 32
Network Mask: 255.255.255.0
Attached Router: 172.20.20.20
Attached Router: 172.16.200.2

```

OSPF Scenario with Two Areas

Example 5-31 highlights the changes made to the baseline configuration of Example 5-25 to create a scenario with two areas corresponding to Figure 5-11. ASA1 is the ABR in this configuration and has one LSDB for each area (Example 5-32).

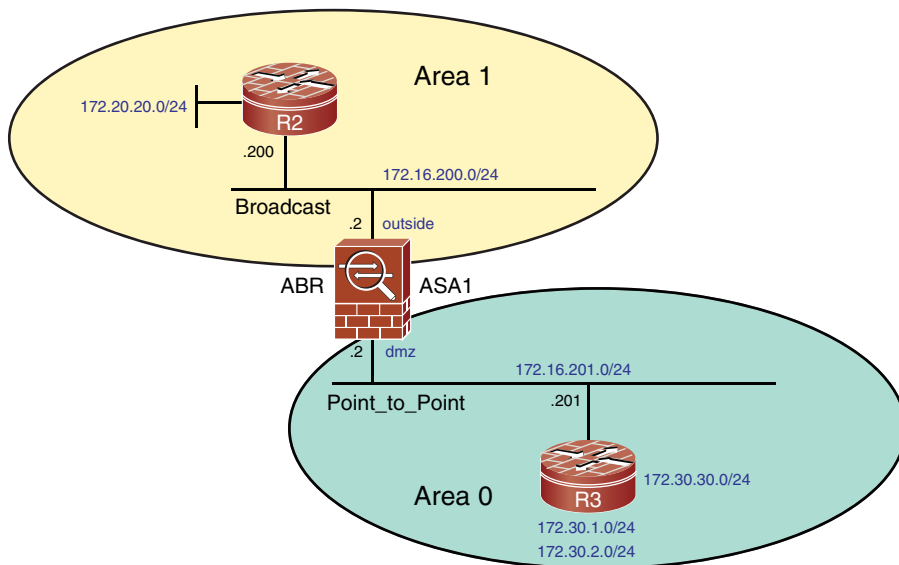


Figure 5-11 *OSPF Scenario with Two Areas*

Example 5-32 also shows that the internal Routers R2 and R3 see only the LSDB specific to the area to which they are attached.

The discussions presented so far have dealt with routing tables that contain routes to networks. OSPF includes a second routing table that specifies border routers reachability (ABRs and ASBRs). This special table displays with the `show ip ospf border router`

command on Cisco routers. In Example 5-32, R3 knows that ASA1 is the ABR and the path to it.

Example 5-31 *Moving R3-ASA1 Adjacency to Area 0*

```
! R3 (Moving the link 172.16.201.0/24 to area 0)
router ospf 200
  network 172.16.201.0 0.0.0.255 area 0
  network 172.30.30.0 0.0.0.255 area 0
!
! ASA1 (Moving the link 172.16.201.0/24 to area 0)
router ospf 200
  network 172.16.200.0 255.255.255.0 area 1
  network 172.16.201.0 255.255.255.0 area 0
  neighbor 172.16.201.201 interface dmz
!
! ASA1 becomes an Area Border Router (ABR) in this scenario
ASA1# show ospf 200
Routing Process "ospf 200" with ID 172.16.200.2 and Domain ID 0.0.0.200
Supports only single TOS(TOS0) routes
Does not support opaque LSA
It is an area border router
SPF schedule delay 5 secs, Hold time between two SPFs 10 secs
[output suppressed]
Number of areas in this router is 2. 2 normal 0 stub 0 nssa
External flood list length 0
  Area BACKBONE(0)
    Number of interfaces in this area is 1
    Area has no authentication
  [output suppressed]
  Area 1
    Number of interfaces in this area is 1
    Area has no authentication
  [output suppressed]
```

Example 5-32 *LSDB and Routing Table on Multiple Areas Scenario*

```
! ASA1 (ABR) has one LSDB for each Area
ASA1# show ospf database
  OSPF Router with ID (172.16.200.2) (Process ID 200)
    Router Link States (Area 0)
    Link ID      ADV Router      Age          Seq#           Checksum Link count
    172.16.200.2  172.16.200.2    395          0x80000002    0x22d9 2
    172.30.30.30  172.30.30.30    395          0x80000003    0x704a 3
```

```

Summary Net Link States (Area 0)
Link ID      ADV Router    Age      Seq#        Checksum
172.16.200.0 172.16.200.2   401      0x80000002 0x4fde
172.20.20.0   172.16.200.2   406      0x80000001 0xeeef

Router Link States (Area 1)
Link ID      ADV Router    Age      Seq#        Checksum Link count
172.16.200.2 172.16.200.2   405      0x80000026 0xe83f 1
172.20.20.20 172.20.20.20    72      0x80000006 0xb989 3
172.30.30.30 172.30.30.30   627      0x80000004 0xa60c 2

Net Link States (Area 1)
Link ID      ADV Router    Age      Seq#        Checksum
172.16.200.200 172.20.20.20  566      0x80000003 0x3759

Summary Net Link States (Area 1)
Link ID      ADV Router    Age      Seq#        Checksum
172.16.201.0 172.16.200.2   397      0x80000001 0x46e7
172.30.30.0   172.16.200.2   387      0x80000001 0x 8c2
172.30.30.0   172.30.30.30   624      0x80000001 0x8fbc
!
! R3 (Internal Router) sees only Area 0
R3# show ip ospf database
      OSPF Router with ID (172.30.30.30) (Process ID 200)
Router Link States (Area 0)
Link ID      ADV Router    Age      Seq#        Checksum Link count
172.16.200.2 172.16.200.2   709      0x80000002 0x0022D9 2
172.30.30.30 172.30.30.30   708      0x80000003 0x00704A 3

Summary Net Link States (Area 0)
Link ID      ADV Router    Age      Seq#        Checksum
172.16.200.0 172.16.200.2   714      0x80000002 0x004FDE
172.20.20.0   172.16.200.2   719      0x80000001 0x00EEEE
!
! Inter-area Routes (IA) derived from Summary Net Link States
R3# show ip route | begin Gateway
Gateway of last resort is not set
172.16.0.0/24 is subnetted, 2 subnets
O IA 172.16.200.0 [110/2] via 172.16.201.2, 00:10:00, FastEthernet4.201
C    172.16.201.0 is directly connected, FastEthernet4.201
172.21.0.0/24 is subnetted, 1 subnets
C    172.21.21.0 is directly connected, Vlan21
172.20.0.0/24 is subnetted, 1 subnets
O IA 172.20.20.0 [110/3] via 172.16.201.2, 00:10:00, FastEthernet4.201

```

```

172.30.0.0/24 is subnetted, 1 subnets
C    172.30.30.0 is directly connected, Loopback0
!
! R3 knows that ASA1 is an ABR reachable through an intra-area route
R3# show ip ospf border-routers
OSPF Process 200 internal Routing Table
Codes: i - Intra-area route, I - Inter-area route
i 172.16.200.2 [1] via 172.16.201.2, FastEthernet4.201, ABR, Area 0, SPF 5

```

Example 5-33 displays the information available on a summary-net LSA, which describes an interarea route. A Type 3 LSA tells a router in one area about subnets residing in different areas but does not report the connectivity details of the other area. These details are part of Type 1 LSAs, which have scope restricted to a single area.

Example 5-33 Sample Summary-Net LSA (Type 3)

```

! Summary LSAs for R3 in the topology of Figure 5-11
R3# show ip ospf database summary 172.20.20.0
      OSPF Router with ID (172.30.30.30) (Process ID 200)
      Summary Net Link States (Area 0)
Routing Bit Set on this LSA
LS age: 794
Options: (No TOS-capability, DC, Upward)
LS Type: Summary Links(Network)
Link State ID: 172.20.20.0 (summary Network Number)
Advertising Router: 172.16.200.2
LS Seq Number: 80000001
Checksum: 0xEEEF
Length: 28
Network Mask: /24
      TOS: 0 Metric: 2

```

Example 5-34 presents a resource that enables the summarization of routing information between areas. The **area range** command, when configured with the **not-advertise** option, completely eliminates the propagation of networks that fall in the range to other areas. Traffic between areas can be further controlled (independently of the creation of *area ranges*) with the *LSA Type-3 Filtering* feature, as shown in Example 5-35. This filtering functionality may be thought of as a security measure in some scenarios. Alternatively, it might reduce interarea flows in situations in which good IP Addressing is not in place.

Example 5-34 *Area Range*

```

! Announcing two new /24 prefixes on R3 (Figure 5-11)
router ospf 200
  network 172.30.1.0 0.0.0.255 area 0
  network 172.30.2.0 0.0.0.255 area 0
!
! New prefixes are visible on ASA1 as internal routes
ASA1# show route dmz | begin Gateway
Gateway of last resort is not set
C    172.16.201.0 255.255.255.0 is directly connected, dmz
O    172.30.30.0 255.255.255.0 [110/2] via 172.16.201.201, 0:02:37, dmz
O    172.30.2.0 255.255.255.0 [110/2] via 172.16.201.201, 0:02:37, dmz
O    172.30.1.0 255.255.255.0 [110/2] via 172.16.201.201, 0:02:37, dmz
!
! New prefixes are visible on R2 as inter-area routes
R2# show ip route ospf
    172.16.0.0/24 is subnetted, 2 subnets
O IA   172.16.201.0 [110/2] via 172.16.200.2, 11:56:01, FastEthernet4.200
    172.30.0.0/24 is subnetted, 3 subnets
O IA   172.30.30.0 [110/3] via 172.16.200.2, 11:55:51, FastEthernet4.200
O IA   172.30.2.0 [110/3] via 172.16.200.2, 00:05:24, FastEthernet4.200
O IA   172.30.1.0 [110/3] via 172.16.200.2, 00:05:34, FastEthernet4.200
!
! Defining and Area Range on ASA1 (ABR)
router ospf 200
  area 0 range 172.30.0.0 255.255.0.0
!
ASA1# show route | i summary
O    172.30.0.0 255.255.0.0 is a summary, 0:04:37
!
ASA1# show ospf 200
[ output suppressed ]
    Area BACKBONE(0)
        Number of interfaces in this area is 1
        Area has no authentication
        SPF algorithm executed 10 times
        Area ranges are
            172.30.0.0 mask 255.255.0.0 Active(2) Advertise
!
! The area-range appears on R2. Component Routes are removed
R2# show ip route ospf
    172.16.0.0/24 is subnetted, 2 subnets
O IA   172.16.201.0 [110/2] via 172.16.200.2, 12:06:55, FastEthernet4.200
O IA 172.30.0.0/16 [110/3] via 172.16.200.2, 00:05:36, FastEthernet4.200
!

```


! Defining an area-range on ASA1 with the not-advertise option

```
router ospf 200
```

```
area 0 range 172.30.0.0 255.255.0.0 not-advertise
```

```
!
```

```
ASA1# show ospf 200
```

```
[ output suppressed ]
```

```
Area BACKBONE(0)
```

```
Number of interfaces in this area is 1
```

```
Area has no authentication
```

```
SPF algorithm executed 9 times
```

```
Area ranges are
```

```
172.30.0.0 mask 255.255.0.0 Passive DoNotAdvertise
```

Example 5-35 LSA Type-3 Filtering

! Defining an additional prefix on R3 (Figure 5-11)

```
router ospf 200
```

```
network 172.31.31.0 0.0.0.255 area 0
```

```
!
```

! New prefix becomes visible on R2 LSDB as a Type-3 LSA

```
R2# show ip ospf database | begin Summary
```

```
Summary Net Link States (Area 1)
```

Link ID	ADV Router	Age	Seq#	Checksum
172.16.201.0	172.16.200.2	879	0x80000001	0x0046E7
172.30.1.0	172.16.200.2	160	0x80000001	0x00489F
172.30.2.0	172.16.200.2	1059	0x80000001	0x003DA9
172.30.30.0	172.16.200.2	879	0x80000001	0x0008C2
172.31.31.0	172.16.200.2	160	0x80000001	0x00F0D7

```
!
```

! Defining and enabling a prefix-list for LSA Type-3 Filtering on ASA1 (ABR)

```
prefix-list BACKBONE-OUT seq 10 deny 172.30.1.0/24 le 32
```

```
prefix-list BACKBONE-OUT seq 20 permit 172.30.0.0/16 le 24
```

```
prefix-list BACKBONE-OUT seq 30 permit 172.16.201.0/24
```

```
!
```

```
router ospf 200
```

```
area 0 filter-list prefix BACKBONE-OUT out
```

```
!
```

!ASA1 removes 172.31.31.0/24 from its flood list

```
OSPF: Send Type 3, LSID 172.31.31.0, Adv rtr 172.16.200.2, age 3600, seq 0x80000002 (0)
```

```
Create retrans unit 0xd8d7d340/0xd8d59120 1 (0/1) 1
```

```
OSPF: Set nbr 1 (0/1) retrans to 4776 count to 1
```

```
Set idb next flood info from d8d5a15c (203) to d8d5a2d8 (204)
```

```

OSPF: Remove Type 3 LSA ID 172.31.31.0 Adv rtr 172.16.200.2 Seq 80000002 from
outside flood list
[ output suppressed]
OSPF: Remove Type 3 LSA ID 172.31.31.0 Adv rtr 172.16.200.2 Seq 80000002 from
172.20.20.20 retransmission list
!
! R2 receives LSA with Age = 3600 (and removes it from LSDB)
OSPF: Rcv Update Type 3, LSID 172.30.1.0, Adv rtr 172.16.200.2, age 3600, seq
0x80000002 Mask /24
!
! Prefixes 172.31.31.0/24 and 172.30.1.0/24 are removed from R2's LSDB
R2# show ip ospf database | begin Summary
Summary Net Link States (Area 1)
Link ID      ADV Router   Age         Seq#         Checksum
172.16.201.0 172.16.200.2 88          0x80000002  0x0044E8
172.30.2.0    172.16.200.2 88          0x80000002  0x003BAA
172.30.30.0   172.16.200.2 88          0x80000002  0x0006C3

```

Figure 5-12 depicts the baseline topology for the analysis of OSPF Redistribution, where-
 as Example 5-36 registers the configuration that enables R3 to become an ASBR. The
 example illustrates the two options of metrics that can be assigned to *AS-External
 Routes*, namely Type 1 and Type 2. The control of redistributed routes with a *route-map*
 is always a recommended practice.

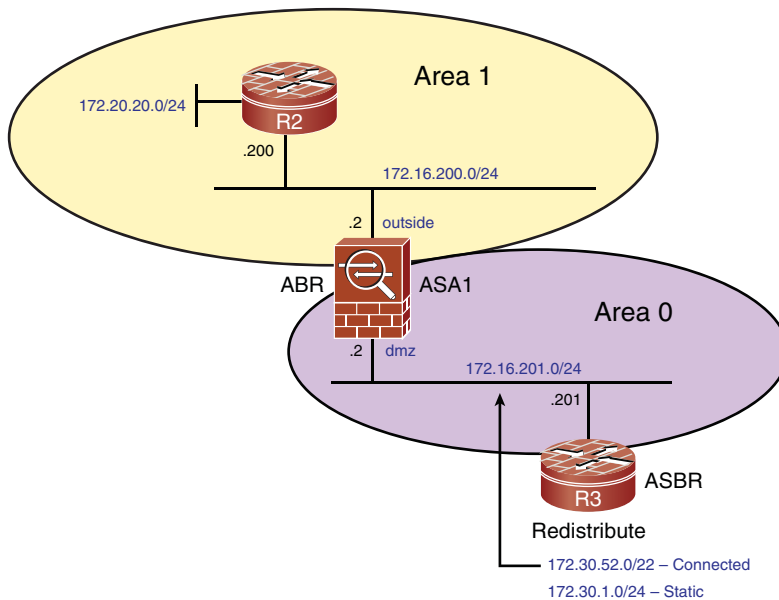


Figure 5-12 Reference Topology for OSPF Redistribution

Example 5-36 *Sample Configuration for Redistribution*

```

! Limiting the redistribution of connected routes to those defined by access-list 1
access-list 1 permit 172.30.52.0
!
route-map CONNECTED1 permit 10
  match ip address 1
!
! Limiting the redistribution of static routes to those defined by access-list 2
access-list 2 permit 172.30.1.0
!
route-map STATIC1 permit 10
  match ip address 2
!
! Enabling controlled redistribution of connected and static routes on R3
router ospf 200
  redistribute connected metric-type 1 subnets tag 3311 route-map CONNECTED1
  redistribute static metric-type 2 subnets tag 3322 route-map STATIC1
!
! R3 becomes an ASBR for process 200
R3# show ip ospf 200
  [ output suppressed ]
  It is an autonomous system boundary router
  Redistributing External Routes from,
    connected, includes subnets in redistribution
    static, includes subnets in redistribution
!
! R2 sees the two types of External OSPF Routes
R2# show ip route ospf
  172.16.0.0/24 is subnetted, 2 subnets
O IA   172.16.201.0 [110/2] via 172.16.200.2, 23:41:43, FastEthernet4.200
  172.30.0.0/16 is variably subnetted, 3 subnets, 2 masks
O E1   172.30.52.0/22 [110/22] via 172.16.200.2, 08:17:24, FastEthernet4.200
O IA   172.30.30.0/24 [110/3] via 172.16.200.2, 08:44:02, FastEthernet4.200
O E2   172.30.1.0/24 [110/20] via 172.16.200.2, 08:17:29, FastEthernet4.200

```

Example 5-37 documents the two metric types for OSPF *AS-External* routes. The cost of a Type 1 (E1) external path is the cost that the ASBR assigned to it during redistribution plus the cost to reach the ASBR. Type 2 (E2) is the default external metric type, and its cost does not take into account the cost to reach the ASBR.

Example 5-37 *Sample AS-External LSAs (Type-5)*

```

! External LSA with Type-1 Metric
R3# show ip ospf database external 172.30.52.0

```

```

        OSPF Router with ID (172.30.30.30) (Process ID 200)
          Type-5 AS External Link States

LS age: 524
Options: (No TOS-capability, DC)
LS Type: AS External Link
Link State ID: 172.30.52.0 (External Network Number )
Advertising Router: 172.30.30.30
LS Seq Number: 8000000F
Checksum: 0xED26
Length: 36
Network Mask: /22
Metric Type: 1 (Comparable directly to link state metric)
TOS: 0
Metric: 20
Forward Address: 0.0.0.0
External Route Tag: 3311
!
! External LSA with Type-2 Metric
R3# show ip ospf database external 172.30.1.0
        OSPF Router with ID (172.30.30.30) (Process ID 200)
          Type-5 AS External Link States

LS age: 516
Options: (No TOS-capability, DC)
LS Type: AS External Link
Link State ID: 172.30.1.0 (External Network Number )
Advertising Router: 172.30.30.30
LS Seq Number: 8000000F
Checksum: 0x7A3E
Length: 36
Network Mask: /24
Metric Type: 2 (Larger than any link state path)
TOS: 0
Metric: 20
Forward Address: 0.0.0.0
External Route Tag: 3322

```

Note The routes redistributed into OSPF can be aggregated through the **summary-address** command, within the **router ospf** configuration mode. This is different from RIP and EIGRP, in which summaries are defined under interface configuration (as shown in Examples 5-13 and 5-20).

Example 5-38 displays the routing table that informs about Border Routers' reachability. It also shows that a new type of LSA (Type 4) is now part of R2's LSDB. As stated earlier,

this kind of LSA describes routes to ASBRs. Example 5-39 further details the information contained in Summary-ASB LSAs.

Example 5-38 Information About OSPF Border Routers

```
! R2 knows that R3 is an ASBR and that it is reachable through an inter-area route
R2# show ip ospf border-routers
OSPF Process 200 internal Routing Table
Codes: i - Intra-area route, I - Inter-area route
I 172.30.30.30 [2] via 172.16.200.2, FastEthernet4.200, ASBR, Area 1, SPF 2
i 172.16.200.2 [1] via 172.16.200.2, FastEthernet4.200, ABR, Area 1, SPF 2
!
! R2 sees Summary-ASB LSA (Type-4) and External LSAs (Type-5)
R2# show ip ospf database
[ output suppressed ]
```

Summary ASB Link States (Area 1)				
Link ID	ADV Router	Age	Seq#	Checksum
172.30.30.30	172.16.200.2	1933	0x8000000F	0x00A6F7

Type-5 AS External Link States				
Link ID	ADV Router	Age	Seq#	Checksum Tag
172.30.1.0	172.30.30.30	1842	0x8000000F	0x007A3E 3322
172.30.52.0	172.30.30.30	1842	0x8000000F	0x00ED26 3311

Example 5-39 Sample Summary-ASB LSA (Type 4)

```
! R2 knows that R3 is an ASBR in the topology of Figure 5-12 by means of a
Type-4 LSA
R2# show ip ospf database asbr-summary
      OSPF Router with ID (172.20.20.20) (Process ID 200)
      Summary ASB Link States (Area 1)
Routing Bit Set on this LSA
LS age: 114
Options: (No TOS-capability, DC, Upward)
LS Type: Summary Links(AS Boundary Router)
Link State ID: 172.30.30.30 (AS Boundary Router address)
Advertising Router: 172.16.200.2
LS Seq Number: 80000010
Checksum: 0xA4F8
Length: 28
Network Mask: /0
      TOS: 0 Metric: 1
```

Note OSPF defines other types of areas such as *stub* and *not-so-stubby* areas (NSSA). This last one can be useful in some designs and is even assigned a special LSA (Type 7). The study of these special scenarios is beyond of the scope of this book..

Configuring Authentication for Routing Protocols

As seen throughout this chapter, the configuration of routing protocols is a key connectivity topic for any network. If properly designed, they bring flexibility, scalability, and the possibility of quickly adapting to topology changes. In the previous sections some mechanisms to limit the learned routes or hiding pieces of topology information were studied. Preventing routers from exchanging information with unknown or unauthorized neighbors (mainly on broadcast segments) is another profitable investment from an overall security policy standpoint.

This section devotes some attention to routing protocol authentication between neighbor routers that use any of the IGPs studied in this chapter. Although the routing protocols support a plain-text version of authentication, this book explores only the MD5 version. (This is the natural choice if security matters.)

Figure 5-13 displays the topology used for the examples in this section. The scenario depicts a hub-and-spoke network composed of a central ASA and remote IOS routers.

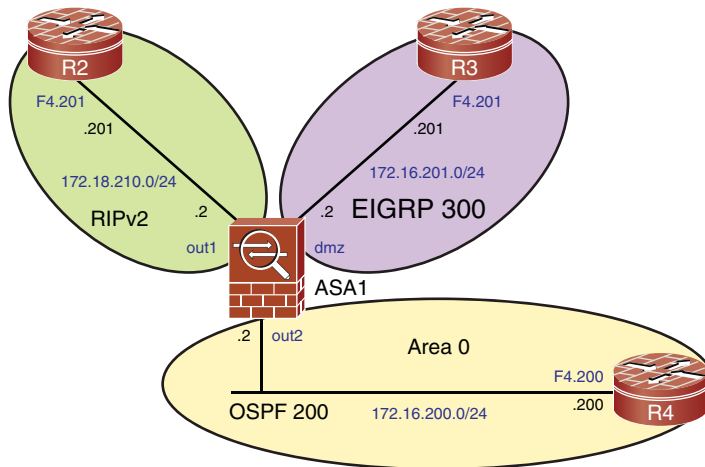


Figure 5-13 Topology for Illustrating Routing Protocol Authentication

Example 5-40 displays the necessary configuration for EIGRP MD5 authentication between ASA and IOS and shows how to verify authentication settings. Example 5-41 brings the equivalent information when the IGP is RIP. In a similar manner, Example 5-42 shows the required commands for configuration and verification of OSPF authentication.

You should notice that in all the scenarios the MD5 string must match on the two routing devices.

Note EIGRP and RIP configuration on IOS uses the **key chain** concept, enabling more flexibility in terms of key management. Multiple keys can be configured within the key chain, and each of them enables the specification of settings such as the *accept lifetime* for the key.

Example 5-40 Sample EIGRP Authentication Between ASA and IOS

```
! Relevant Configurations for EIGRP Authentication on ASA1
router eigrp 300
 network 172.16.201.0 255.255.255.0
!
interface Vlan201
 authentication key eigrp 300 ASA1-R3 key-id 1
 authentication mode eigrp 300 md5
!

! Relevant Configurations for EIGRP Authentication on R3
router eigrp 300
 network 172.16.201.0 0.0.0.255
!
key chain EIGRP300
 key 1
  key-string ASA1-R3
!
interface FastEthernet4.201
 ip authentication mode eigrp 300 md5
 ip authentication key-chain eigrp 300 EIGRP300
!

! Verifying EIGRP Authentication Settings
R3# show ip eigrp 300 interface detail f4.201 | include Authentication
Authentication mode is md5, key-chain is "EIGRP300"
!
R3# show key chain
Key-chain EIGRP300:
  key 1 — text "ASA1-R3"
    accept lifetime (always valid) - (always valid) [valid now]
    send lifetime (always valid) - (always valid) [valid now]
```

Example 5-41 Sample RIPv2 Authentication Between ASA and IOS

```
! Relevant Configurations for RIP Authentication on ASA1
router rip
```

```

version 2
network 172.18.0.0
!
interface Vlan210
  rip authentication mode md5
  rip authentication key ASA1-R2 key_id 1
!
! Relevant Configurations for RIP Authentication on R2
router rip
version 2
network 172.18.0.0
!
key chain RIP
  key 1
    key-string ASA1-R2
!
interface FastEthernet4.210
  ip rip authentication mode md5
  ip rip authentication key-chain RIP
!
! Verifying RIP Authentication settings
R2# show ip protocols | begin rip
Routing Protocol is "rip"
[ output suppressed ]
  Default version control: send version 2, receive version 2
    Interface          Send Recv Triggered RIP Key-chain
      FastEthernet4.210    2      2              RIP
!
R2# show key chain
Key-chain RIP:
  key 1 — text "ASA1-R2"
    accept lifetime (always valid) - (always valid) [valid now]
    send lifetime (always valid) - (always valid) [valid now]

```

Example 5-42 Sample OSPF Authentication Between ASA and IOS

```

! Relevant Configurations for OSPF Authentication on ASA1
router ospf 200
  area 0 authentication message-digest
  network 172.16.200.0 255.255.255.0 area 0
!
interface Vlan201
  ospf authentication message-digest
  ospf message-digest-key 1 md5 R2R3

```



```

!
! Relevant Configurations for OSPF Authentication on R4
router ospf 200
  area 0 authentication message-digest
  network 172.16.200.0 0.0.0.255 area 0
!
interface FastEthernet4.200
  ip ospf authentication message-digest
  ip ospf message-digest-key 1 md5 R2R3
!
! Verifying OSPF Authentication settings
R4# show ip ospf interface f4.200 | begin authentication
Message digest authentication enabled
  Youngest key id is 1
!
R4# show ip ospf 200 | include authentication|Area
Area BACKBONE(0)
  Area has message digest authentication

```

Bridged Operation

A bridge uses the MAC to interface associations in its MAC address table as the basis for forwarding of Layer 2 frames. When a frame is received on one of its interfaces, the bridge looks up the destination MAC address in its internal table. If the bridge finds a mapping between the destination MAC and a port (distinct from the one on which the frame arrived), it forwards the frame only to the specified port. If no mapping exists, the frame is flooded to all outbound ports. Multicast and broadcast packets are also flooded in this way. Figure 5-14 illustrates basic Bridged Operation.

The classic firewall design assumes its insertion in the network topology as a Layer 3 device; this option is certainly registered in the collective unconscious of the security community. Nevertheless, as a result of requiring minimal topology reconfiguration, the younger L2 connectivity option (*Bridge mode*) is appealing in many environments.

Other attractive usages of firewalls in Bridge mode (*transparent firewalls*) include the following:

- Simpler insertion of dedicated firewalls (ASA family) into multicast topologies. In this way, the ASA devices do not need to participate in multicast routing but are still capable of filtering multicast traffic.
- Simpler insertion of dedicated firewalls (ASA family) between routers that have built an IGP adjacency. With the L2 approach, the routing protocol packets can be filtered, and there is no need to use workarounds such as Generic Routing Encapsulation (GRE) tunnels to establish a connection through the ASA. This is equally true for First Hop Redundancy Protocols such as VRRP and HSRP.

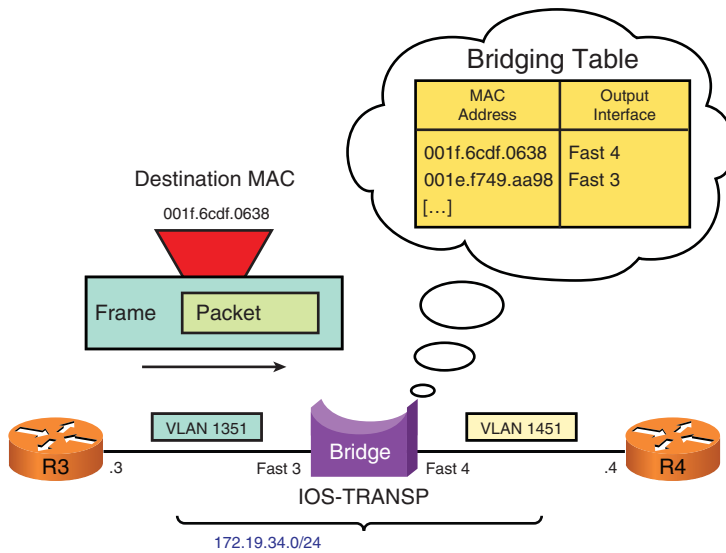


Figure 5-14 Topology for Analysis of IOS Transparent Bridging Operation

- Forwarding of nonIP traffic through ASA Firewalls becomes possible, which is not the case with the routed mode.

It is convenient to emphasize that a firewall operating as a L2 device will not lose its capabilities of filtering traffic passing through it. It will actually behave as a *conditional bridge*.

One important difference between IOS and ASA, when operating in Transparent mode is that IOS, by default, enables bridged traffic to pass through. To work as an L2 Firewall, IOS needs specific configuration to select the traffic to be blocked. ASA, with the exception of a few L2 multicast addresses (and the broadcast address FFFF.FFFF.FFFF), blocks everything, only enabling traffic explicitly defined via ACLs.

Note The examples in this chapter deal with the operation of Bridge mode as a connectivity option. The detailed filtering capabilities are covered in future chapters.

Figure 5-14 depicts the baseline topology for initial analysis of the Bridging operation in IOS. Examples 5-43 and 5-44 refer to this topology and illustrate how the Address Resolution Protocol (ARP) operates through the bridge.

Example 5-43 Baseline Configuration for IOS Transparent Bridging

```
! IOS-TRANSP is a 871 router (which has an embedded LAN switch)

vlan 1351
```

```

name R3-BG1
!
interface FastEthernet3
  switchport access vlan 1351
!
interface Vlan1351
  no ip address
  bridge-group 1
!
interface FastEthernet4
  no ip address
  bridge-group 1
!
bridge 1 protocol ieee

```

Example 5-44 ARP Resolution Through the IOS Transparent Bridge

! Initial situation for ARP

R3# show arp

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	172.19.34.3	-	001e.f749.aa98	ARPA	
FastEthernet0/0.1351					

!

R4# show arp

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	172.19.34.4	-	001f.6cdf.0638	ARPA	
FastEthernet0/0.1451					

!

! R4 starts a ping session to 172.19.34.3 (R3)

IP ARP: creating incomplete entry for IP address: 172.19.34.3 interface FastEthernet0/0.1451

IP ARP: sent req src 172.19.34.4 001f.6cdf.0638,
dst 172.19.34.3 0000.0000.0000 FastEthernet0/0.1451

IP ARP: rcvd rep src 172.19.34.3 001e.f749.aa98, dst 172.19.34.4 FastEthernet0/0.1451

!

! Resultant ARP table on R4

R4# show arp

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	172.19.34.3	0	001e.f749.aa98	ARPA	
FastEthernet0/0.1451					
Internet	172.19.34.4	-	001f.6cdf.0638	ARPA	
FastEthernet0/0.1451					

!

! ARP Resolution (and resultant table) on R3

```

IP ARP: rcvd req src 172.19.34.4 001f.6cdf.0638, dst 172.19.34.3
FastEthernet0/0.1351
IP ARP: creating entry for IP address: 172.19.34.4, hw: 001f.6cdf.0638
IP ARP: sent rep src 172.19.34.3 001e.f749.aa98,
          dst 172.19.34.4 001f.6cdf.0638 FastEthernet0/0.1351
!
R3# show arp

```

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	172.19.34.3	-	001e.f749.aa98	ARPA	FastEthernet0/0.1351
Internet	172.19.34.4	1	001f.6cdf.0638	ARPA	FastEthernet0/0.1351

IOS supports the simultaneous operation of routing and bridging. Among the different bridging options available in IOS, Integrated Routing and Bridging (IRB) deserves special mention. IRB uses a logical interface called Bridged Virtual Interface (BVI) to route between a bridge-group and a routed interface. The use of BVIs enables the interconnection of bridged interfaces to the remaining routed interfaces.

Figure 5-15 depicts a sample topology for the study of Integrated Routing and Bridging (IRB) on IOS. Example 5-45 refers to this figure and shows that VLANs 1351 and 1451 are bridged by IOS-TRANSP. (The Routers R3 are connected to the same IP Subnet and establish an EIGRP adjacency.)

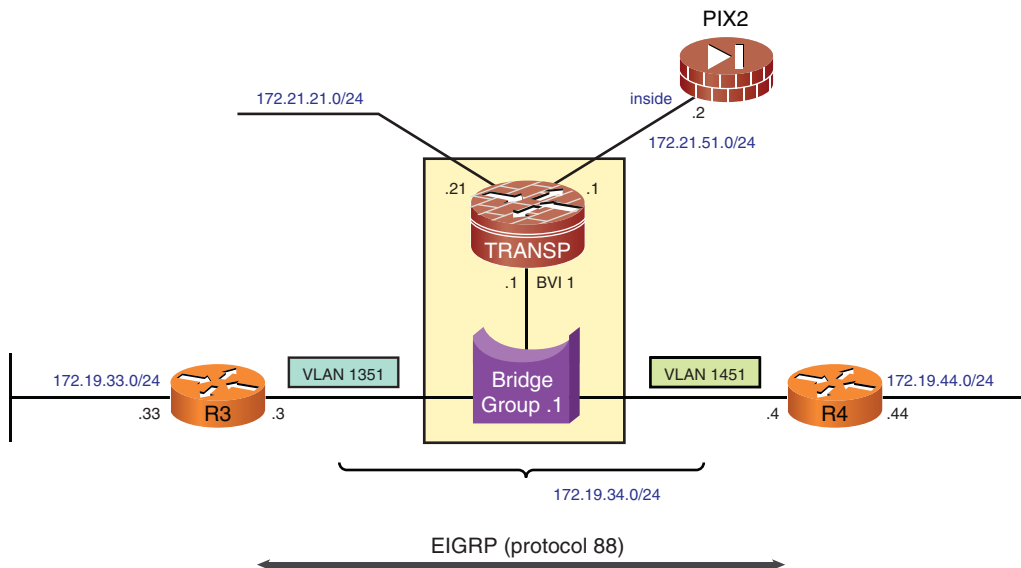


Figure 5-15 Integrated Routing and Bridging Scenario for IOS

The access to subnets of the 172.19.0.0/16 network is guaranteed via EIGRP, whereas addresses in the 172.21.0.0/16 range are reachable via static routes that use the BVI address as a gateway (172.19.34.1). Example 5-45 shows sample EIGRP packets exchanged by R3 and R4 and a ping from R4 to a destination outside Bridge-Group 1 (172.21.51.1).

Example 5-45 *Enabling Integrated Routing and Bridging (IRB) on IOS*

```

! Additional Configurations to enable IRB
bridge irb
bridge 1 route ip
!
interface BVI1
  ip address 172.19.34.1 255.255.255.0
!

! Routing Configuration on R4 (static route points to BVI address)
router eigrp 500
  network 172.19.34.0 0.0.0.255
  network 172.19.44.0 0.0.0.255
  no auto-summary
  eigrp router-id 172.19.44.44
!
ip route 172.21.0.0 255.255.0.0 172.19.34.1
!

! R3 and R4 exchange EIGRP (IP protocol 88) packets through the IOS Bridge
IP: s=172.19.34.4 (BVI1), d=224.0.0.10, len 60, input feature, proto=88, MCI
Check(64),
  rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
FIBipv4-packet-proc: route packet from BVI1 src 172.19.34.4 dst 224.0.0.10
FIBfwd-proc: Default:224.0.0.0/24 receive entry
FIBipv4-packet-proc: packet routing failed
IP: s=172.19.34.4 (BVI1), d=224.0.0.10, len 60, unroutable, proto=88

IP: s=172.19.34.3 (BVI1), d=224.0.0.10, len 60, input feature, proto=88, MCI
Check(64),
  rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
FIBipv4-packet-proc: route packet from BVI1 src 172.19.34.3 dst 224.0.0.10
FIBfwd-proc: Default:224.0.0.0/24 receive entry
FIBipv4-packet-proc: packet routing failed
IP: s=172.19.34.3 (BVI1), d=224.0.0.10, len 60, unroutable, proto=88
!

! R4 sends a ping to 172.21.51.1 (outside the BVI on IOS-TRANSP)
R4# ping 172.21.51.1 size 300 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
!
IP: s=172.19.34.4 (BVI1), d=172.21.51.1, len 300, input feature

```

```

    ICMP type=8, code=0, MCI Check(64), rtype 0, forus FALSE, sendself FALSE, mtu
    0, fwdchk FALSE
FIBipv4-packet-proc: route packet from BVI1 src 172.19.34.4 dst 172.21.51.1
FIBfwd-proc: Default:172.21.51.1/32 receive entry
FIBipv4-packet-proc: packet routing failed
IP: tableid=0, s=172.19.34.4 (BVI1), d=172.21.51.1 (FastEthernet4.1251), routed
via RIB
IP: s=172.19.34.4 (BVI1), d=172.21.51.1, len 300, rcvd 4
    ICMP type=8, code=0
IP: s=172.19.34.4 (BVI1), d=172.21.51.1, len 300, stop process pak for forus packet
    ICMP type=8, code=0
FIBipv4-packet-proc: route packet from (local) src 172.21.51.1 dst 172.19.34.4
FIBipv4-packet-proc: packet routing succeeded
IP: s=172.21.51.1 (local), d=172.19.34.4 (BVI1), len 300, sending
    ICMP type=0, code=0
IP: s=172.21.51.1 (local), d=172.19.34.4 (BVI1), len 300, sending full packet
    ICMP type=0, code=0

```

Figure 5-16 displays a basic topology for the discussion of ASA acting as a transparent firewall. When operating in Transparent mode, an ASA appliance must be configured with an IP address that belongs to the subnet to which it connects. This address is typically used for management purposes and for next-hop resolution (via ARP, for instance).

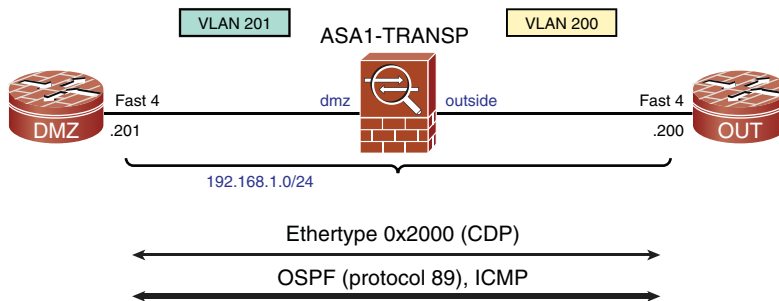


Figure 5-16 ASA operating as a Transparent Firewall

Example 5-46 directly relates to the topology of Figure 5-16 and shows not only the basic commands to enable Bridged mode but also how to verify that this is the mode of choice. It is important to remember that the IP address configured in this example (192.168.1.2/24) is shared by interfaces VLAN 200 (outside) and 201 (dmz).

Note In pre 8-4 releases ASA enables only one pair of bridged interfaces and, unlike IOS, does not support any routing protocols when configured as a transparent firewall.

Note The concepts of **bridge-groups** and BVI interfaces were introduced in ASA's release 8.4.1.

Example 5-46 *Basic Operation of ASA as a Transparent Firewall (Pre 8.4)*

```

firewall transparent
!
interface Vlan200
  nameif outside
  security-level 0
!
interface Vlan201
  nameif dmz
  security-level 100
!
ip address 192.168.1.2 255.255.255.0
!
! Verifying Operation
ASA1-TRANSP# show firewall
Firewall mode: Transparent
!
ASA1-TRANSP# show interface ip brief | include Vlan
Vlan200                192.168.1.2      YES unset  up
Vlan201                192.168.1.2      YES unset  up

```

As stated earlier in this section, ASA needs to be explicitly instructed to enable traffic through it when in Transparent mode. Example 5-47 shows the ACL configurations to permit the establishment of an OSPF adjacency between Routers DMZ and OUT in the scenario shown in Figure 5-16.

Example 5-48 refers to Figure 5-16 and illustrates the operation of Cisco Discovery Protocol (CDP) through the Transparent Firewall, ASA1-TRANSP. CDP is an L2 network discovery protocol that uses Ethertype 0x2000 and the multicast address 0100.0ccc.cccc.

Example 5-47 *Allowing OSPF Through the Transparent Firewall*

```

! Allowing OSPF between routers DMZ and OUT
access-list OUTSIDE extended permit ospf host 192.168.1.200 host 224.0.0.5
access-list OUTSIDE extended permit ospf host 192.168.1.200 host 224.0.0.6
access-list OUTSIDE extended permit ospf host 192.168.1.200 host 192.168.1.201
!
access-list DMZ extended permit ospf host 192.168.1.201 host 224.0.0.5
access-list DMZ extended permit ospf host 192.168.1.201 host 224.0.0.6
access-list DMZ extended permit ospf host 192.168.1.201 host 192.168.1.200

```

```

!
access-group OUTSIDE in interface outside
access-group DMZ in interface dmz
!
! Viewing OSPF connections
ASA1# show conn
6 in use, 6 most used
OSPF outside 192.168.1.200 dmz 192.168.1.201, idle 0:00:28, bytes 424
OSPF outside 224.0.0.5 dmz 192.168.1.201, idle 0:00:01, bytes 2972
OSPF outside 192.168.1.200 dmz 192.168.1.201, idle 0:00:18, bytes 432
OSPF outside 192.168.1.200 dmz 224.0.0.6, idle 0:00:03, bytes 396
OSPF outside 192.168.1.200 dmz 224.0.0.5, idle 0:00:05, bytes 2724

```

Example 5-48 *Allowing Cisco Discovery Protocol Through the Transparent Firewall*

```

! Ethertype ACLs to allow Cisco Discovery Protocol through ASA
access-list 200 ethertype permit 2000
access-list 201 ethertype permit 2000
!
access-group 200 in interface outside
access-group 201 in interface dmz
!
ASA1-TRANSP# show access-list 200
access-list 200; 1 elements
access-list 200 ethertype permit 2000 (hitcount=153)
!
! DMZ router sends a CDP packet to OUT router
CDP-PA: version 2 packet sent out on FastEthernet4
!
! OUT router receives a CDP packet from DMZ router
CDP-PA: Packet received from DMZ on interface FastEthernet4
**Entry found in cache**

```

Note The FWSM supports up to eight pairs of bridged interfaces per security context. Another interesting capability of the FWSM is that it simultaneously enables the creation of some contexts in Routed mode and others in Bridged mode. As of the writing of this book, ASA appliances require all the contexts to be of the same type. The specific behavior of FWSM is covered on Chapter 6, “Virtualization in the Firewall World.”

Summary

This chapter described how to insert Cisco Firewalls in the network topology, either in Routed (L3) or Bridged (L2) mode.

The main theoretical concepts and the associated configuration elements of IP routing were presented in a level of detail intended to give even unacquainted people with the subject, to get a good working knowledge of this foundational topic.

- Static routing is easy to implement but more difficult to maintain and troubleshoot, mainly in large networks. It might be combined with Dynamic Routing protocols in complex scenarios.
- RIP is classic distance vector Interior Gateway Protocol (IGP) typically suited for small networks. You can use If RIP, the recommendation is to choose version 2 (RIPv2), to avoid the limitations of the original version (no VLSM, no support to discontinuous subnets and no authentication, for instance).
- EIGRP, an IGP developed by Cisco, is a modern and effective option of distance vector protocol. It enables the usage of bandwidth and delay information for metric calculation (instead of relying merely on hop count) and permits the precomputing of alternative paths (through Feasible Successors), resulting in faster convergence. EIGRP supports features such as distribute-lists, route summarization, and stub routing, which, with good IP Addressing design, make it scalable and stable.
- OSPF is a standard Link State IGP that natively enables address hiding between areas and the creation of hierarchical topologies. Although distance vector protocols inform about routes on their updates, OSPF relies on Link State Advertisements (LSA) to get a description of the internetwork topology. With the information provided by each type of LSA, the SPF algorithm can run locally on each participating router. The best paths calculated by SPF are moved to the routing table.
- The operation of firewalls in Transparent mode might sound attractive in some network environments, mainly when addressing changes are undesirable and protocols other than IP need to pass through dedicated firewalls.
- A transparent firewall can be thought of as a *conditional bridge*, in which the security policies determine the types of traffic that have permission to be bridged between a given pair of interfaces.

Virtualization in the Firewall World

This chapter covers the following topics:

- Some initial definitions
- Starting with the Data Plane: VLANs and VRFs
- VRF-aware services
- Beyond the Data Plane: virtual contexts
- Management access to virtual contexts
- Allocating resources for virtual contexts
- Interconnecting virtual elements
- Issues associated with virtual contexts
- The complete architecture for virtualization

“Between the idea / And the reality / Between the motion / And the act / Falls the shadow” —Thomas S. Eliot

Networks have clearly evolved from an original perception of mere infrastructure up to their recognition as a true business asset. Nevertheless, the capability of using the available telecommunication budget more effectively is still top of mind for most organizations. A similar challenge is faced by IT managers about the usage of their scarce Data Center dollars.

A careful examination of the overall IT resources often unveils a mismatch between original system performance targets and the results actually delivered. A dynamic infrastructure based upon *virtualization* capabilities might significantly contribute to improve the utilization levels of the various classes of IT resources.

Well, the (virtually) almighty word was written: *Virtualization*. But what is that all about?

Is a virtual device a good or a bad thing? Is it “almost” a device?

Drawing on that it is more of a marketing term rather than an engineering one, the simple mention to the word *virtualization* tends to bring more confusion than comfort in technical security discussions.

Because, historically, the word virtualization has been taken as synonymous to *server partitioning*, it is with some awe that many customers find out that security and networking devices might have various features virtualized. For example, by the time this book was written, a single Cisco physical firewall could act as up to 250 virtual firewalls with totally different sets of security rules and management characteristics.

This chapter presents the different levels of virtualization supported by Cisco Firewalls and quickly reviews a potential end-to-end *virtualization architecture*. One of the motivations for this architectural play relates to the opportunity to create several logical layers of converged networking in such a way that traffic belonging to different user communities is effectively segmented and protected throughout the whole path from network ingress to the Data Center.

Some Initial Definitions

Before starting the study of virtualization, you need to become acquainted with two sets of new terms. The first set relates to *network planes*, whereas the second concerns the possible meanings of the word virtualization.

Traffic handled by network devices may be classified into three functional planes:

- **Data Plane:** Also known as the Forwarding Plane, this functional component relates to traffic crossing the device. This typically corresponds to the largest amount of packets flowing through network elements such as routers and firewalls.
- **Control Plane:** This category is composed of traffic directed to the network device and includes functions such as the construction of routing and topology tables, signaling, and maintenance of interface status.
- **Management Plane:** This is a functional class that manages a device by means of its connection to the network. Important examples of protocols belonging to this plane are Syslog, SNMP, and SSH.

Note It is common to see technical texts treating the Management and Control Planes as a single entity. There is no significant issue in adopting such an approach because, even though the protocols involved are different, the traffic in both cases is directed to the router.

The term *virtualization* appears everywhere and can be employed in many different senses within the networking and security domains. To avoid confusion, it is interesting to review some of its classic meanings:

- **Abstraction:** An abstract entity represents physical elements. The Virtual IPs used for Network Address Translation (NAT) or Server Load Balancing and the virtual

addresses employed by First Hop Redundancy Protocols (HSRP, VRRP, and GLBP) fall within this category.

- **Pooling:** Multiple physical elements are treated as one logical entity. Two examples are port aggregation methods (also known as port channeling) and TCP connection pooling, such as used by WAN optimization solutions such as Cisco Wide Area Application Services (WAAS) devices.
- **Partitioning:** In this case, a single physical element is divided into multiple logical entities. VLANs and Virtual Routing and Forwarding (VRF) instances exemplify partitioning applied to the Data Plane (of routers and switches). Security contexts are a way of simultaneously dividing the data and control planes of devices such as firewalls and Server Load Balancers.

This chapter basically focuses on the individual analysis of some partitioning methods and how they can be combined to produce powerful segmentation solutions.

Starting with the Data Plane: VLANs and VRFs

In the beginning was the network. The network was formless and empty, the darkness brought by collisions was over the surface of the deep, and the spirit of the Netlord hovered over the all-encompassing *broadcast domain*.

And the Netlord said, “Let there be routers,” and there was hope for frames and packets. The Netlord saw that routers were good and that they separated broadcast domains, each of them being named a *subnet*.

And the Netlord said, “Let there be an expanse inside each subnet to control the reach of collision domains.” And it was so, and the Netlord called the expansion a LAN switch.

Well, this was just a quick review of *the beginning*. All these creations proved good and relevant, but, as usual, there was room for some evolution. Mainly by considering the possibility of defining smaller realms inside networks (either L2 or L3) that could preserve the characteristics of their parental entities.

Virtual LANs

The usage of LAN switches, in spite of almost eliminating collisions, still resulted in a single broadcast domain. As the number of hosts in networks expanded and the variety of applications running on each computer increased, there was a clear demand for confining broadcasts to well-defined areas inside a physical LAN switch. This scenario gave birth to Virtual LANs (VLAN), a significant implementation of a virtualization technique in the world of networking.

The classic deployment of VLANs is port-based, which means that specific physical ports become part of a given virtual LAN. In the original conception, hosts belonging to this VLAN could communicate directly and were part of the same logical subnet and of a corresponding broadcast domain. Whenever there was need for a host in one VLAN to

communicate with a host in another, a router would come into play, performing its inter-networking functions.

Virtual LANs represent an effort to isolate broadcast domains inside a LAN switch. But VLANs would be much less important if there were no mean to preserve this separation when interconnecting devices (switch-to-switch or switch-to-router, for example) over a single physical link.

To accomplish the goal of carrying traffic belonging to different VLANs over the same link, VLAN trunks are used. IEEE 802.1Q is the standard for tagging Ethernet frames on a VLAN trunk. The 802.1Q trunking device inserts a tag (4 bytes in length) into the original frame and recalculates the frame check sequence (FCS) before sending this frame over the trunk interface. At the other end of the trunk, the tag is removed by the receiving device, and the frame is forwarded to the pertinent VLAN.

Note VLANs might contribute decisively to the scaling of firewall deployments. If a physical interface were to be dedicated to every logical subnet (L3), some firewalls would end up with hundreds of links, which is neither practical nor cheap. (Remember for a while that most firewall devices are built on top of server platforms and that each interface would consume a port on a LAN switch.)

VRFs

A Virtual Routing and Forwarding (VRF) instance is a container of segmented routing information in Layer 3 devices. The basic components of a VRF follow:

- **IP routing table:** Built by the Control Plane of the underlying L3 device, using any combination of static and dynamic means.
- **Forwarding table:** Derived from the IP routing table.
- **Set of interfaces:** The L3 device not only exchanges routing information over these but also forwards the packets whose destinations are reachable through the VRF.

By default a Cisco router has only the *global table*. (And no VRF is defined.) The total number of routing tables in a router can be represented by the formula $N + 1$, in which N corresponds to the number of VRFs created. Figure 6-1 shows a router that has 03 VRFs in addition to its original global table. This figure also shows how to display the global table and the routing table for a specific VRF.

VRFs are created using the `ip vrf` command, which gives access to VRF configuration mode. The indispensable parameter for a VRF to work is the *Route Distinguisher (RD)*, a 64-bit prefix prepended to IP addresses that are part of the VRF, to guarantee their uniqueness. The RD must be unique for each VRF inside a Layer 3 device.

One important bonus that comes with the RD is the possibility of using overlapping IP addresses. To better understand this effect, just recall the role played by area codes in the context of the telephone system. A local phone number XYZW might exist in multiple

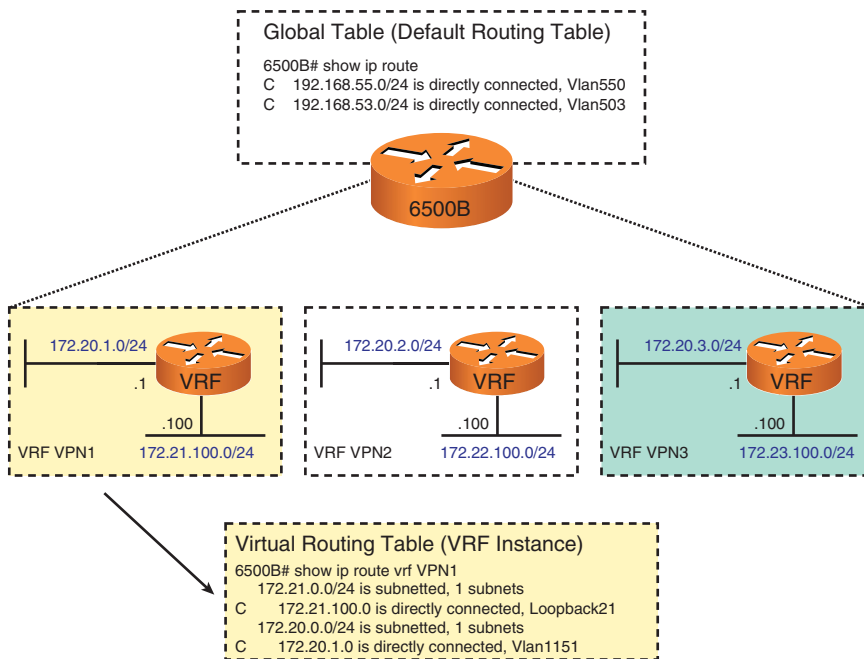


Figure 6-1 Routing Tables on a VRF-Enabled L3 Device

areas (cities, provinces, states, and so on). When communicating to a number that is not part of the same area, the area prefix (area code) must be used to avoid any ambiguity.

Note This section focuses only on the local usage of VRFs, which is referred to as *VRF-lite* or *multi-VRF* capability in Cisco terminology. When used in the original sense of MPLS VPNs, routing information for multiple VRFs can be exchanged between two remote Provider Edge (PE) routers with the help of Multiprotocol BGP and the **route-target** (RT) extended community. The RT should not be confused with the RD. The former serves the purpose of selecting the routes to be included in a given virtual table, whereas the latter ensures that addresses are unique.

Example 6-1 relates to the topology presented in Figure 6-1. First, VRFs are created, and unique RDs are assigned for each of them. In the sequence, the local router interfaces that belong to each VRF are appropriately assigned using the **ip vrf forwarding** command (at the interface level).

The example also shows the interfaces and routes included in each VRF and demonstrates how to ping or telnet to destinations reachable through an interface that is part of a certain VRF.

Example 6-1 *VRF Basics*

```
! Defining VRFs : the Route Distinguisher (RD) is the minimum parameter
ip vrf VPN1
  rd 65060:1
!
ip vrf VPN2
  rd 65060:2
!
ip vrf VPN3
  rd 65060:3
!
! Assigning interfaces to VRFs and configuring IP Addresses
interface Loopback21
  ip vrf forwarding VPN1
  ip address 172.21.100.100 255.255.255.0
  ip ospf network point-to-point
!
interface Loopback22
  ip vrf forwarding VPN2
  ip address 172.22.100.100 255.255.255.0
!
interface Loopback23
  ip vrf forwarding VPN3
  ip address 172.23.100.100 255.255.255.0
!
interface Vlan1151
  ip vrf forwarding VPN1
  ip address 172.20.1.1 255.255.255.0
!
interface Vlan1152
  ip vrf forwarding VPN2
  ip address 172.20.2.1 255.255.255.0
!
interface Vlan1153
  ip vrf forwarding VPN3
  ip address 172.20.3.1 255.255.255.0
!
! Displaying information about a given VRF
6500B#show ip vrf VPN1
  Name                               Default RD          Interfaces
  VPN1                               65060:1             Vlan1151
                                           Loopback21
!
6500B# show ip route vrf VPN1 | begin Gateway
Gateway of last resort is not set
```

```

172.21.0.0/24 is subnetted, 2 subnets
C      172.21.1.0 is directly connected, Vlan1161
C      172.21.100.0 is directly connected, Loopback21
!
! Ping to a destination that is reachable through the 'VPN1' VRF
6500B# ping vrf VPN1 172.20.1.2
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
!
! Telnet to a host that is reachable through the 'VPN2' VRF
6500B#telnet 172.20.2.2 /vrf VPN2
Trying 172.20.2.2 ... Open
User Access Verification
Password:
R1>show tcp brief

```

TCB	Local Address	Foreign Address	(state)
82F5B46C	172.20.2.2.23	172.20.2.1 .15604	ESTAB

Tip The `ip vrf forwarding` command removes the IP address previously configured on the interface. The most convenient approach (to save time and work) is to configure the IP address after assigning the interface to the VRF of interest.

Note Multiple VLANs (and the associated L3 subnets inside each VRF) might exist; however, there is no demand of mapping a VLAN to a single VRF.

Figure 6-2 portrays a scenario used for the study of static routes in VRFs. The VRF-enabled router, whose relevant configurations were registered in Example 6-1, points a static route to the aggregate of networks 172.20.0.0/14, using R1 as its gateway (reachable through interface VLAN 1151, which resides on VRF VPN1).

Example 6-2 shows more details of the arrangement of Figure 6-2. R1 (the external router) has no concept of VRF and treats VRF VPN1 as any regular router.

Example 6-2 VRFs and Static Routes

```

! Configuring a Static Route for VRF 'VPN1' on 6500B
ip route vrf VPN1 172.20.0.0 255.252.0.0 172.20.1.2
!
! Viewing the Static Routes on 6500B
6500B# show ip route vrf VPN1 static
S      172.20.0.0/14 [1/0] via 172.20.1.2
!
! R1 has no concept of VRF and treats the VRF 'VPN1' as a regular router

```



```
R1# show ip route | begin Gateway
Gateway of last resort is not set
    172.20.0.0/24 is subnetted, 2 subnets
C       172.20.20.0 is directly connected, Loopback20
C       172.20.1.0 is directly connected, FastEthernet0/0.1151
S       172.21.0.0/16 [1/0] via 172.20.1.1
```

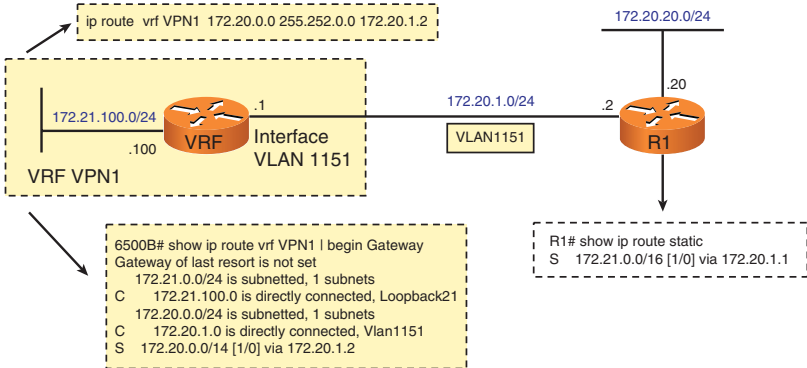


Figure 6-2 VRFs and Static Routes

Figure 6-3 represents a situation in which the three IGP studied in Chapter 5, “Firewalls in the Network Topology,” are deployed for a VRF-enabled router. The routes that populate the VRFs named VPN1, VPN2, and VPN3 were respectively learned using OSPF, EIGRP, and RIPv2. Examples 6-3 through 6-5 present the details for each IGP.

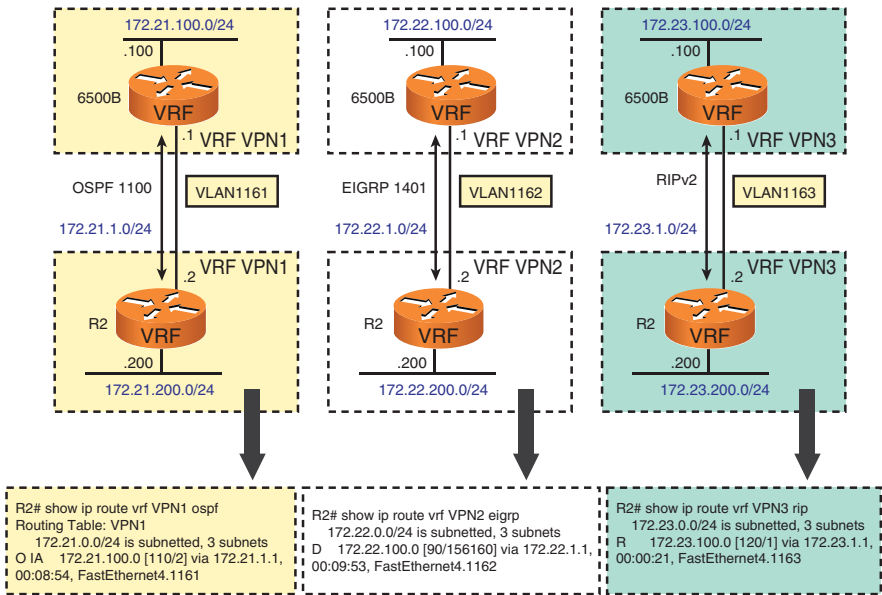


Figure 6-3 VRFs and Dynamic Routing Protocols

Example 6-3 deals with the OSPF portion of the scenario illustrated in Figure 6-3. One distinctive characteristic of VRF-enabled OSPF is that each VRF requires a dedicated OSPF process. In the example, the number 1100 was chosen to accommodate the OSPF process bound to VRF VPN1. Another point of attention in this case is the **capability vrf-lite** command, which serves to tell that this is a *VRF-lite* router and not a full MPLS-VPN Provider Edge Router (Some more complex checks typical in the MPLS-VPN world do not need to be performed.)

Example 6-3 *Enabling OSPF for a VRF*

```
! Relevant configurations on 6500B

interface Vlan 1161
  ip vrf forwarding VPN1
  ip address 172.21.1.1 255.255.255.0
!
router ospf 1100 vrf VPN1
  router-id 172.21.100.100
  capability vrf-lite
  network 172.21.1.0 0.0.0.255 area 0
  network 172.21.100.0 0.0.0.255 area 1
!
! Relevant configurations on R2

interface FastEthernet4.1161
  encapsulation dot1Q 1161
  ip vrf forwarding VPN1
  ip address 172.21.1.2 255.255.255.0
!
router ospf 1100 vrf VPN1
  router-id 172.21.200.200
  capability vrf-lite
  network 172.21.1.0 0.0.0.255 area 0
  network 172.21.200.0 0.0.0.255 area 2
!
! OSPF Adjacency is established (6500B's perspective)
%OSPF-5-ADJCHG: Process 1100, Nbr 172.21.200.200 on Vlan1161 from LOADING to FULL,
Loading Done
!
```

```

6500B# show ip ospf 1100 neighbor
Neighbor ID      Pri   State             Dead Time   Address        Interface
172.21.200.200   1     FULL/DR           00:00:36    172.21.1.2     Vlan1161
!
! Routes learned via OSPF for VRF VPN1
6500B# show ip route vrf VPN1 ospf
      172.21.0.0/24 is subnetted, 3 subnets
0 IA      172.21.200.0 [110/2] via 172.21.1.2, 00:00:14, Vlan1161
!
! Routing Protocols enabled for VRF VPN1
6500B# show ip protocols vrf VPN1
Routing Protocol is "ospf 1100"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 172.21.100.100
  It is an area border router
  Number of areas in this router is 2. 2 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
    172.21.1.0 0.0.0.255 area 0
    172.21.100.0 0.0.0.255 area 1
  Routing Information Sources:
    Gateway         Distance      Last Update
    172.21.200.200    110          08:27:02
  Distance: (default is 110)

```

Note Because VRF-enabled OSPF needs a separate routing process, all the OSPF-specific commands (`show ip ospf neighbor` and `show ip ospf database`, for instance) should refer to the OSPF process number, rather than to the VRF.

Example 6-4 takes care of the EIGRP portion of Figure 6-3, and Example 6-5 does an equivalent job for RIPv2. In the opposite range of the spectrum from OSPF, both EIGRP and RIP enable the creation of several routing instances (defined with the `address-family ipv4 vrf` command) below the same routing process. The configurations for each address-family are basically identical to those presented in Chapter 5. The noticeable exception is that each EIGRP instance needs a dedicated AS number, which is achieved with the `autonomous-system` command.

Example 6-4 Enabling EIGRP for a VRF

```

! Relevant configurations on 6500B

interface Vlan 1162

```

```

ip vrf forwarding VPN2
ip address 172.22.1.1 255.255.255.0
!
router eigrp 1400
no auto-summary
!
address-family ipv4 vrf VPN2
network 172.22.1.0 0.0.0.255
network 172.22.100.0 0.0.0.255
no auto-summary
autonomous-system 1401
eigrp router-id 172.22.100.100
exit-address-family
!
! Relevant configurations on R2

interface FastEthernet4.1162
encapsulation dot1Q 1162
ip vrf forwarding VPN2
ip address 172.22.1.2 255.255.255.0
!
router eigrp 1400
no auto-summary
!
address-family ipv4 vrf VPN2
network 172.22.1.0 0.0.0.255
network 172.22.200.0 0.0.0.255
no auto-summary
autonomous-system 1401
eigrp router-id 172.22.200.200
exit-address-family
!
! EIGRP Adjacency is established (6500B's perspective)

%DUAL-5-NBRCHANGE: IP-EIGRP(6) 1401: Neighbor 172.22.1.2 (Vlan1162) is up:
new adjacency
!
6500B# show ip route vrf VPN2 eigrp
      172.22.0.0/24 is subnetted, 3 subnets
D       172.22.200.0 [90/130816] via 172.22.1.2, 00:00:39, Vlan1162
!
6500B# show ip eigrp vrf VPN2 neighbors
IP-EIGRP neighbors for process 1401
H   Address                               Interface      Hold Uptime    SRTT   RTO   Q   Seq
                               (sec)          (ms)          Cnt Num

```

```

0    172.22.1.2                V11162                14 00:02:30    4    200    0    15
!
6500B# show ip eigrp vrf VPN2 topology
IP-EIGRP Topology Table for AS(1401)/ID(172.22.100.100) Routing Table: VPN2
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status
P 172.22.200.0/24, 1 successors, FD is 130816
    via 172.22.1.2, Vlan1162
P 172.22.1.0/24, 1 successors, FD is 2816
    via Connected, Vlan1162
P 172.22.100.0/24, 1 successors, FD is 128256
    via Connected, Loopback22

```

Example 6-5 Enabling RIPv2 for a VRF

```

! Relevant configurations on 6500B

interface Vlan 1163
 ip vrf forwarding VPN3
 ip address 172.23.1.1 255.255.255.0
!
router rip
 version 2
 no auto-summary
!
address-family ipv4 vrf VPN3
 network 172.23.0.0
 no auto-summary
 exit-address-family
!

! Relevant configurations on R2

interface FastEthernet4.1163
 encapsulation dot1Q 1163
 ip vrf forwarding VPN3
 ip address 172.23.1.2 255.255.255.0
!
router rip
 version 2
 no auto-summary
!
address-family ipv4 vrf VPN3
 network 172.23.0.0
 no auto-summary

```

```

exit-address-family
!
! RIP routes are received on 6500B

6500B# show ip route vrf VPN3 rip
      172.23.0.0/24 is subnetted, 3 subnets
R       172.23.200.0 [120/1] via 172.23.1.2, 00:00:24, Vlan1163
!
6500B# show ip rip database vrf VPN3
172.23.0.0/16      auto-summary
172.23.1.0/24      directly connected, Vlan1163
172.23.100.0/24    directly connected, Loopback23
172.23.200.0/24
      [1] via 172.23.1.2, 00:00:19, Vlan1163
!
6500B# show ip protocols vrf VPN3
Routing Protocol is "rip"
  Sending updates every 30 seconds, next due in 12 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface          Send  Recv  Triggered RIP  Key-chain
    Vlan1163            2     2
    Loopback23          2     2
Maximum path: 4
  Interface          Send  Recv  Triggered RIP  Key-chain
Routing for Networks:
  172.23.0.0
Routing Information Sources:
  Gateway            Distance      Last Update
  172.23.1.2          120          00:00:02
Distance: (default is 120)

```

Note Example 6-3 registered that the OSPF-specific commands make reference to the routing process number instead of the VRF. On the contrary, RIP and EIGRP refer to the specific VRF where the IGP is configured. (For an illustration of this, revisit the `show ip eigrp neighbors` and `show ip rip database` commands in the previous two examples.)

VRF-Aware Services

As virtualization gains strength in the networking and security worlds, IOS development shows a clear trend of being more VRF-oriented. The term *VRF-aware* is common in recent IOS feature description documents, meaning that a known functionality has been adapted so that it can be configured within the scope of a particular VRF.

This is useful because sometimes there is a business (or operational) demand for making a service available only for a specific customer (in the case of service providers) or a given internal department (for Enterprises). As VRF-awareness becomes widespread, users have the freedom to allow a given service only for the groups that directly need it, rather than having the traditional obligation of configuring it globally. This serves resource savings initiatives and also provides administrators with more control and visibility of service usage.

Example 6-6 assembles some sample commands to enable services such as IP multicast routing, Syslog, NTP, and SNMP on a per-VRF basis. Other remarkable examples are NAT, RADIUS, TACACS+, stateful firewall, intrusion prevention, and IPsec VPNs.

Example 6-6 Sample VRF-Aware Services

```
ip multicast-routing vrf VPN2
logging host 172.20.20.20 vrf VPN1
ntp server vrf VPN1 172.20.20.21
snmp-server host 172.20.20.22 vrf VPN1 *****
```

Beyond the Data Plane—Virtual Contexts

The two previous sections reviewed how VLANs and VRFs lend themselves to the demand of virtualizing the Data Plane, either for Layer 2 or Layer 3. Although this method already brings much value for network and security designs, it is indispensable to observe that, for instance, the device configuration file is shared between all the virtual elements.

This section introduces the concept of Virtual Contexts, which focus on materializing, as faithfully as possible, the behavior of a virtual device. For devices such as ASA and FWSM, each context owns a set of interfaces, a routing table, security policies (ACLs, NAT rules, and inspection policies), and management settings (configuration files, admin users, and so on).

ASA and FWSM include a kind of root context known as the System Execution Space (or System Partition), which is used for defining all the other contexts. This special partition is tightly bound to an entity called the *admin-context*. The admin-context includes, among its responsibilities, the provision of network interfaces to the reserved system partition. The default name for the admin-context is *admin*.

Users entitled to access the admin-context can access any other context. That is the reason for avoiding, as much as possible, using the admin context just as a regular one (non-admin).

Figure 6-4 illustrates the creation of the admin-context and two regular contexts, inside the system execution space, for a dedicated Cisco Firewall (FWSM or ASA). The figure also reveals the main pieces of information that can be individualized for contexts and some particularities of the admin context.

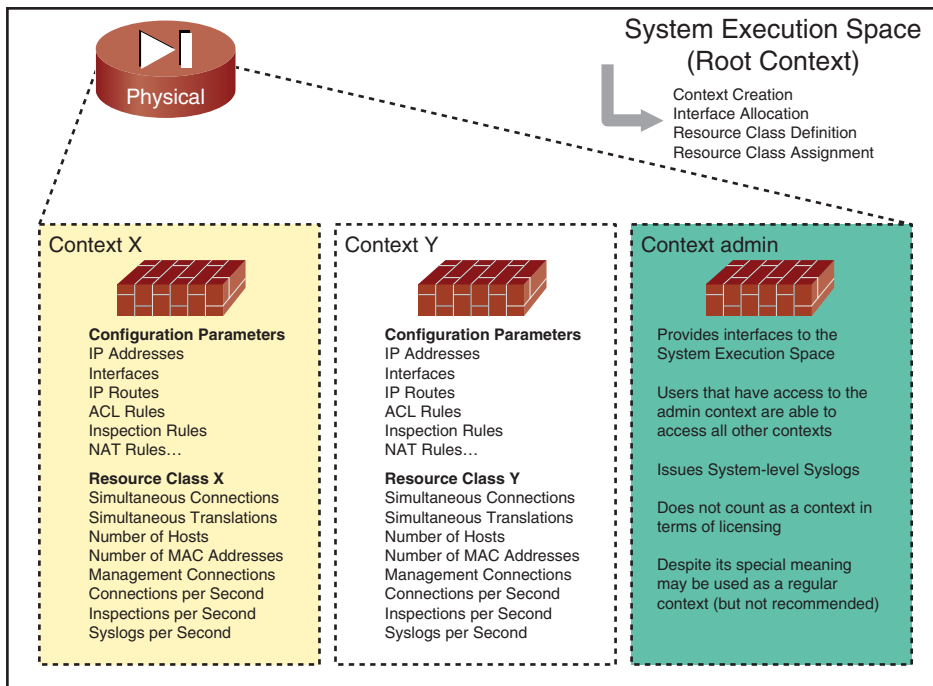


Figure 6-4 Overview of Virtual Contexts

Example 6-7 teaches how to find the operation mode for a dedicated Cisco Firewall. The example also characterizes that *single mode* has no concept of security context.

Example 6-7 Single Mode Operation

```
FWSM# show mode
Security context mode: single
The flash mode is the SAME as the running mode.
!
! Viewing the contents of the Flash Memory in single mode
FWSM# dir
```



```

Directory of disk:/
4      -rw-   1903      14:09:28 Sep 02 2008  old_running.cfg
59674624 bytes total (59666432 bytes free)
!
! There is no concept of context for single mode operation
FWSM# show context
      ^
ERROR: % Invalid input detected at '^' marker.

```

Example 6-8 documents the process of converting from single to multiple mode on FWSM. Some relevant facts follow:

- After entering all the confirmations, the module reboots.
- The admin-context is automatically created, named admin, and assigned the VLAN interfaces that were available in single mode operation. The **show context** command reveals that admin is the only active context initially.
- A file named admin.cfg holds the configuration of the admin context.

Example 6-9 adds more details concerning the results of the conversion process. The **show startup-config** command reveals the existence of the admin context (with its allocated interfaces and **config-url**), whereas the **more disk:/admin.cfg** command is a way to display the configuration of the admin context from the system partition. Some noticeable points follow:

- The initial interface state is administratively down (or simply shut down).
- The system partition shows the available interfaces for the entire physical module, which were assigned using the firewall module **vlan-group** command from Catalyst 6500 configuration.
- Most of the commands that were active in single mode (**ip address**, **security-level**, **nameif**, **class-map**, **policy map**, and so on) now appear inside the admin context.

Example 6-8 Enabling Multiple Mode

```

! Converting to multiple mode operation

FWSM(config)# mode multiple
WARNING: This command will change the behavior of the device
WARNING: This command will initiate a Reboot
Proceed with change mode? [confirm] <Enter>
Convert the system configuration? [confirm] <Enter>
!
The old running configuration file will be written to disk

```

```

The admin context configuration will be written to disk
The new running configuration file was written to disk
Security context mode: multiple
!
! The Firewall Module is rebooted

[Connection to 127.0.0.41 closed by foreign host]
6500B#
SP: The PC in slot 4 is shutting down. Please wait ...
SP: shutdown_pc_process:No response from module 4
%C6KPWR-SP-4-DISABLED: power to module in slot 4 set off (Reset)
%DIAG-SP-6-RUN_MINIMUM: Module 4: Running Minimal Diagnostics...
%DIAG-SP-6-DIAG_OK: Module 4: Passed Online Diagnostics
%OIR-SP-6-INSCARD: Card inserted in slot 4, interfaces are now online
!
! Result of the conversion process

FWSM# show mode
Security context mode: multiple
The flash mode is the SAME as the running mode.
!
FWSM# show context
Context Name      Class      Interfaces      Mode      URL
*admin            default    Vlan1102,Vlan1240,
                  Vlan1242      Routed      disk:/admin.cfg

Total active Security Contexts: 1
!
FWSM# show context detail admin
Context "admin", is ADMIN and active
  Config URL: disk:/admin.cfg
  Real Interfaces: Vlan1102, Vlan1240, Vlan1242
  Mapped Interfaces: Vlan1102, Vlan1240, Vlan1242
  Class: default, Flags: 0x00001857, ID: 1
!
FWSM# dir
Directory of disk:/
4      -rw-   1562      23:48:00 Mar 10 2010  admin.cfg
3      -rw-   1645      23:48:00 Mar 10 2010  old_running.cfg
59674624 bytes total (59662336 bytes free)

```

Example 6-9 More Details About the Conversion Process

```

! Configuration of the System Partition
FWSM# show startup-config

```

```

: Saved
: Written by enable_15 at 23:48:00.470 BRT Thu Mar 11 2010
FWSM Version 4.0(3) <system>
hostname FWSM
enable password 2KFQnbNIdI.2KYOU encrypted
interface Vlan1102
  description *** Management Access ***
interface Vlan1240
  shutdown
interface Vlan1242
  shutdown
ftp mode passive
no pager
no failover
no asdm history enable
arp timeout 14400
console timeout 0
prompt hostname context
class default
  limit-resource all 0
!
admin-context admin
context admin
  allocate-interface Vlan1102
  allocate-interface Vlan1240
  allocate-interface Vlan1242
  config-url disk:/admin.cfg
!
! Configuration of the admin Context

FWSM# more disk:/admin.cfg
: Saved
: Written by enable_15 at 23:48:00.470 BRT Thu Mar 11 2010
FWSM Version 4.0(3) <context>
hostname FWSM
enable password 2KFQnbNIdI.2KYOU encrypted
names
dns-guard
interface Vlan1102
  description *** Management Access ***
  nameif mgmt
  security-level 100
  ip address 192.168.2.22 255.255.255.0
interface Vlan1240

```

```

shutdown
nameif out1
security-level 0
ip address 172.16.240.22 255.255.255.0
interface Vlan1242
shutdown
nameif dmz1
security-level 50
ip address 172.16.242.22 255.255.255.0
passwd 2KFQnbNIdI.2KYOU encrypted
no pager
logging enable
logging console debugging
[ output suppressed ]
class-map inspection_default
  match default-inspection-traffic
class-map default
policy-map global_policy
  class inspection_default
    inspect dns maximum-length 512
    inspect ftp
  [ output suppressed ]
    inspect xdmcp
service-policy global_policy global
: end

```

Example 6-10 assembles the basic context management tasks:

- It shows how to remove interfaces that are automatically allocated to the **admin-context** while converting from single to multiple mode.
- It teaches how to create a context (named CT1 in this case), assign the appropriate interfaces, and define its **config-url** (which holds the configuration file).
- You can see that, although the **config-url** was created (and displays with the **show context** command), it does not appear yet in the **dir** command output.
- The **changeto context** CT1 grants access to the context just defined. It is interesting to observe that the context prompt is FWSM/CT1# (resulting from the combination “hostname for the system partition / context name”).
- The **show firewall** command reveals that the default operation mode for a context is routed. In this example, the operation mode is changed to transparent.
- After saving the configuration of the CT1 context with a write memory and going back to the system partition with the **changeto system** command, the CT1.cfg file shows up in the dir output

- The **show context** command displays the active contexts, the interfaces allocated for each of them, and the respective **config-urls**. It also shows information about the context mode of operation (transparent or routed).

Note The FWSM supports any combination of contexts in transparent or routed mode. This is not possible for ASA appliances. An important design decision for ASA about virtualization relates to determining the operation mode for contexts: all of them routed or all transparent.

Example 6-10 Basic Context Management Tasks

```

! Removing interfaces from the admin context
FWSM(config)# context admin
FWSM(config-ctx)# no allocate-interface Vlan1240
FWSM(config-ctx)# no allocate-interface Vlan1242
!
! Creating a context called CT1 and allocating an interface

FWSM(config)# context CT1
%FWSM-5-504001: Security context CT1 was added to the system
FWSM(config-ctx)# allocate-interface vlan1261
%FWSM-4-411001: Line protocol on Interface , changed state to up
!
! Defining the file that will store context configuration
FWSM(config)# context CT1
FWSM(config-ctx)# config-url disk:/CT1.cfg
WARNING: Could not fetch the URL disk:/CT1.cfg
INFO: Creating context with default config
!
FWSM# show context CT1
Context Name      Class      Interfaces      Mode      URL
CT1               default    Vlan1261,Vlan1262  Routed    disk:/CT1.cfg
!
FWSM# dir
Directory of disk:/
4      -rw-   1562      23:48:00 Mar 10 2010  admin.cfg
3      -rw-   1645      23:48:00 Mar 10 2010  old_running.cfg
!
! Accessing context CT1 and verifying its configuration

FWSM# changeto context CT1
FWSM/CT1# show firewall
Firewall mode: Router
!

```

```

! Changing the context to Transparent Mode
FWSM/CT1(config)# firewall transparent
FWSM/CT1# write memory
!
FWSM/CT1# show firewall
Firewall mode: Transparent
!
FWSM/CT1# show running-config
: Saved
:
FWSM Version 4.0(3) <context>
!
firewall transparent
hostname CT1
enable password 8Ry2YjIyt7RRXU24 encrypted
names
dns-guard
!
interface Vlan1261
  no nameif
  no security-level
!
interface Vlan1262
  no nameif
  no security-level
!
passwd 2KFQnbNIdI.2KYOU encrypted
[ output suppressed ]
policy-map global_policy
  class inspection_default
    inspect dns maximum-length 512
[ output suppressed ]
    inspect tftp
!
service-policy global_policy global
Cryptochecksum:d0259d274c651b1aa3f8f6b42bd1a0cd
: end
!
! Back to the system execution space

FWSM/CT1# changeto system
FWSM# dir
Directory of disk:/
4          -rw-   1562          23:48:00 Mar 10 2010  admin.cfg

```

```
5      -rw-   1215           17:38:50 Mar 12 2010  CT1.cfg
3      -rw-   1645           23:48:00 Mar 10 2010  old_running.cfg
!
FWSM# show context
Context Name      Class      Interfaces      Mode      URL
*admin            default    Vlan1102        Routed     disk:/admin.cfg
CT1               default    Vlan1261,Vlan1262  Transparent disk:/CT1.cfg
Total active Security Contexts: 2
```

Figure 6-5 employs context CT1, from Example 6-10, to illustrate typical Transparent mode operation for the FWSM. Two bridge-groups are created inside CT1:

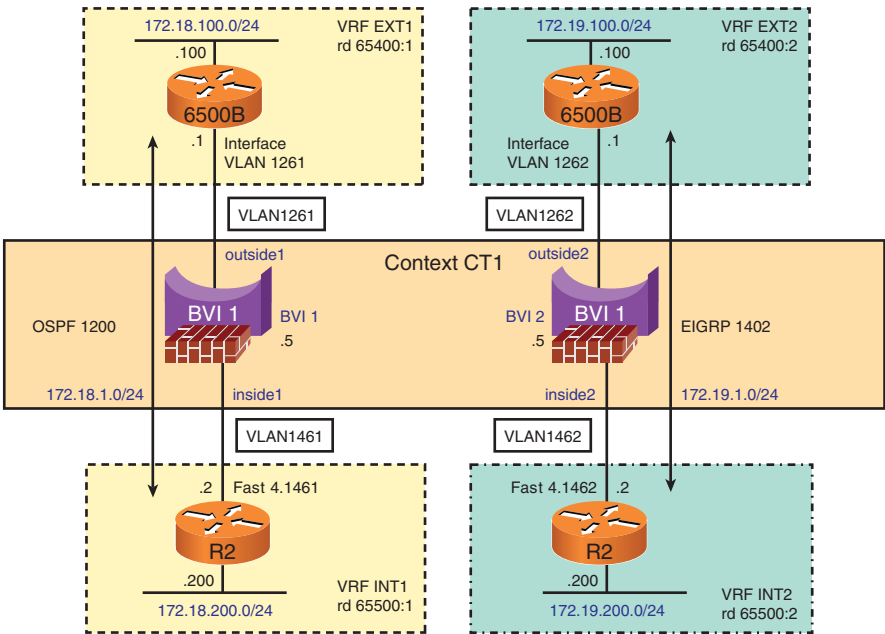


Figure 6-5 Sample Virtualization Scenario with a Context in Transparent Mode

- **Bridge-group 1:** Interconnects VLANs 1261 and 1461. The logical interface BVI 1 assigns the IP address 172.18.1.5/24 to this **bridge-group** for management purposes.
- **Bridge-group 2:** Interconnects VLANs 1262 and 1462. Interface BVI 2 provides the management address 172.19.1.5/24 for this **bridge-group**.

The configuration details for this setup are registered in Example 6-11, which includes information about the interfaces belonging to **bridge-group 1**. Although configured on

interface BVI 1, the IP address appears on the member interfaces (VLAN 1261 and VLAN 1461).

Example 6-12 assembles the routing details for the topology shown in Figure 6-5:

- An OSPF adjacency is built between VRF EXT1 on 6500B (VLAN 1261) and VRF INT1 (VLAN 1461) on R2. Given that the FWSM has implicit deny rules in action, the adjacency does not come up initially. (Refer to **deny protocol 89** messages on interfaces **inside1** and **outside1**.) After configuring the appropriate permissions, 6500B and R2 become OSPF neighbors.
- EIGRP 1402 is configured on Routers 6500B and R2 on VRFs EXT2 (VLAN 1262) and INT2 (VLAN 1462), respectively. The **implicit deny** rules for transparent mode block the connection setup between the potential EIGRP neighbors. The adequate ACLs are then configured, therefore enabling the adjacency to be established.

In Examples 6-11 and 6-12 connection setup in transparent and routed modes are basically identical. (With the exception that Transparent mode enables IGP connections to be set up through the firewall.)

Example 6-11 Sample Scenario Using a Transparent Context (1)

```
! Viewing information about the context CT1 on the System Partition
FWSM# show running-config context CT1
context CT1
  allocate-interface Vlan1261
  allocate-interface Vlan1262
  allocate-interface Vlan1461
  allocate-interface Vlan1462
  config-url disk:/CT1.cfg
!
! Relevant Configurations on Context CT1

firewall transparent
!
interface Vlan1261
  nameif outside1
  bridge-group 1
  security-level 0
!
interface Vlan1262
  nameif outside2
  bridge-group 2
  security-level 0
!
interface Vlan1461
  nameif inside1
```



```

bridge-group 1
security-level 100
!
interface Vlan1462
nameif inside2
bridge-group 2
security-level 100
!
interface BVI1
description ** L3 interface for Bridge-Group 1 **
ip address 172.18.1.5 255.255.255.0
!
interface BVI2
description ** L3 interface for Bridge-Group 2 **
ip address 172.19.1.5 255.255.255.0
!
! Displaying Information about interfaces

FWSM/CT1# show interface vlan 1261 | include line|address
Interface Vlan1261 "outside1", is up, line protocol is up
      MAC address 001b.d4de.3580, MTU 1500
      IP address 172.18.1.5, subnet mask 255.255.255.0
!
FWSM/CT1# show interface vlan 1461 | include line|address
Interface Vlan1461 "inside1", is up, line protocol is up
      MAC address 001b.d4de.3580, MTU 1500
      IP address 172.18.1.5, subnet mask 255.255.255.0
!
FWSM/CT1# show interface bvi 1
Interface BVI1 "", is up, line protocol is up
      Description: ** L3 interface for Bridge-Group 1 **
      Available but not configured via nameif

```

Example 6-12 Sample Scenario Using a Transparent Context (2)

```

! Relevant Configurations on 6500B
router ospf 1200 vrf EXT1
router-id 172.18.100.100
capability vrf-lite
network 172.18.1.0 0.0.0.255 area 0
network 172.18.100.0 0.0.0.255 area 1
!
router eigrp 1400
no auto-summary

```

```

!
address-family ipv4 vrf EXT2
    network 172.19.1.0 0.0.0.255
    network 172.19.100.0 0.0.0.255
    no auto-summary
    autonomous-system 1402
    eigrp router-id 172.19.100.100
exit-address-family

```

! Relevant Configurations on R2

```

router ospf 1200 vrf INT1
    router-id 172.18.200.200
    capability vrf-lite
    network 172.18.1.0 0.0.0.255 area 0
    network 172.18.200.0 0.0.0.255 area 2

```

```
!
```

```
router eigrp 1400
```

```
    no auto-summary
```

```
!
```

```

address-family ipv4 vrf INT2
    network 172.19.1.0 0.0.0.255
    network 172.19.200.0 0.0.0.255
    no auto-summary
    autonomous-system 1402
    eigrp router-id 172.19.200.200
exit-address-family

```

! OSPF messages are denied by the FWSM implicit rules

```
%FWSM-4-106023: Deny protocol 89 src outside1:172.18.1.1 dst inside1:224.0.0.5 by
access-group "" [0x0, 0x0]
```

```
%FWSM-4-106023: Deny protocol 89 src inside1:172.18.1.2 dst outside1:224.0.0.5 by
access-group "" [0x0, 0x0]
```

```
!
```

! EIGRP messages are denied by the FWSM implicit rules

```
%FWSM-4-106023: Deny protocol 88 src outside2:172.19.1.1 dst inside2:224.0.0.10 by
access-group "" [0x0, 0x0]
```

```
%FWSM-4-106023: Deny protocol 88 src inside2:172.19.1.2 dst outside2:224.0.0.10 by
access-group "" [0x0, 0x0]
```

```
!
```

! OSPF connection setup after appropriate ACLs come into play

```
%FWSM-6-106100: access-list INSIDE1 permitted ospf inside1/172.18.1.2(1) ->
outside1/224.0.0.5(1) hit-cnt 1 (first hit) [0x72c8e825, 0x0]
```

```
%FWSM-6-302022: Built IP protocol 89 connection 144547124565905351 for
```

```

inside1:172.18.1.2 (172.18.1.2) to outside1:224.0.0.5 (224.0.0.5)
%FWSM-6-106100: access-list OUTSIDE1 permitted ospf outside1/172.18.1.1(1) ->
inside1/224.0.0.5(1) hit-cnt 1 (first hit) [0xcffc50a, 0x0]
%FWSM-6-302022: Built IP protocol 89 connection 144547128860872649 for
outside1:224.0.0.5 (224.0.0.5) to inside1:172.18.1.1 (172.18.1.1)
!
! EIGRP connection setup after appropriate ACLs are applied

%FWSM-6-106100: access-list INSIDE2 permitted eigrp inside2/172.19.1.2(1) ->
outside2/224.0.0.10(1) hit-cnt 1 (first hit) [0xc0c106b6, 0x0]
%FWSM-6-302022: Built IP protocol 88 connection 144547137450807247 for
inside2:172.19.1.2 (172.19.1.2) to outside2:224.0.0.10 (224.0.0.10)
%FWSM-6-106100: access-list OUTSIDE2 permitted eigrp outside2/172.19.1.1(1) ->
inside2/172.19.1.2(1) hit-cnt 1 (first hit) [0x2713c899, 0x0]
%FWSM-6-302022: Built IP protocol 88 connection 144547137450807248 for
outside2:172.19.1.2 (172.19.1.2) to inside2:172.19.1.1 (172.19.1.1)
%FWSM-6-106100: access-list OUTSIDE2 permitted eigrp outside2/172.19.1.1(1) ->
inside2/224.0.0.10(1) hit-cnt 1 (first hit) [0x8836d8c1, 0x0]
%FWSM-6-302022: Built IP protocol 88 connection 144547133155839947 for
outside2:224.0.0.10 (224.0.0.10) to inside2:172.19.1.1 (172.19.1.1)
%FWSM-6-106100: access-list INSIDE2 permitted eigrp inside2/172.19.1.2(1) ->
outside2/172.19.1.1(1) hit-cnt 1 (first hit) [0x415d3ab0, 0x0]
%FWSM-6-302022: Built IP protocol 88 connection 144547137450807249 for
inside2:172.19.1.2 (172.19.1.2) to outside2:172.19.1.1 (172.19.1.1)

```

Example 6-13 shows the system partition on an ASA appliance, with the goal of pointing out some differences for FWSM:

- FWSM always uses VLAN interfaces allocated from the pool of VLANs available on the hosting Catalyst 6500. On the contrary, ASA can use physical and logical interfaces.
- The VLAN assignment for a Dot1Q subinterface on ASA is done at the system execution space. No VLAN assignment is performed inside virtual contexts.

This example shows that the process for allocating physical and logical interfaces for a context in ASA is exactly the same.

Example 6-13 Interface Allocation on ASA

```

interface GigabitEthernet0/1.1201
  vlan 1201
!
interface GigabitEthernet0/1.1202
  vlan 1202
!

```

```

interface Management0/0.1100
description *** MANAGEMENT SEGMENT ***
vlan 1100
!
context LAB2
description * VPN LAB (with ASA1) **
allocate-interface GigabitEthernet0/1.1201-GigabitEthernet0/1.1202
config-url disk0:/LAB2.cfg
!
context admin
allocate-interface GigabitEthernet0/0
allocate-interface Management0/0.1100
config-url disk0:/admin.cfg

```

Management Access to Virtual Contexts

After learning how to change firewall operations from single to multiple mode and getting familiar with context setup, it is now time to focus on management access to the Virtual Contexts.

The previous section already established that the system partition and the **admin-context** play the special role to provide access to the physical element as a whole. It also showed that the **changeto context** command enables an user entitled to access **the admin-context** to view (and change) the configuration of any regular context. Although this might fit the management demands of most environments, there exist situations in which direct access to a regular context becomes important. Two such scenarios follow:

- **Virtual Context assigned for a customer of a service provider:** The customer might access its own regular context, without touching any other context.
- **Virtual Context assigned for a department within an Enterprise deployment:** An admin group is in charge of setting up contexts and assigning the suitable interfaces and resources for each of them. After that, each department creates and administers its specific parameters (ACLs, NAT definitions, inspection rules, routes, and so on).

Figure 6-6 shows an ASDM System View for an ASA appliance that includes multiple contexts. The figure highlights an individual context's perspective for interface status, CPU, memory, and connections.

Figure 6-7 presents an ASDM screen that summarizes information about Virtual Contexts for an ASA appliance. This view is basically a graphic version of the **show context** command (when executed within the system partition).

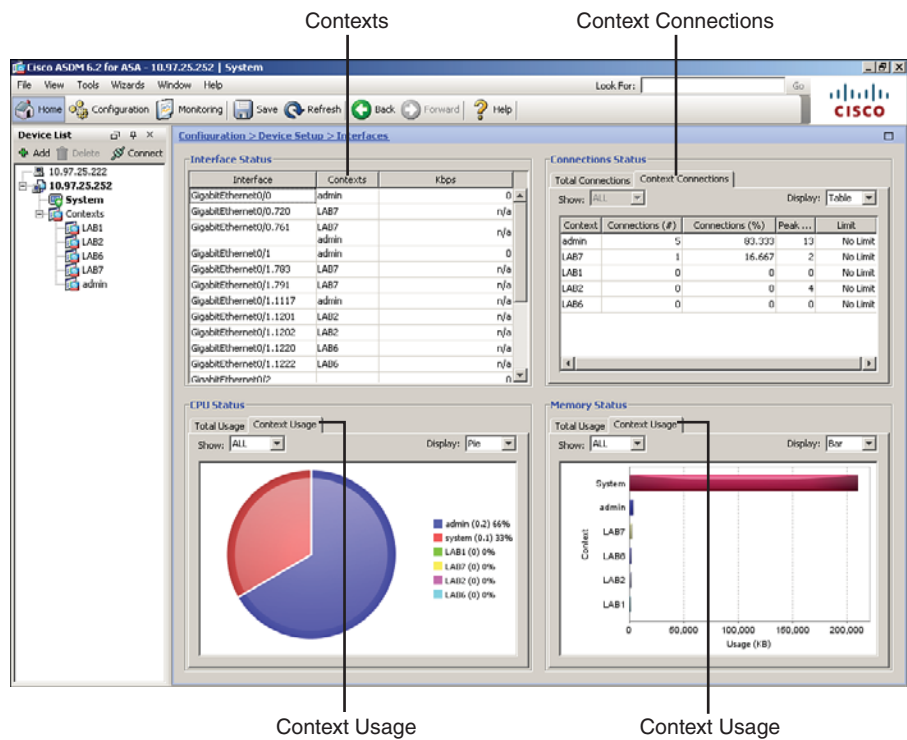


Figure 6-6 ASDM—System View

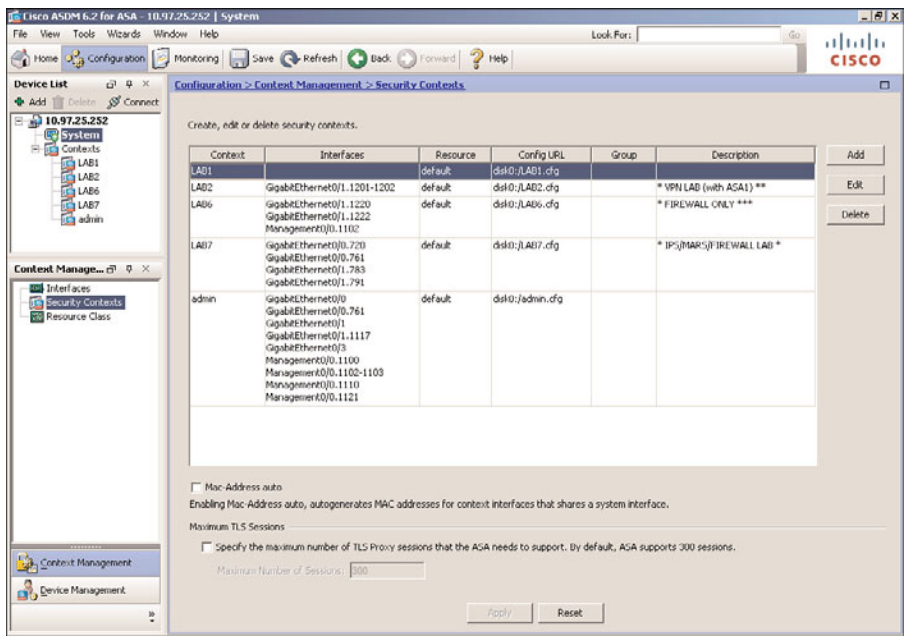


Figure 6-7 ASDM—Displaying Information About Virtual Contexts

Figure 6-8 shows the Firewall Rules table for a regular context named LAB2 on an ASA appliance. This is the view of a privileged user who is allowed to remotely access the system partition (through the admin context) and switch between contexts.

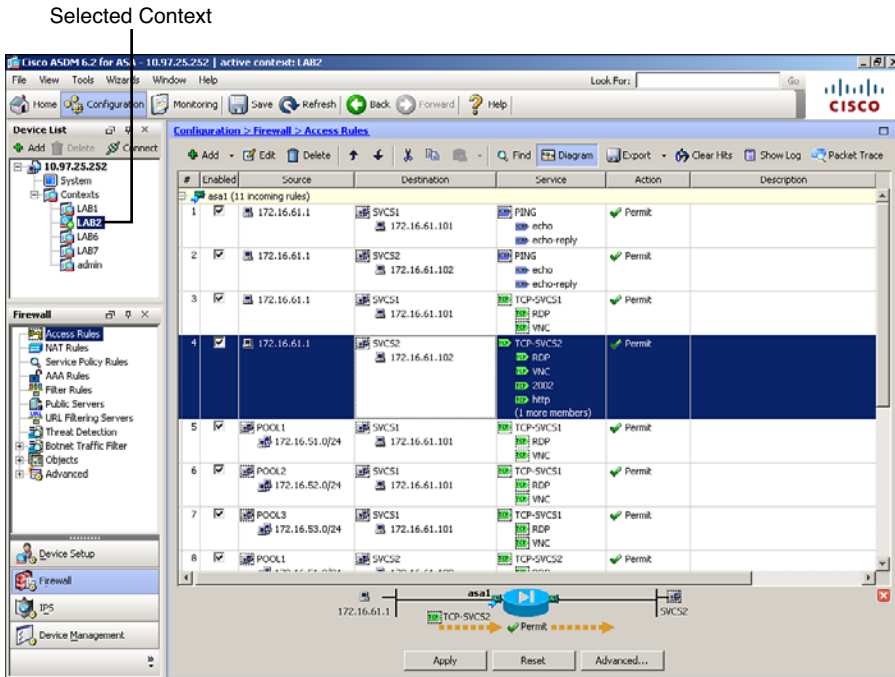


Figure 6-8 ASDM— Displaying the Rules Table for a Virtual Context

Figure 6-9 shows an ASDM screen for a user that directly reached a regular context on ASA. This was made possible by specific management settings inside this particular context (similar to those documented for a physical firewall in Chapter 3, “Configuration Fundamentals”). The CLI tool was used in ASDM (refer to Chapter 4, “Learn the Tools. Know the Firewall”) to underline some relevant facts:

- The **show context** command reveals that this is the context called LAB2.
- The attempt to access the **admin-context** by issuing the **changeto context admin** command shows that this is not allowed from a regular context. The behavior would be exactly the same for a similar attempt from a Telnet or SSH prompt.

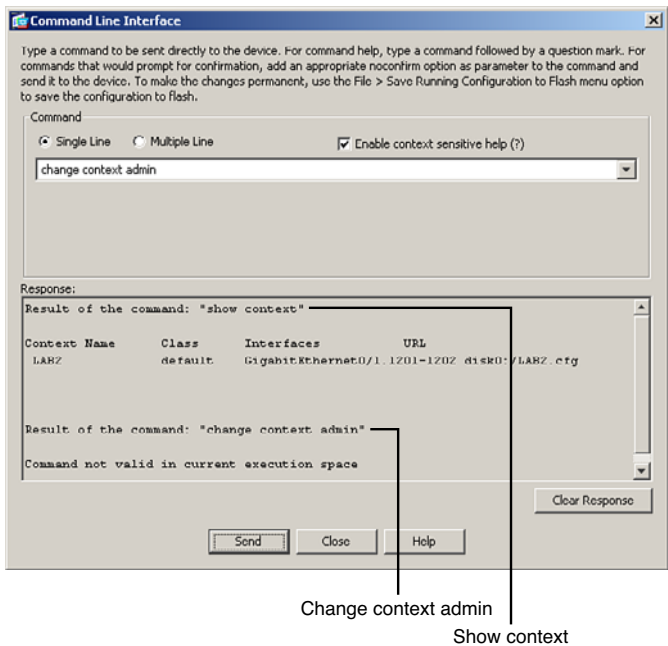


Figure 6-9 ASDM— Direct Access to a non-admin Context

Allocating Resources to Virtual Contexts

One important aspect of virtualized devices is the possibility of allocating resources for each virtual context. There is no requirement for equally dividing the resources among the configured contexts.

Suppose for a while that your company has three computing environments in the Data Center, respectively named INTERNET, INTRANET, and APPS-DEV. Also consider the following:

- There is a demand of supporting up to 10,000 simultaneous connections and 1,000 cps (connections per second) for the INTERNET.
- The correspondent values for these parameters within the INTRANET environment are 20,000 and 5000 cps.
- The APPS-DEV environment is intended to serve as a testing platform for applications under development. It was conceived to emulate functionality as accurately as possible but does not require compatible performance attributes.

The scenario just described might be materialized with the aid of Virtual Contexts on ASA appliances or FWSMs. By employing resource allocation mechanisms, contexts of suitable size might be constructed.

Example 6-14 shows the configurable resource types on ASA-based Firewalls. There are resource categories whose values are given as a *rate* and others that use *absolute limits*.

Example 6-14 *Configurable Resource Types*

```

FWSM# show resource types
Rate limited resource types:
  Conns           Connections/sec
  Fixups           Fixups/sec
  Syslogs          Syslogs/sec
Absolute limit types:
  Conns           Connections
  Hosts           Hosts
  IPSec           IPSec Mgmt Tunnels
  Mac-addresses    MAC Address table entries
  ASDM            ASDM Connections
  SSH             SSH Sessions
  Telnet          Telnet Sessions
  Xlates          XLATE Objects
  All             All Resources
!
class default
  limit-resource IPSec 5
  limit-resource Mac-addresses 65535
  limit-resource ASDM 5
  limit-resource SSH 5
  limit-resource Telnet 5
  limit-resource All 0

```

Figure 6-10 depicts a reference topology conceived for the analysis of resource allocation within FWSM (which is totally analogous on ASA). In the scenario, a packet generator directs connections at a rate of 60 cps toward the host 172.16.242.10 (reachable through context CR1). In Example 6-15, a resource class called **BASIC1** is applied to context CR1. This class defines the absolute limit of 100 for the parameter simultaneous connections.

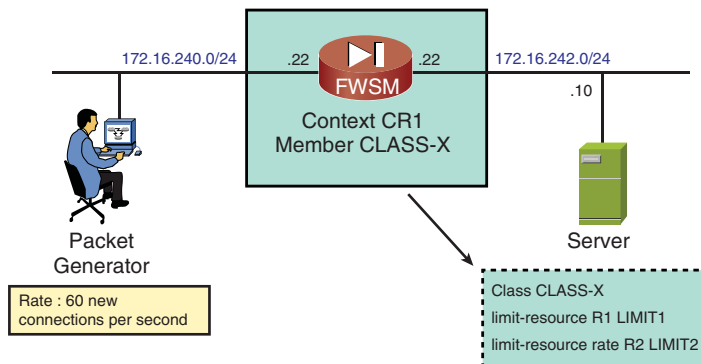


Figure 6-10 *Reference Topology for Resource Allocation Analysis*

Example 6-15 also shows the class **BASIC1** in action:

- When the limit is reached, a Syslog message is visible in the **admin-context**.
- The **show resource usage context** command reveals that the peak value of 100 is reached (which corresponds to the configured limit) and that, as a result, some connections were denied.

Example 6-15 *Sample Resource Class (1)*

```
! Creating a resource-class and assigning to a Context

class BASIC1
  limit-resource Conns 100
!
context CR1
  member BASIC1
!
FWSM# show context CR1
Context Name      Class      Interfaces      Mode      URL
CR1               BASIC1     Vlan1240,Vlan1242  Routed    disk:/CR1.cfg
!
! Resource Class in action (perspective of the admin-context)

%FWSM-5-321001: Resource 'conns' limit of 100 reached for context 'CR1'
!
FWSM# show resource usage context CR1
Resource      Current      Peak      Limit      Denied Context
Conns         0            100       100        24 CR1
Xlates        1            1 unlimited 0 CR1
Hosts         1            1 unlimited 0 CR1
```

Example 6-16 refers to the topology shown in Figure 6-10 and complements what you learned in Example 6-15. An absolute limit of 600 simultaneous connections and an admission rate of 40 connections per second (cps) are configured for class **BASIC2**. The two instances of the **show resource usage context** command characterize the following:

- Before reaching the absolute limit of 600 established connections, the admission rate is held at 40 cps (connections arriving at a higher rate are denied by the 40 cps rule).
- After the threshold of 600 connections is crossed, the absolute limit starts to prevail, and no new connections are admitted for a while. Later on, established connections start to time out, and some others might be accepted again (but now, at a lower rate).

Example 6-16 also documents the kind of data unveiled with the aid of the **show resource allocation detail** command. For each configurable parameter, the limits are shown on a per-class basis, and the percentage of the total number they represent is calculated.

Example 6-16 *Sample Resource Class (2)*

```

! Creating a Resource-class that sets limits for 02 parameters
class BASIC2
    limit-resource rate Conns 40
    limit-resource Conns 600
!
context CR1
    member BASIC2
!
! Connections denied because of exceeding the configured rate of 40 CPS
FWSM# show resource usage context CR1
Resource                Current      Peak      Limit      Denied Context
Conns                  269      269      600      0 CR1
Xlates                   1         1 unlimited  0 CR1
Hosts                    1         1 unlimited  0 CR1
Conns [rate]          40       40       40      120 CR1
Fixups [rate]           40       40 unlimited  0 CR1
!
! Connections denied after exceeding the absolute limit of 600
FWSM# show resource usage context CR1
Resource                Current      Peak      Limit      Denied Context
Conns                  600      600      600      305 CR1
Xlates                   1         1 unlimited  0 CR1
Hosts                    1         1 unlimited  0 CR1
Conns [rate]          21       40       40      280 CR1
Fixups [rate]           21       40 unlimited  0 CR1
!
! Detailing Resource Allocation

FWSM# show resource allocation detail
Resource Origin:
    A      Value was derived from the resource 'all'
    C      Value set in the definition of this class
    D      Value set in default class
Resource      Class      Mmbrs  Origin      Limit      Total      Total %
Conns [rate] default    all     CA unlimited
               BASIC2      1       C         40         40      0.02%
               All Contexts: 3         40      0.02%
[ output suppressed ]
Conns        default    all     CA unlimited
               BASIC2      1       C         600        600      0.06%
               All Contexts: 3         600      0.06%

```

Tip Netflow can be of great value when you need to determine the adequate size of a virtual context. By building a simple Flow Record (for instance, focusing on packets that have the value 0x02 for the TCP Flags field) and enabling it on the Catalyst 6500 interface that leads to a given context, you can have a grasp of how many new connection setups are directed to that context.

Interconnecting Virtual Elements

Now that the basic behavior of virtual elements has been analyzed, it is time to study how they might interconnect.

Interconnecting VRFs with an External Router

The standard method for VRF interconnection is to employ an external routing element. In Figure 6-11, Router R1 acts as such a device. Although each VRF on 6500B points a static route to the aggregate 172.20.0.0/14 via R1, R1 knows the specific networks handled by each VRF.

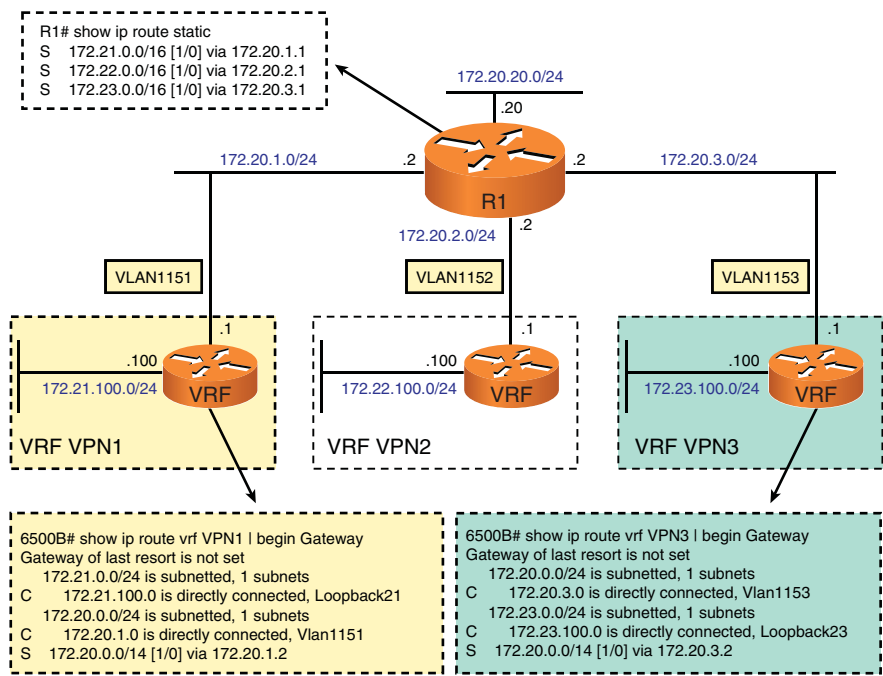


Figure 6-11 Sample VRFs Interconnection Scenario

Example 6-17 refers to the topology of Figure 6-11 and presents the pertinent details:

- The static routes visible on each VRF on 6500B.

- The routing table on R1, pointing, for instance, to gateway 172.20.1.1 to reach the major network 172.20.0.0/16 (under control of VRF VPN1). R1 is not VRF-aware and the VRF VPN1 on 6500B is deemed a regular router.
- A traceroute test originated on VRF VPN1 and destined to host 172.23.100.100, on VRF VPN3. The example shows that R1 (172.20.1.2) is an L3 hop in the path.

Example 6-17 VRF Interconnection Using an External Router

```

! Viewing the Static Routes on 6500B
6500B# show ip route vrf VPN1 static
S    172.20.0.0/14 [1/0] via 172.20.1.2
!
6500B# show ip route vrf VPN2 static
S    172.20.0.0/14 [1/0] via 172.20.2.2
!
6500B# show ip route vrf VPN3 static
S    172.20.0.0/14 [1/0] via 172.20.3.2
!
! R1 interconnects the VRFs of 6500B by means of Static Routes
R1# show ip route | begin Gateway
Gateway of last resort is not set
    172.20.0.0/24 is subnetted, 4 subnets
C        172.20.20.0 is directly connected, Loopback20
C        172.20.1.0 is directly connected, FastEthernet0/0.1151
C        172.20.2.0 is directly connected, FastEthernet0/0.1152
C        172.20.3.0 is directly connected, FastEthernet0/0.1153
S    172.21.0.0/16 [1/0] via 172.20.1.1
S    172.22.0.0/16 [1/0] via 172.20.2.1
S    172.23.0.0/16 [1/0] via 172.20.3.1
!
! Tracing the route to a host accessible though R1 (from VRF 'VPN1')

6500B# traceroute vrf VPN1 ip 172.23.100.100
Tracing the route to 172.23.100.100
  1 172.20.1.2 0 msec 4 msec 0 msec
  2 172.20.3.1 0 msec

```

Interconnecting Two Virtual Contexts That Do Not Share Any Interface

The connection of two virtual contexts that do not share any interface is totally analogous to that of VRFs, just analyzed. Figure 6-12 depicts a network topology in which routing between two contexts of an ASA-based Firewall is achieved with the help of the external router called EXT.

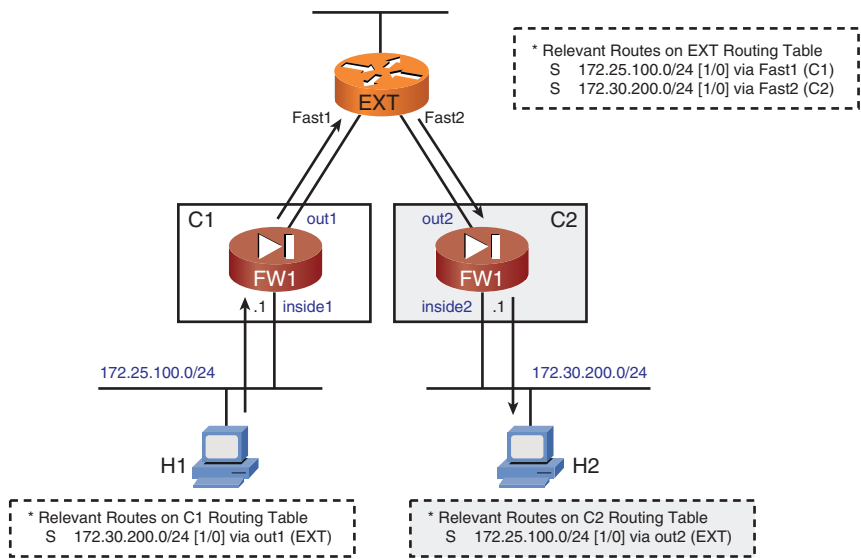


Figure 6-12 *Interconnecting Contexts That Do Not Share Any Interface*

In this scenario, if a packet destined to host H2 reaches the **inside1** interface of context C1, it is routed through EXT. Likewise, if traffic bound to host H1 arrives at **inside2** interface on context C2, it is forwarded via **out2** to the EXT gateway.

Interconnecting Two FWSM Contexts That Share an Interface

Interconnecting two FWSM contexts that share an interface introduces specific challenges. Figure 6-13 shows a reference topology for the analysis of such a situation. In this figure, the Interface VLAN 1242 is simultaneously allocated to routed contexts CR1 and CR2.

Example 6-18 builds upon the scenario in Figure 6-13. R0 is connected to VLAN 1240 on context CR1, and R1 is connected to VLAN 1241 on context CR2. To reach Router R2 (connected to the shared interface on VLAN 1242), R0 uses a static route through CR1, and similarly, R1 uses CR2 as its gateway.

For connections initiating on the outside (VLANs 1240 or 1241), packet classification is done by the FWSM based on source interface. A packet that reaches the FWSM using VLAN 1240 is mapped to CR1 by CLASSIFIER1 because this interface is part of the context. In a totally analogous manner, a packet arriving on the FWSM via VLAN 1241 is assigned to CR2 because this interface belongs to this particular context.

Well, no big issue so far. The challenges come to the scene when a flow is starts in the opposite direction (or in other words, from a host connected to the shared interface on VLAN 1242). Example 6-18 shows that a ping packet from R2 to R1 (172.16.241.10) cannot cross the FWSM because of a failure of CLASSIFIER2 to determine the appropriate security context. This happens because none of the contexts consider themselves as the

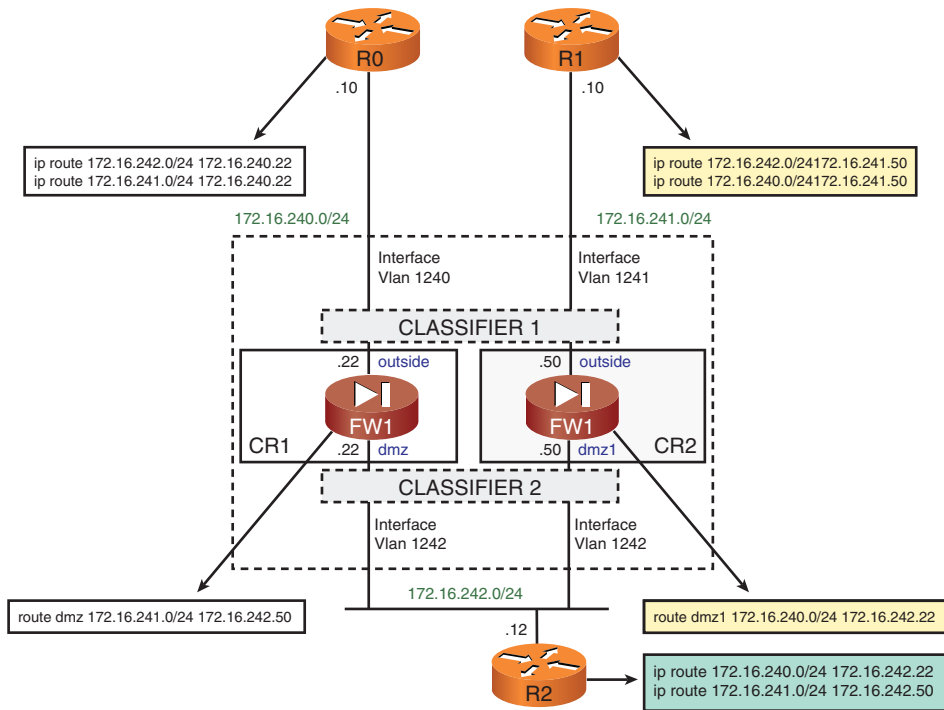


Figure 6-13 *Interconnecting Two FWSM Contexts That Share an Interface*

owner of the destination address, and because VLAN 1242 is common to CR1 and CR2, classification cannot be performed based on source interface.

Example 6-19 also relates to Figure 6-13 and complements what was studied in Example 6-18. In this new scenario, context CR2 employs static NAT to render the address 172.16.241.10 (which exists on the outside) visible on its *dmz1* interface (VLAN 1242). As a result of such an arrangement, CR2 sees itself as the owner of address 172.16.241.10; the packet from R2 to R1 can then flow. (NAT offers a helping hand to the CLASSIFIER2 process.)

Example 6-18 *Packet Classification on FWSM using Source Interface*

```
! The packet from R0 to R2 flows through Context CR1
R0# ping 172.16.242.12 source 172.16.240.10 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
!
FWSM/CR1# %FWSM-6-106100: access-list OUTSIDE permitted icmp
outside/172.16.240.10(0) -> dmz/172.16.242.12(8) hit-cnt 1 (first
hit) [0xec36b045, 0x0]
%FWSM-6-305009: Built dynamic translation from dmz:172.16.242.12 to
outside:172.16.242.12
```

```
%FWSM-6-302020: Built inbound ICMP connection for faddr
172.16.240.10/21 gaddr 172.16.242.12/21 laddr 172.16.242.12/0
!
! The packet from R1 to R2 crosses Context CR2

R1# ping 172.16.242.12 source 172.16.241.10 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
!
FWSM/CR2 # %FWSM-6-106100: access-list OUTSIDE permitted icmp
outside/172.16.241.10(0) -> dmz1/172.16.242.12(8) hit-cnt 1 (first
hit) [0xec36b045, 0x0]
%FWSM-6-305009: Built dynamic translation from dmz1:172.16.242.12 to
outside:172.16.242.12
%FWSM-6-302020: Built inbound ICMP connection for faddr
172.16.241.10/22 gaddr 172.16.242.12/22 laddr 172.16.242.12/0
!
! Ping from R2 (connected to the shared interface) to R1 fails

R2# ping 172.16.241.10 source 172.16.242.12 repeat 1
Success rate is 0 percent (0/1)
!
FWSM/admin # %FWSM-6-106025: Failed to determine security context for
packet: vlan1242 icmp src 172.16.242.12/2048 dest 172.16.241.10/16157
```

Example 6-19 Packet Classification on FWSM Using NAT

```
! Creating an Static NAT entry on Context CR2
static (outside,dmz1) 172.16.241.10 172.16.241.10 netmask 255.255.255.255
!
! Ping from R2 to the address published by Context CR2 is successful

R2# ping 172.16.241.10 source 172.16.242.12 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
!
FWSM/CR2 # %FWSM-6-305009: Built static translation from outside:172.16.241.10
to dmz1:172.16.241.10
%FWSM-6-106100: access-list DMZ1 permitted icmp dmz1/172.16.242.12(0) ->
outside/172.16.241.10(8) hit-cnt 1 (first hit) [0xb5e41d5b, 0x0]
%FWSM-6-305009: Built dynamic translation from dmz1:172.16.242.12 to
outside:172.16.242.12
%FWSM-6-302020: Built outbound ICMP connection for faddr
172.16.241.10/42 gaddr 172.16.242.12/42 laddr 172.16.242.12/8
```

Although the NAT configuration of the previous example minimized the problem shown in Example 6-18, it does not eliminate all the potential issues. With that said, some summarizing notes might help to better understand the FWSM behavior:

- The FWSM uses the same global MAC address across all its interfaces. (To see this address, refer to the **show interface** output in Example 6-11.) Example 6-18 shows that the challenges derive from interface sharing. How can a router direct packets to IP addresses on a given network if multiple IP addresses map to the same MAC address? The common MAC address would also constantly move from one VLAN interface to another, therefore affecting the bridging table on the Catalyst switch.
- If a packet arrives on an interface that belongs to more than one context, the classifier performs a lookup on the destination address and, therefore, must have knowledge about the subnets located behind each virtual context. To accomplish this task, the classifier relies on active NAT sessions (rather than on the routing table). As shown on Example 6-18 (for the scenario of Figure 6-13), the static routes configured on R2 were not enough. Only after inserting an appropriate NAT statement, R2 succeeded to send a packet to R1 through the FWSM.
- A packet sourced from the dmz interface address on context CR1 (172.16.242.22) and directed to the dmz1 interface address on context CR2 (172.16.242.50) never leaves the device. For example, even though a ping between the FWSM addresses on VLAN 1242 succeeds, the packets are not seen on the receiving side.
- Figure 6-13 shows that R0 points a route to CR1 to reach R1 (172.16.241.10/24) and that R1 relies on CR2 to reach R0 (172.16.240.10/24). The figure also reveals that the corresponding routes to R1 and R0 have been configured on CR1 and CR2 (always using as the gateway the address on the other side of the shared interface). In spite of all these provisions, CR2 cannot reach R0 through CR1. As a result, R1 cannot contact R0 via CR2. Likewise, R0 cannot reach R1 through CR1. This happens because the FWSM does not succeed in classifying packets originated from one side of the shared interface and destined to the other.
- When the shared interface connects to the outside (typically the Internet), destination addresses on the inside are limited and well known. Configuring NAT to interconnect contexts in this scenario is feasible. On the contrary, sharing an inside interface might render the NAT workaround unmanageable because the outside addresses are unlimited.
- Unique interfaces for contexts are required when operating in Transparent mode. Source interface is always used as the classification criterion.

Note The CLASSIFIER1 and CLASSIFIER2 blocks were used just to represent (in a single figure) that classification can be done based on source interface or destination address. Actually, there is a single classifier software module.

Note NAT functionality for ASA-based Firewalls is thoroughly discussed in Chapter 8, “Through ASA Using NAT.”

Interconnecting Two ASA Contexts That Share an Interface

One special feature that gives an advantage to ASA for interface sharing, when compared with FWSM, is the possibility of statically configuring the MAC address on each context for the common interface. This is accomplished using the `mac-address` command at the interface level. Another possible way to eliminate the problem is to use the `mac-address auto` command at the global level on the system execution space.

Figure 6-14 is a close ASA version of the scenario previously studied in Figure 6-13. In the topology, interface G0/1.1222 is common to contexts LAB5 and LAB6.

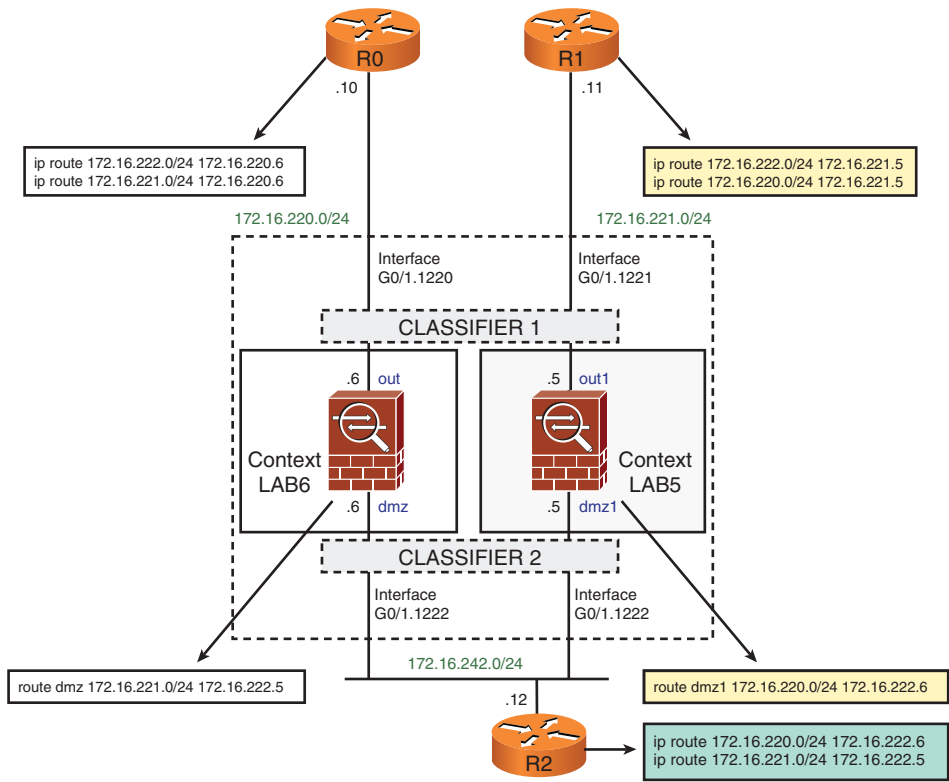


Figure 6-14 Interconnecting Two ASA Contexts That Share an Interface

Example 6-20 refers to Figure 6-14 and shows that the last four HEX digits of the original MAC address of G0/1.1222 were respectively changed to 5555 and 6666 on contexts

LAB5 and LAB6. Some remarkable differences when comparing to FWSM behavior follow:

- A ping from LAB6 interface address (172.16.222.6) successfully reaches 172.16.221.11 (R1), crossing context LAB5.
- A ping from 172.16.220.10 (R0) uses LAB6 as the gateway and reaches 172.16.221.11 (R1) after traversing context LAB5.
- You can even perform a traceroute to 172.16.221.11 and further characterize the sequence of Layer 3 hops to reach this destination. The details on how to enable ASA to display itself as a hop during traceroute tasks is covered in Chapter 7, “Through ASA Without NAT.”

The various tests documented in Example 6-20 demonstrate that the selection of a unique MAC address is the most effective method to avoid the issues associated with interface sharing between Virtual Contexts.

Example 6-20 *Packet Classification on ASA Using Static MAC Addresses*

```
! Relevant Configuration on Context LAB5
interface GigabitEthernet0/1.1221
  nameif out1
  security-level 0
  ip address 172.16.221.5 255.255.255.0
!
interface GigabitEthernet0/1.1222
  mac-address 0017.9527.5555
  nameif dmz1
  security-level 50
  ip address 172.16.222.5 255.255.255.0
!
route dmz1 172.16.220.0 255.255.255.0 172.16.222.6 1
!
! Relevant Configuration on Context LAB6

interface GigabitEthernet0/1.1220
  nameif out
  security-level 0
  ip address 172.16.220.6 255.255.255.0
!
interface GigabitEthernet0/1.1222
  mac-address 0017.9527.6666
  nameif dmz
  security-level 50
  ip address 172.16.222.6 255.255.255.0
```

```

!
route dmz 172.16.221.0 255.255.255.0 172.16.222.5 1
!
! Displaying the ARP Tables on each Context

ASA2/LAB5# show arp | include dmz1
      dmz1 172.16.222.12 000f.3461.9621 9476
      dmz1 172.16.222.6 0017.9527.6666 9476
!
ASA2/LAB6# show arp | include dmz
      dmz 172.16.222.12 000f.3461.9621 9554
      dmz 172.16.222.5 0017.9527.5555 9554
!
! Despite the PING success, no packet is seen on Context LAB5

ASA2/LAB6# ping 172.16.222.5 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 1/1/1 ms
!
! Ping from LAB6 interface address successfully crosses Context LAB5

ASA2/LAB6# ping 172.16.221.11 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 1/1/1 ms
%ASA-6-302020: Built outbound ICMP connection for faddr
172.16.221.11/0 gaddr 172.16.222.6/4388 laddr 172.16.222.6/4388
!
ASA2/LAB5 # %ASA-6-106100: access-list DMZ1 permitted icmp
dmz1/172.16.222.6(8) -> out1/172.16.221.11(0) hit-cnt 1 first hit
[0x1a3a6847, 0x0]
%ASA-6-302020: Built outbound ICMP connection for faddr
172.16.221.11/0 gaddr 172.16.222.6/4388 laddr 172.16.222.6/4388
!
! Ping from R0 to R1 traverses contexts LAB6 and LAB5, in this order

R0# ping 172.16.221.11 source 172.16.220.10 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
!
ASA2/LAB6 # %ASA-6-106100: access-list OUT permitted icmp
out/172.16.220.10(8) -> dmz/172.16.221.11(0) hit-cnt 1 first hit
[0xb4faa0af, 0x0]
%ASA-6-302020: Built inbound ICMP connection for faddr
172.16.220.10/60 gaddr 172.16.221.11/0 laddr 172.16.221.11/0
!
ASA2/LAB5 # %ASA-6-302020: Built outbound ICMP connection for faddr
172.16.221.11/0 gaddr 172.16.220.10/60 laddr 172.16.220.10/60
!

```

```
! Tracing the route to R1 (from R0)
```

```
R0# traceroute 172.16.221.11 source 172.16.220.10 probe 1
```

```
Tracing the route to 172.16.221.11
```

```
 1 172.16.222.6 4 msec
 2 172.16.221.5 0 msec
 3 172.16.221.11 4 msec
```

Issues Associated with Security Contexts

The various sections about security contexts have shown that this is a powerful tool for organizations that want to make a better use of resources. Nonetheless, some features are still not available for multiple mode operation. For the FWSM, a product that was conceived to be a *firewall-only* device, the issues are not so noticeable:

- FWSM security contexts support only static routes and BGP stub routing. The IGP's are not available.
- Multicast routing is not supported. This means that a FWSM context cannot act, for instance, as a Protocol Independent Multicast (PIM) routing node. Notwithstanding, multicast bridging is still possible.

Note The FWSM was one among many service modules launched for the Catalyst 6500 product line. For instance, there are modules that specialize in VPN and IDS/IPS features; therefore, if these services were needed, the assumption was that the pertinent modules would be deployed.

For ASA multifunctional security appliances, the current limitations are more visible, but it does not mean that this will remain unchanged on future OS versions:

- Static routes are the only option. No dynamic routing protocols are supported.
- Multicast routing is not supported. Multicast bridging is still available.
- VPN termination inside a context (either IPsec or SSL) is not supported.
- Threat detection is not supported. (This resource is analyzed in Chapter 11, “Additional Protection Mechanisms.”)

Note It is advisable to stay tuned to the roadmap of ASA appliances. As new software releases become available, some of the original restrictions concerning multiple context operation might be lifted.

Complete Architecture for Virtualization

Now that you have learned the main details about individual virtual elements, it is time to examine some scenarios that can serve as the base for broader virtualization solutions. This section even proposes a combination of virtualization components with the aim to achieve an end-to-end architecture for virtualization.

Virtualized FWSM and ACE Modules

The Cisco Application Control Engine (ACE) module is a natural companion for the FWSM within the scope of Data Center virtualization. In a typical deployment, the FWSM offers firewall services, whereas the ACE module is in charge of Server Load Balancing (SLB) tasks.

Figure 6-15 highlights the hierarchical nature of VLAN allocation for service modules and contexts on a Catalyst 6500. Initially, a group of VLANs is taken out of the 6500 pool (which contains approximately 4000 VLANs) and allocated to each physical module. In the sequence, subpools of VLANs are assigned to the appropriate virtual contexts.

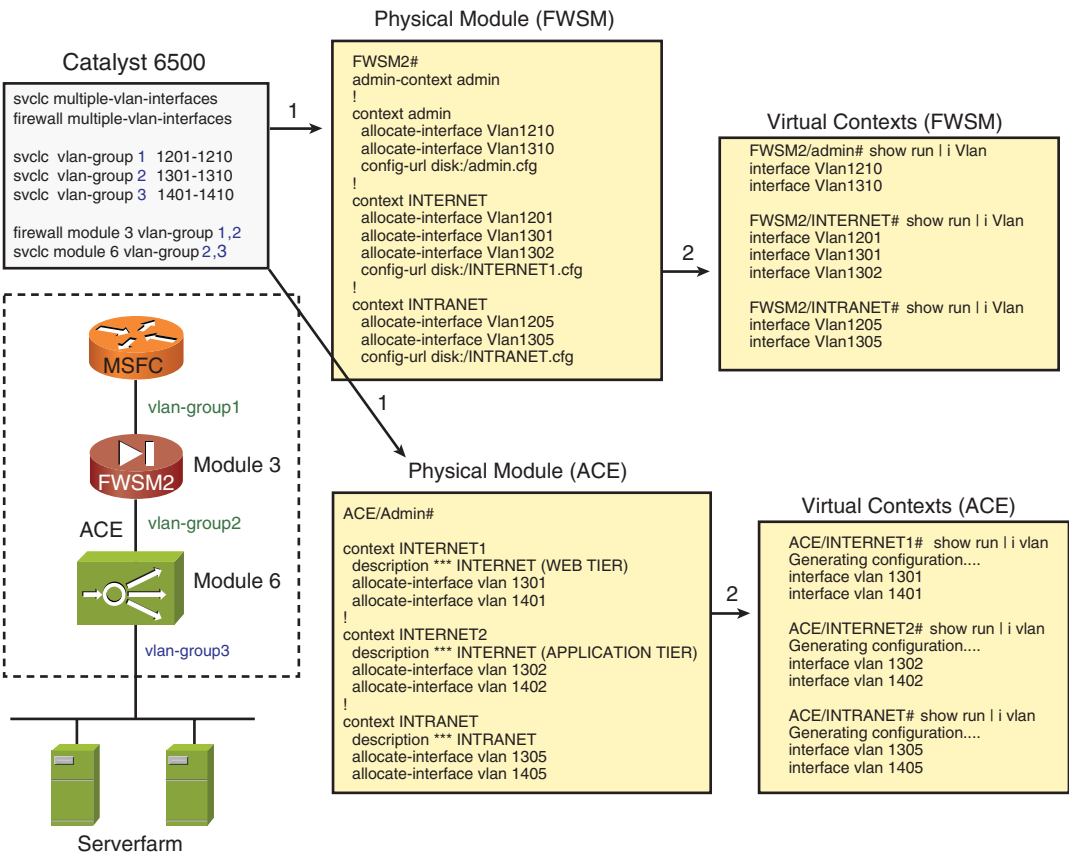


Figure 6-15 VLAN Allocation for Virtual Contexts
<https://t.me/learningnets>
www.it-ebooks.info

Figure 6-15 proposes a VLAN allocation scheme for a Catalyst 6500 that hosts a FWSM and an ACE module. The figure emphasizes that it is convenient in practical usage scenarios to clearly define (whenever possible) not only a structured distribution of IP subnets but also predictable VLAN numbering rules:

- VLANs on the 12XX range (**vlan-group 1**) were reserved for connecting the Catalyst 6500 MSFC to the outside interfaces on the FWSM.
- VLANs on the 13XX space (**vlan-group 2**) were reserved for connecting the FWSM inside interfaces to the ACE modules outside interfaces.
- VLANs belonging to **vlan-group 3** (14XX range) are assigned to the server farms handled by the ACE module.

The profits that derive from investing time and effort on such an approach are visible on daily maintenance (easier troubleshooting because the layer on which the VLAN resides is quickly identified using the VLAN number), and provisioning. (Even if you do not need many contexts initially, a proper numbering model is ready for usage when necessary.)

Figure 6-16 further details the allocation scheme proposed in Figure 6-15 by showing two virtualized structures named INTERNET and INTRANET. (Each of them uses regular contexts on the ACE and FWSM and dedicated VRFs on the MSFC). The management structure uses the **admin-context** on the service modules and the Global Table in the MSFC.

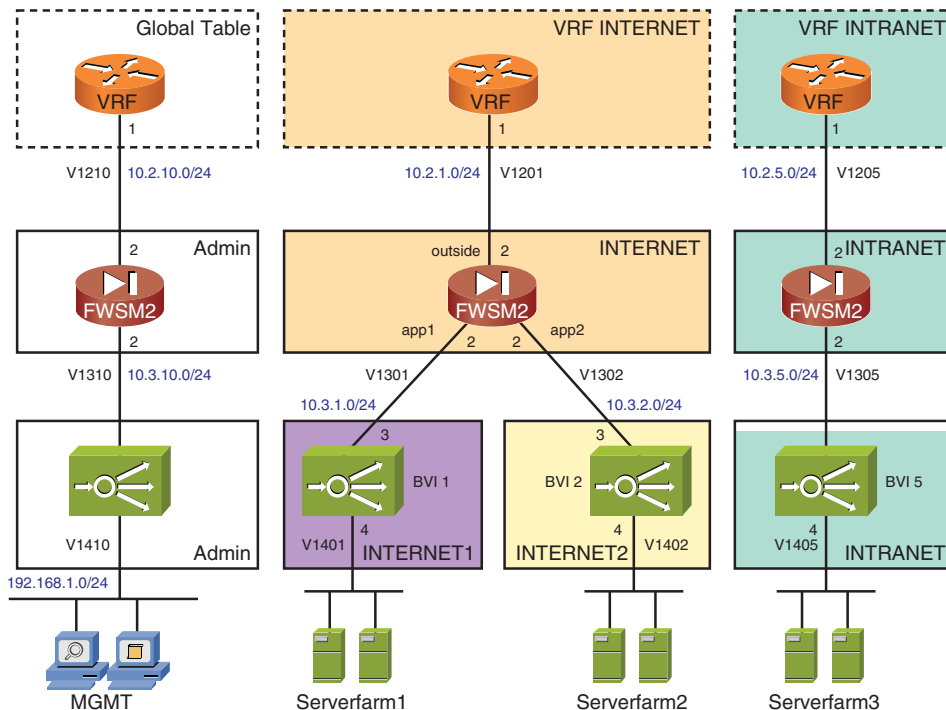


Figure 6-16 Sample Virtualization Scenario with VRFs and Virtual Contexts

Segmented Transport

Figure 6-17 depicts a network virtualization scheme designed to enable the segmentation of user group traffic all the way from the network ingress point up to the Data Center:

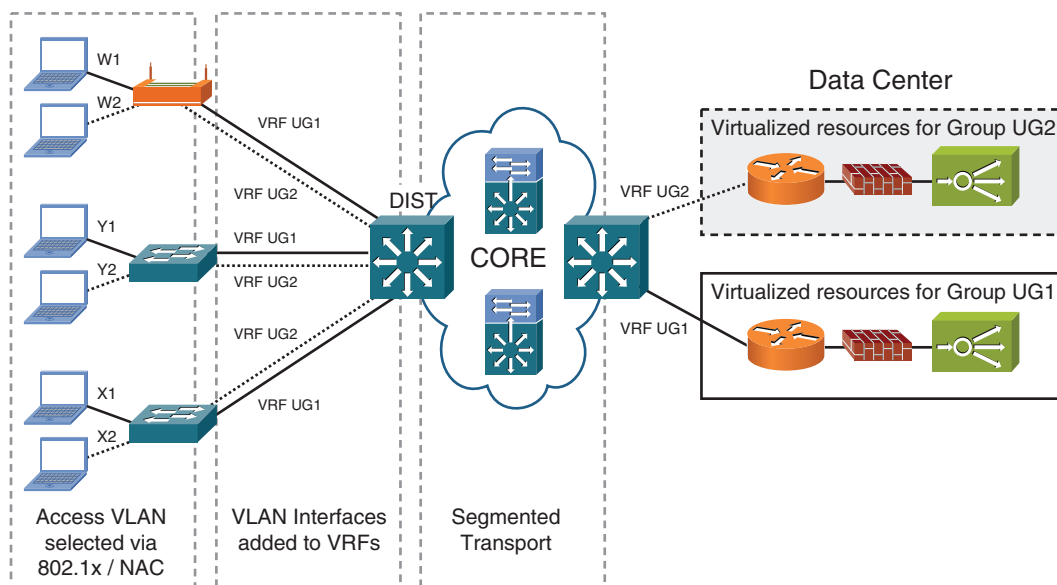


Figure 6-17 *Segmented Transport from Access to Data Center*

- On the access switches (or Wireless Access Points), the appropriate VLAN is assigned to a port, based on the role of the user requesting access. This user profile validation is accomplished using mechanisms such as 802.1X or Network Admission Control (NAC). The VLAN assignment ideally reflects the privilege levels needed for users in a group to perform their duties.
- In the scenario, the access devices connect to the distribution switches via 802.1Q trunks that transport the user VLANs (dynamically assigned to user ports but pre-provisioned on trunks).
- The VLAN Interface (L3 counterpart of the L2 VLAN) is associated with the VRF that defines the user group. For the sake of simplicity, only two user groups are shown.
- The VRFs are transported in a segmented fashion over the CORE using technologies such as MPLS-VPN or GRE tunnels (bound to VRFs).
- Each **user-group** has access to a single virtualized portion of the Data Center, possibly materialized using VRFs, FWSM, and ACE contexts. (Refer to Figure 6-16 for a sample arrangement of these virtual elements.)

Virtual Machines and the Nexus 1000V

Server-side virtualization has been around for quite a while. A well-known solution is brought by VMWARE using the ESX platform. This classic solution is now enabled to go one step further, after the advent of the Cisco Nexus 1000V virtual switch.

Figure 6-18 shows a high-level overview of the relationship between the Cisco Nexus 1000V and VMWARE ESX:

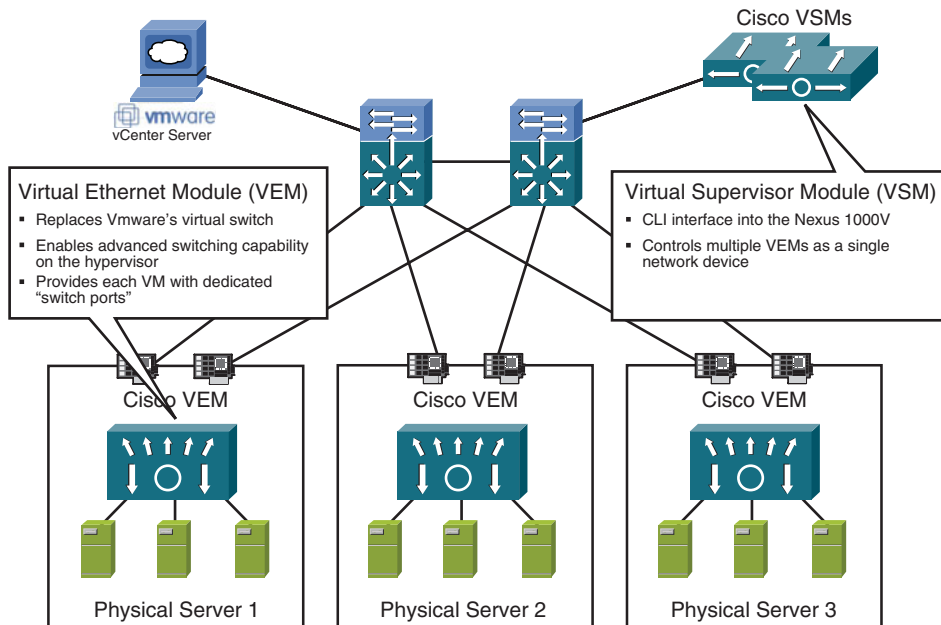


Figure 6-18 Overview of the Nexus 1000V Virtual Switch

- The Virtual Ethernet Module (VEM) is installed in each physical server running ESX and replaces the VMWARE's virtual switch. This piece of software plays the role of an interface module on a modular LAN switch.
- Each virtual machine inside a physical server is treated as if connected to a port of the Virtual Ethernet Module.
- The Virtual Supervisor Module is an external management solution that acts as the supervisor of a modular LAN switch.

This add-on to the original ESX solution has a lot of management and security features that were previously available solely on physical switches. With Nexus 1000V, you can now allow the following on a per-VM basis:

- Port security
- DHCP snooping

- Dynamic ARP inspection
- IP Source Guard
- Private VLANs
- Port mirroring (SPAN)
- Netflow v9
- QoS features such as 802.1p CoS marking and rate limiting

This was just a quick reference to the Nexus 1000V. At this point, if you already use (or plan to use) the VMWARE ESX platform, it is probably worth it to get more details about this powerful solution.

Another interesting exercise is to try to connect the dots between the solutions outlined in this last section to visualize an example of the end-to-end virtualization architecture.

Summary

This chapter described how the important trend of virtualization is materialized within Cisco Firewall products and solutions. The most important topics covered in this chapter follow:

- How VLANs and VRFs apply to the virtualization of the Data Plane
- The definition of security contexts and how to allocate resources to them
- How to interconnect virtual elements
- A sample way to implement an end-to-end network virtualization architecture

This chapter brings the second part of the book to an end. After completing the study of the fundamentals, the next part focuses on the main security features. You can find more details about the insertion of firewalls in virtualized environments in Chapter 17, “Firewall Interactions,” which is dedicated to security design.

Through ASA Without NAT

This chapter covers the following topics:

- Types of access through the firewall
- Additional thoughts about security levels
- ICMP connection examples
- UDP connection examples
- TCP connection examples
- Same security access
- Handling ACLs and Object-Group

“When liberty destroys order, the hunger for order will destroy liberty.” —Will Durant

After getting acquainted with the tools that help you understand the behavior of firewall functionality and with the various connectivity models, it is time to turn the attention to the security features. This chapter studies the establishment of *connections* through firewalls that are based on the Adaptive Security Algorithm (ASA family), in situations that do not involve Network Address Translation (NAT), which is the object of analysis in Chapter 8, “Through ASA Using NAT.”

In the context of computer networks, a firewall is a security system that acts as a Policy Enforcement Point between pairs of networks to which it connects. The firewall can only control traffic that passes *through* it, therefore turning itself into a sort of reference point in the network topology, isolating the interface that should be treated as *trusted* from the other interfaces classified as *untrusted* (or less trusted).

Following this initial state of isolation created by the firewall, the administrator can define rules to selectively allow specific protocols (*to* and *from* a given set of hosts) through the firewall, meaning that this device works as a kind of *conditional gateway*.

The conditions to permit traffic are normally defined in the *firewall policy* and ideally should materialize what is stated in the *security policy* of the organization.

To reflect this degree of trustworthiness of a given firewall interface, Cisco has used, since the early days of the PIX Firewalls, the concept of *Security Level (sec-lvl)* in the ASA algorithm. This parameter assumes values in the range 0 to 100, where the highest possible value, 100, is used for the most trusted network under the firewall control, which is, by default, called *inside*. On the other hand, the name *outside* is reserved for the least trusted network (frequently the connection to the Internet) and, by default, the value 0 (zero) is assigned to its *sec-lvl*.

Keep in mind that the *sec-lvl* parameter assumes relative values, defined by the firewall administrator in direct correspondence with the perceived degree of trust of a given interface. Although the names *inside* and *outside* are used by default, they can be modified manually.

Types of Access Through ASA-Based Firewalls

With the advent of PIX/ASA release 7.0, there are three possible categories of access through the firewall: *outbound*, *inbound*, and *same security access*. Each of these is presented in great detail in the discussion that follows. Never lose sight that throughout this chapter the discussion is centered around the **no nat-control** model, in which NAT is not mandatory.

- **Outbound access:** This term is used for connections through the firewall in which the ingress interface of the traffic is more trusted than the egress interface. In this case, the connection is initiated on a high security level interface and ends on a lower *sec-lvl* interface. Classic examples of this type of access are connections initiating on the inside and going to the outside or to a dmz. ASA appliances enable, by default, without the need of any ACL, the initiation of a connection from a high to a lower *sec-lvl* interface. On the other hand, the Firewall Services Module (FWSM) behaves differently, requiring manual definition of the rules to enable outbound access.
- **Inbound access:** Conversely, when a certain element tries to initiate a connection through the firewall to a network resource that resides on an interface with a higher *sec-lvl* than the ingress interface, it represents an instance of inbound access. Access rules to explicitly enable the initiation of a certain protocol connection from the lower *sec-lvl* interface are a must in this case.
- **Same security access:** The default behavior of ASA-based Firewalls is to completely block connections between any two interfaces that have the same configured value for the *sec-lvl* parameter. To instruct ASA to start enabling this category of access, you need to use the **same-security-traffic permit** commands. No ACLs are required for connection initiation. Notwithstanding, ACLs can be used to restrict what combinations of addresses and services are enabled between the two given interfaces.

ASA ACLs have an *implicit deny* at their end. This way, if only a few specific types of connections need to be enabled between the interfaces, it is convenient to start the ACL

construction using **permit** statements that define the interesting traffic and deny everything else (as a result of the default deny approach).

Figure 7-1 reveals the implicit rules for an ASA device configured with the security levels defined in Example 7-1 and inserted in the reference topology of Figure 7-2. These special rules are further detailed within Examples 7-3 and 7-4.

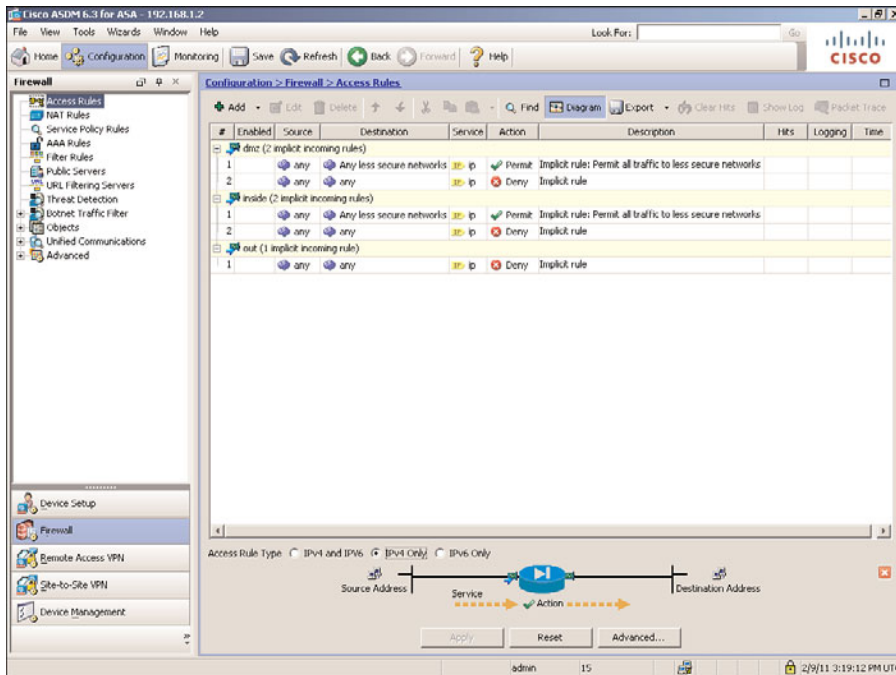


Figure 7-1 ASA Implicit Rules

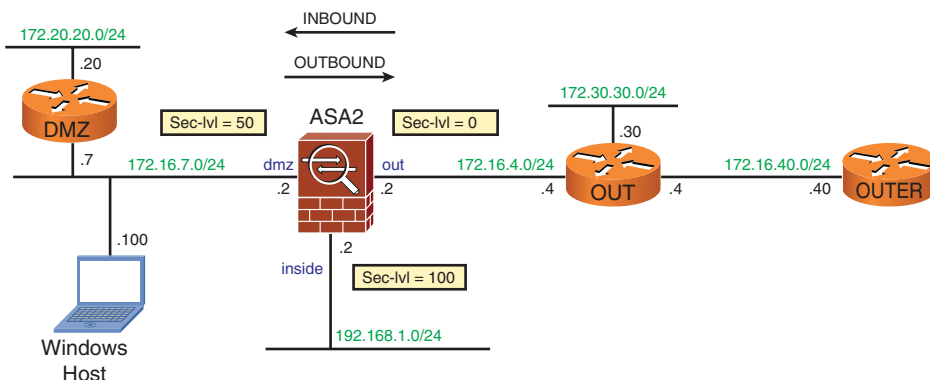


Figure 7-2 Reference Network Topology

Note All the Access Control Lists used in this chapter are interface-specific ACLs. The Global ACLs introduced in ASA release 8.3 are examined in Appendix A, “NAT and ACL changes in ASA 8.3.”

Example 7-1 *Security Levels*

ASA2# show nameif		
Interface	Name	Security
Vlan4	out	0
Vlan7	dmz	50
Vlan100	inside	100

Example 7-2 characterizes a situation in which the **no nat-control** model is in place. An ERROR message is issued when the **show nat dmz** command is issued. This is because there is no NAT Rule configured. (This last command is important and discussed further in Chapter 8).

Example 7-2 *No nat-control model*

```
! Verifying nat-control configuration
ASA2# show running-config nat-control
no nat-control
!
! Viewing NAT policies for a given interface
ASA2# show nat dmz
ERROR: No matching NAT policy found
```

Example 7-3 applies the Packet Tracer tool to illustrate the *implicit permit* behavior for outbound connections on ASA. Its counterpart is Example 7-4, which documents the *implicit deny* rule for inbound traffic.

Example 7-3 *Packet Tracer for Outbound TCP (Implicit Permit Rule)*

```
ASA2# packet-tracer input dmz tcp 172.16.7.7 2000 172.16.4.4 23
Phase: 1
Type: FLOW-LOOKUP
Subtype:
Result: ALLOW
Config:
Additional Information:
Found no matching flow, creating a new flow
[ output suppressed ]
Phase: 5
Type: FLOW-CREATION
```

```

Subtype:
Result: ALLOW
Config:
Additional Information:
New flow created with id 518, packet dispatched to next module
[ output suppressed ]
Action: allow
%ASA-6-302013: Built outbound TCP connection 518 for out:172.16.4.4/23
(172.16.4.4/23) to dmz:172.16.7.7/2000 (172.16.7.7/2000)

```

Example 7-4 Packet Tracer for Inbound TCP (Implicit Deny Rule)

```

ASA2# packet-tracer input out tcp 172.16.4.4 2000 172.16.7.7 23
Phase: 1
Type: FLOW-LOOKUP
[ output suppressed ]
Found no matching flow, creating a new flow
Phase: 2
Type: ROUTE-LOOKUP
[ output suppressed ]
in 172.16.7.0 255.255.255.0 dmz

%ASA-2-106001: Inbound TCP connection denied from 172.16.4.4/2000 to 172.16.7.7/23
flags SYN on interface out

Phase: 3
Type: ACCESS-LIST
Subtype:
Result: DROP
Config:
Implicit Rule
Additional Information:
[ output suppressed ]
Drop-reason: (acl-drop) Flow is denied by configured rule

```

Example 7-5 registers the logging messages associated with inbound connections that were denied by the implicit rules. Notice that six of the Netflow *key-fields* appear in the TCP and UDP sample connections. (The ToS field is not used.)

Example 7-6 shows a connection attempt between two interfaces that were assigned equal security levels. The Packet Tracer tool can confirm that the connection was denied by an implicit rule.

Example 7-5 *Sample Inbound Traffic Denied by Implicit Rules*

```

! Sample ICMP Connection attempts denied by implicit rules
%ASA-3-106014: Deny inbound icmp src out:172.16.4.4 dst dmz:172.16.7.7 (type 8,
code 0)
%ASA-3-106014: Deny inbound icmp src out:172.16.4.4 dst dmz:172.16.7.7 (type 3,
code 3)
!
! Sample UDP Connection attempts denied by implicit rules
%ASA-2-106006: Deny inbound UDP from 172.16.4.4/54459 to 172.16.7.7/2055 on
interface out
!
! Sample TCP Connection attempts denied by implicit rules
%ASA-2-106001: Inbound TCP connection denied from 172.16.4.4/58370 to
172.16.7.7/23 flags SYN on interface out

```

Example 7-6 *Default Behavior for Interfaces with Same Security Level*

```

! Interfaces 'out' and 'dmz' now have the same security level
ASA2# show nameif

```

Interface	Name	Security
Vlan4	out	50
Vlan7	dmz	50

```

!
! Host named DMZ (172.16.7.7) attempts a Telnet to 172.16.4.4
DMZ# telnet 172.16.4.4
Trying 172.16.4.4 ...
% Connection timed out; remote host not responding
!
%ASA-2-106001: Inbound TCP connection denied from 172.16.7.7/28674 to
172.16.4.4/23 flags SYN on interface dmz
!
! Confirming the result with the Packet-tracer tool
ASA2# packet-tracer input dmz tcp 172.16.7.7 2000 172.16.4.4 telnet
Phase: 1
Type: FLOW-LOOKUP
[output suppressed]
Phase: 3
Type: ACCESS-LIST
Subtype:
Result: DROP
Config:
Implicit Rule
[output suppressed]
Action: drop
Drop-reason: (acl-drop) Flow is denied by configured rule

```

Additional Thoughts About Security Levels

It is common to see customers ask a question about the need and importance of associating security levels with firewall interfaces, mainly because this concept is not frequent on other vendors' products. It is not worth it to promote the philosophical fight that comes from individual perceptions, but perhaps some insight into situations in which the concept is useful might motivate you to invest the time to fully understanding it.

In its original conception, the *Adaptive Security Algorithm (ASA)* determined that no connection could be set up between interfaces with equal security level. This somewhat resembles the Electricity Theory because no electrical current can flow between two points of a circuit that have the same value of electrical potential. Some practical usage scenarios are described next with the aim to facilitate the learning process.

Figure 7-3 portrays two simple firewall topologies that highlight the usability of the security level concept. In these two classic scenarios, the *sec-lvl* greatly simplifies the firewall policy implementation.

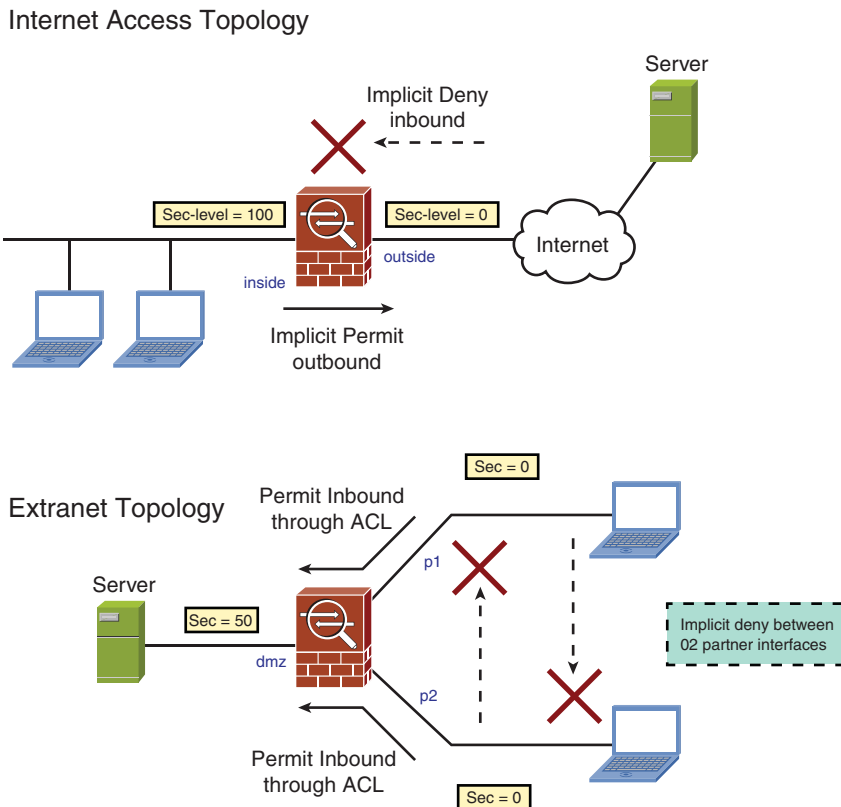


Figure 7-3 Sample Firewall Topologies

Internet Access Firewall Topology

For the simplest firewall topology known as *Internet Access*, in which internal users are provided with access to the Internet, and no internal network resource is available to external users, the concept of security level makes the initial setup easy. In the old PIX models, there was one inside interface (with the **sec-lvl** equal to 100 by default) and one outside interface (**sec-lvl** = 0 by default). In this way, the reference inside/outside (or trusted/untrusted) was automatically created, and the internal users could initiate connections to the outside world without the need to configure any ACL. Further, there was no need to worry about the inbound connections because no connection would be enabled from the untrusted interface.

Extranet Topology

Consider an Extranet topology in which several business partners connect to different firewall interfaces with the following requirements:

- No access exists between any two partner interfaces.
- From every partner interface, you can establish a controlled connection to an internal network where some services reside.

A simple manner of materializing this scenario is to define the security level for all the partner interfaces with the value **X** and the **sec-lvl** for the *services* interface as **Y**, in such a way that $X < Y$.

Using such an arrangement, no communication would be enabled between any pair of partner interfaces, even with **permit ip any any** ACLs. This can save work. Further, considering that the *services* network is on a higher **sec-lvl**, any traffic from a given partner interface would need to be explicitly enabled by means of ACLs.

Isolating Internal Departments

A similar scenario is to replace the partner interfaces described in the Extranet topology with internal departments (engineering, marketing, and sales, for instance) that need to access common services (not tied to any particular department) but are not allowed to share resources directly.

ICMP Connection Examples

This section uses the reference topology of Figure 7-2 to analyze some examples of ICMP connections through ASA. Besides the regular logging messages, the **debug icmp** trace was turned on to facilitate understanding.

Outbound Ping

Example 7-7 shows that, although the *ICMP echo request* was enabled by the **implicit permit** rule for outbound, the expected *echo reply* was dropped by the **implicit deny** for inbound. This happens because ICMP, in nature, is not a Stateful Protocol, and then the echo reply message is not tied to the original echo request.

Example 7-8 eliminates the problem faced in Example 7-7 by creating an explicit *Access Control Entry* to allow the inbound echo replies through. Two ICMP connections are involved in this example:

- **Outbound Connection:** Represented by the echo request and enabled by the implicit permit rule.
- **Inbound Connection:** Corresponding to the echo reply and made possible by means of an explicit rule that is part of the ACL named OUT.

Example 7-7 Initial Ping Test with Implicit Rules

```
! DMZ sends an ICMP Echo Request to OUT (172.16.4.4) but does not receive a Reply
DMZ# ping 172.16.4.4 source 172.16.7.7 repeat 1
Success rate is 0 percent (0/1)
!
! ASA2 builds outbound ICMP connection, allowing the echo request
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.4.4/0 gaddr
172.16.7.7/6 laddr 172.16.7.7/6
ICMP echo request from dmz:172.16.7.7 to out:172.16.4.4 ID=6 seq=0 len=72
!
! OUT receives the Echo Request and sends back the Echo Reply
OUT# ICMP: echo reply sent, src 172.16.4.4, dst 172.16.7.7
!
! Echo reply is denied by ASA (ICMP not inherently stateful and no ACL configured)
%ASA-3-106014: Deny inbound icmp src out:172.16.4.4 dst dmz:172.16.7.7 (type 0,
code 0)
%ASA-6-302021: Teardown ICMP connection for faddr 172.16.4.4/0 gaddr 172.16.7.7/6
laddr 172.16.7.7/6
```

Example 7-8 Modifying Example 7-7 to Allow Echo Replies

```
! ACL defined and associated to the out interface with the 'access-group' command
ASA2# show running-config access-list
access-list OUT extended permit icmp host 172.16.4.4 host 172.16.7.7 echo-reply
ASA2# show running-config access-group
access-group OUT in interface out
!
! DMZ host now receives the Echo Reply from 172.16.4.4
```

```
DMZ# ping 172.16.4.4 source 172.16.7.7 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
DMZ# ICMP: echo reply rcvd, src 172.16.4.4, dst 172.16.7.7
!
! Echo reply (inbound ICMP) now permitted by ASA ACL
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.4.4/0 gaddr
172.16.7.7/7 laddr 172.16.7.7/7
ICMP echo request from dmz:172.16.7.7 to out:172.16.4.4 ID=7 seq=0 len=72
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.4.4/0 gaddr
172.16.7.7/7 laddr 172.16.7.7/7
ICMP echo reply from out:172.16.4.4 to dmz:172.16.7.7 ID=7 seq=0 len=72
```

Example 7-9 reveals the existence of a sequence number for each packet that belongs to an ICMP connection. Although there are ten ping packets crossing the ASA, they are all part of the same flow, and a single pair of connections (one outbound and one inbound) was created. The connection ID is 8 and the sequence numbers range from 0 to 9.

Example 7-9 ICMP ID and Sequence Number

```
DMZ# ping 172.16.4.4 source 172.16.7.7 repeat 10
Success rate is 100 percent (10/10), round-trip min/avg/max = 1/2/4 ms
!
! ASA treats the 10 Pings as a pair of connections (Request outbound, Reply
inbound)
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.4.4/0 gaddr
172.16.7.7/8 laddr 172.16.7.7/8
ICMP echo request from dmz:172.16.7.7 to out:172.16.4.4 ID=8 seq=0 len=72
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.4.4/0 gaddr
172.16.7.7/8 laddr 172.16.7.7/8
ICMP echo reply from out:172.16.4.4 to dmz:172.16.7.7 ID=8 seq=0 len=72
ICMP echo request from dmz:172.16.7.7 to out:172.16.4.4 ID=8 seq=1 len=72
ICMP echo reply from out:172.16.4.4 to dmz:172.16.7.7 ID=8 seq=1 len=72
[ output suppressed ]
ICMP echo request from dmz:172.16.7.7 to out:172.16.4.4 ID=8 seq=9 len=72
ICMP echo reply from out:172.16.4.4 to dmz:172.16.7.7 ID=8 seq=9 len=72
```

Example 7-10 shows a manner of forcing ICMP to behave as a stateful protocol when flowing through ASA. This is achieved with the addition of the **inspect icmp** statement to the global inspection policy. By using this approach, the ASA algorithm enforces that there is a single return packet corresponding to each request and ensures that the sequence numbers in the request and response packets match. This method of allowing ICMP through ASA is more secure than its stateless counterpart.

Example 7-10 *Instructing ASA to Treat ICMP as a Stateful Protocol*

```

! Configuring Stateful Inspection of ICMP within the default global policy
policy-map global_policy
  class inspection_default
    inspect icmp
!
service-policy global_policy global
!
! DMZ host sends an echo request to OUT (172.16.4.4)
DMZ# ping 172.16.4.4 source 172.16.7.7 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 1/1/1 ms
DMZ# ICMP: echo reply rcvd, src 172.16.4.4, dst 172.16.7.7
!
! ASA now considers the echo request and reply parts of a single connection
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.4.4/0 gaddr
172.16.7.7/0 laddr 172.16.7.7/0
ICMP echo request from dmz:172.16.7.7 to out:172.16.4.4 ID=0 seq=0 len=72
ICMP echo reply from out:172.16.4.4 to dmz:172.16.7.7 ID=0 seq=0 len=72

```

Inbound Ping

Example 7-11 employs a sample Access Control Entry that enables echo requests from host 172.16.4.4 to host 172.16.7.7 (located on a higher `sec-lvl` interface). The hit count generated by this PING task is also registered with the `show access-list` command.

Example 7-11 *Enabling Pings from the Outside*

```

! Defining the ACL to allow echo requests from the OUT host to the DMZ host
access-list OUT extended permit icmp host 172.16.4.4 host 172.16.7.7 echo
!
OUT# ping 172.16.7.7 source 172.16.4.4 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
OUT# ICMP: echo reply rcvd, src 172.16.7.7, dst 172.16.4.4
!
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.4.4/1 gaddr
172.16.7.7/0 laddr 172.16.7.7/0
ICMP echo request from out:172.16.4.4 to dmz:172.16.7.7 ID=1 seq=0 len=72
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.4.4/1 gaddr
172.16.7.7/0 laddr 172.16.7.7/0
ICMP echo reply from dmz:172.16.7.7 to out:172.16.4.4 ID=1 seq=0 len=72
!
ASA2# show access-list OUT
access-list OUT; 1 elements; name hash: 0xcd7d0798
access-list OUT line 1 extended permit icmp host 172.16.4.4 host 172.16.7.7 echo
(hitcnt=1) 0xd9f0edd2

```

Windows Traceroute Through ASA

Echo request and echo reply messages are doubtlessly useful for connectivity tests. Nonetheless, some situations do require more information about the L3 hops traversed as packets travel from source to destination. This process is referred to as *tracing a route* and its main components follow:

- **A protocol used for the probe packets:** The protocols are typically ICMP (Windows) and UDP (UNIX and Cisco IOS, for instance).
- **TTL Field Manipulation:** Routers decrement the Time-To-Live field as they forward IP packets. A datagram that arrives with a value of one for the TTL parameter is discarded. (Because the TTL becomes zero after being decremented by this device.)
- **ICMP time-exceeded messages (Type 11, Code 0):** These are sent back to the originating host of the probe with the source address of the router that discarded the packet. This is the base mechanism for determining each hop in the path.
- **ICMP echo reply messages:** For each ICMP probe received, the final host sends an echo reply back to the source.
- **ICMP port unreachable messages:** For each UDP probe received, an ICMP port unreachable message (Type 3, Code 3) is sent back to the source by the final host.

In case a firewall exists between source and destination, the pertinent ICMP messages need to be permitted through it so that the trace can go on. Examples 7-12 to 7-16 deal with tracing activities that take place in an environment that includes an ASA appliance.

Example 7-12 registers a **tracert** started from a *command prompt* on the *Windows* host 172.16.7.100 and intended to determine the path to host 172.16.4.4. ASA, by default, does not count itself as a hop, and the resultant route shows only one hop.

The Windows **tracert** tool sends a set of 03 echo request probes for each hop it tries to identify. Example 7-12 focuses on the first of these packets. Similarly to what was done in Example 7-8, the request is enabled by the implicit permit for outbound and the reply by creating an explicit permission.

Example 7-13 illustrates Windows **tracert** in action for a path that includes two routers. The first router (172.16.4.4) sends a *time-exceeded* message, whereas the second (172.16.40.40) sends the echo reply (because it represents the final host). Figure 7-4 further details this by showing two probe packets captured by ASA:

- The first PING of the initial probe set is from 172.16.7.100 to 172.16.40.40 with TTL = 1. This is used to determine the first hop.
- The first PING of the second probe set is now TTL = 2, with the purpose of discovering the second hop.

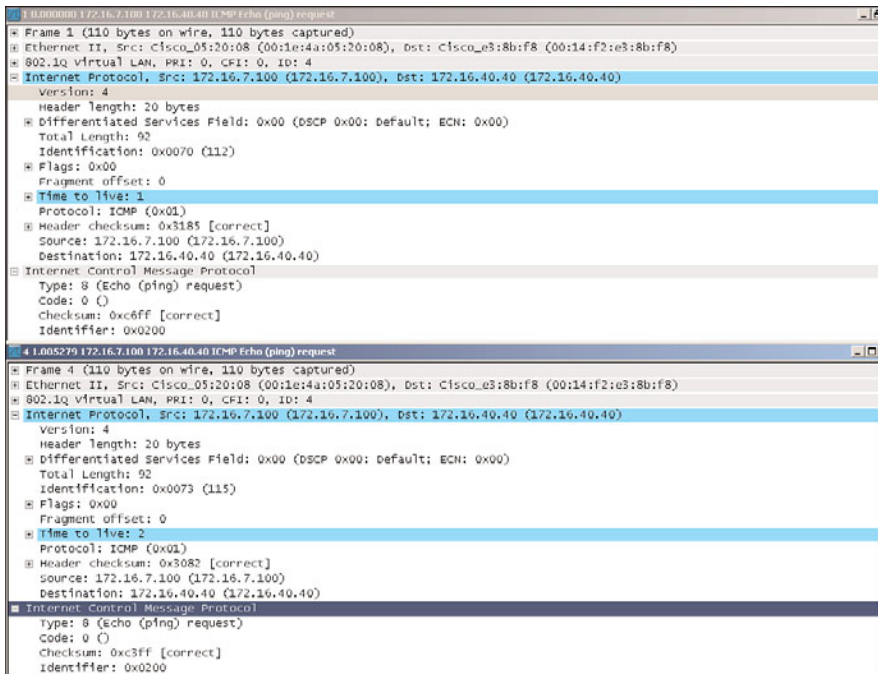


Figure 7-4 Packet Capture for the Windows Traceroute of Example 7-13

Example 7-12 Sample Windows Traceroute (1)

```
! Tracert to 172.16.4.4 from the host 172.16.7.100 (03 probes are sent)
C:\Documents and Settings\Administrator>tracert 172.16.4.4
Tracing route to 172.16.4.4 over a maximum of 30 hops
  1      1 ms      1 ms      1 ms    172.16.4.4
Trace complete.
C:\Documents and Settings\Administrator>
!
! ACL on ASA catching the inbound packets (interface 'out')
ASA# show access-list OUT | grep -v elements|=0
access-list OUT line 3 extended permit icmp host 172.16.4.4 host 172.16.7.100
echo-reply (hitcnt=1) 0x55e4f6e6
!
! Connection Creation and 'debug icmp trace'(for 01 probe) on ASA
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.4.4/0 gaddr
172.16.7.100/512 laddr 172.16.7.100/512
ICMP echo request from dmz:172.16.7.100 to out:172.16.4.4 ID=512 seq=11264 len=64
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.4.4/0 gaddr
172.16.7.100/512 laddr 172.16.7.100/512
ICMP echo reply from out:172.16.4.4 to dmz:172.16.7.100 ID=512 seq=11264 len=64
!
! Echo Reply from final host (172.16.4.4)
OUT#ICMP: echo reply sent, src 172.16.4.4, dst 172.16.7.100
```

Example 7-13 *Sample Windows Traceroute (2)*

```

! Tracert to 172.16.40.40 from the host 172.16.7.100 (03 probes are sent)
C:\Documents and Settings\Administrator>tracert 172.16.40.40
Tracing route to 172.16.40.40 over a maximum of 30 hops
    1     1 ms     1 ms     1 ms    172.16.4.4
    2     1 ms     1 ms     1 ms    172.16.40.40
Trace complete.
C:\Documents and Settings\Administrator>
!
ASA2# show access-list OUT | grep -v elements|=0
access-list OUT line 1 extended permit icmp host 172.16.4.4 host 172.16.7.100
time-exceeded (hitcnt=1) 0xef623c62
access-list OUT line 6 extended permit icmp host 172.16.40.40 host 172.16.7.100
echo-reply (hitcnt=1) 0x611f5a50
!
! Connection creation and 'debug icmp trace' on ASA (first hop in the path)
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.40.40/0 gaddr
172.16.7.100/512 laddr 172.16.7.100/512
ICMP echo request from dmz:172.16.7.100 to out:172.16.40.40 ID=512 seq=12032 len=64
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.4.4/0 gaddr
172.16.7.100/0 laddr 172.16.7.100/0
ICMP echo request from dmz:172.16.7.100 to out:172.16.40.40 ID=512 seq=12288 len=64
ICMP echo request from dmz:172.16.7.100 to out:172.16.40.40 ID=512 seq=12544 len=64
!
! Connection creation and 'debug icmp trace' on ASA (second hop in the path)
ICMP echo request from dmz:172.16.7.100 to out:172.16.40.40 ID=512 seq=12800 len=64
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.40.40/0 gaddr
172.16.7.100/512 laddr 172.16.7.100/512
ICMP echo reply from out:172.16.40.40 to dmz:172.16.7.100 ID=512 seq=12800 len=64
!
! ICMP Time-exceeded message from intermediate host (172.16.4.4)
OUT# ICMP: time exceeded (time to live) sent to 172.16.7.100 (dest was
172.16.40.40)
!
! Echo Reply from final host (172.16.40.40)
OUTER# ICMP: echo reply sent, src 172.16.40.40, dst 172.16.7.100

```

UDP Connection Examples

The setup of UDP connections through ASA takes into account six of the traditional Netflow *key-fields* presented in Chapter 4, “Lean the Tools. Know the Firewall.” Because UDP does not define a finite state machine and has no concept of connection flags, the

ASA algorithm relies on timing information (*inactivity timeouts*) to dynamically close UDP connections. The default ASA timeout for UDP is 02 minutes.

This section uses the reference topology of Figure 7-2 to study the creation of some UDP connections through ASA. The utility chosen for the tests is the **IOS Traceroute**.

Outbound IOS Traceroute Through ASA

Examples 7-12 and 7-13 documented the Windows **tracert** resource. The current section devotes attention to **IOS Traceroute**, which relies on UDP probes (instead of ICMP) and focuses on some ICMP error messages (received from hops throughout the path) to register the complete route.

Example 7-14 registers the behavior of outbound IOS Traceroute through ASA. An ACL called OUT was crafted to catch the possible ICMP messages in the context of Figure 7-2. The first probe uses, by default, a destination port numbered 33434 and a random source port (49210 in the example) and sets TTL = 1. The second probe uses UDP destination port 33435 and increases the TTL to 2. The procedure is repeated, as needed, up to 30 hops.

In Example 7-14, the first hop (172.16.4.4) sent back to the source (172.16.7.7) is a *time-exceeded* message, whereas the last hop (172.16.40.40) sent a *port unreachable* (confirming that it is the last hop). The UDP connections were displayed with the **show conn** command, and an output filter for the **show access-list** command was employed to select the ACEs with non-null hit counts. Figure 7-5 portrays the packet capture (performed by ASA) for the two probes of the current example.

Example 7-14 ASA Not Seen as a Hop of the Traceroute

```
! ACL to catch possible ICMP messages from hops in the path
access-list OUT extended permit icmp host 172.16.4.4 host 172.16.7.7 echo-reply
access-list OUT extended permit icmp host 172.16.4.4 host 172.16.7.7 unreachable
access-list OUT extended permit icmp host 172.16.4.4 host 172.16.7.7 time-exceeded
access-list OUT extended permit icmp host 172.16.40.40 host 172.16.7.7 unreachable
access-list OUT extended permit icmp host 172.16.40.40 host 172.16.7.7 time-
exceeded
access-list OUT extended permit icmp host 172.16.40.40 host 172.16.7.7 echo-reply
!
! Traceroute started from IOS Device (uses UDP probes and initial source port
33434)
DMZ# traceroute 172.16.40.40 source 172.16.7.7 probe 1
Tracing the route to 172.16.40.40
 1 172.16.4.4 4 msec
 2 172.16.40.40 4 msec
DMZ#
ICMP: time exceeded rcvd from 172.16.4.4
ICMP: dst (172.16.7.7) port unreachable rcv from 172.16.40.40
!
```



```

! Outbound UDP and Inbound ICMP connections associated with the Traceroute
%ASA-6-302015: Built outbound UDP connection 336 for out:172.16.40.40/33434
(172.16.40.40/33434) to dmz:172.16.7.7/49210 (172.16.7.7/49210)
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.4.4/0 gaddr
172.16.7.7/0 laddr 172.16.7.7/0
%ASA-6-302015: Built outbound UDP connection 338 for out:172.16.40.40/33435
(172.16.40.40/33435) to dmz:172.16.7.7/49211 (172.16.7.7/49211)
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.40.40/0 gaddr
172.16.7.7/0 laddr 172.16.7.7/0
!
! UDP connections used for Traceroute (default timeout is 02 seconds)
ASA2# show conn
2 in use, 30 most used
UDP out 172.16.40.40:33435 dmz 172.16.7.7:49211, idle 0:00:07, bytes 0, flags -
UDP out 172.16.40.40:33434 dmz 172.16.7.7:49210, idle 0:00:07, bytes 0, flags -
!
! Displaying ACEs with non-null hit counts
ASA2# show access-list OUT ! exclude =0
access-list OUT; 5 elements; name hash: 0xcd7d0798
access-list OUT line 3 extended permit icmp host 172.16.4.4 host 172.16.7.7
time-exceeded (hitcnt=1) 0x6f793dcd
access-list OUT line 4 extended permit icmp host 172.16.40.40 host 172.16.7.7
unreachable (hitcnt=1) 0x0033cdbc

```

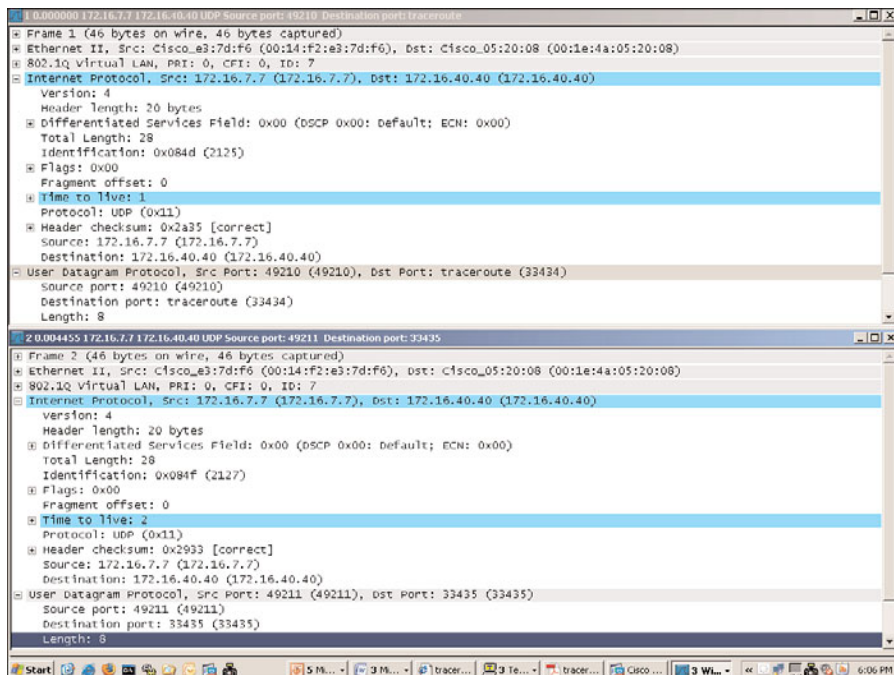


Figure 7-5 Packet Capture for Traceroute of Example 7-14

Example 7-15 illustrates a way to instruct ASA to insert itself as a hop in the path. ASA sends a *time-exceeded* message with source address 172.16.4.2, which now appears as the first hop in the **traceroute** result (refer to Figure 7-2). Additionally, three UDP connections related to the traceroute task are visible with the **show conn** command.

Example 7-15 *Instructing ASA to Display Itself as a Hop in the Traceroute*

```

policy-map global_policy
class class-default
  set connection decrement-ttl
!
service-policy global_policy global
!
! The traceroute initiator now sees ASA as a hop
DMZ# traceroute 172.16.40.40 source 172.16.7.7 probe 1
Tracing the route to 172.16.40.40
  1 172.16.4.2 4 msec
  2 172.16.4.4 4 msec
  3 172.16.40.40 4 msec
ICMP: time exceeded rcvd from 172.16.4.2
ICMP: time exceeded rcvd from 172.16.4.4
ICMP: dst (172.16.7.7) port unreachable rcv from 172.16.40.40
!
! UDP connections derived from the traceroute task
ASA2# show conn
3 in use, 30 most used
UDP out 172.16.40.40:33436 dmz 172.16.7.7:49215, idle 0:00:44, bytes 0, flags -
UDP out 172.16.40.40:33435 dmz 172.16.7.7:49214, idle 0:00:44, bytes 0, flags -
UDP out 172.16.40.40:33434 dmz 172.16.7.7:49213, idle 0:00:44, bytes 0, flags -

```

Example 7-15 intended to change the behavior of the ASA algorithm for **traceroute** purposes, but the configuration used affected every packet crossing the firewall. A more controlled intervention is frequently desirable and, as such, motivates the inclusion of Example 7-16. In this new scenario (still based upon the topology of Figure 7-2), ASA acts on only the TTL value for packets that have source and destination in the 172.16.0.0/16 major network and whose connection setup attempts arrive at the 'dmz' interface. This example also points out how to locate flows that match solely the **global policy** and those that are also in accordance with the interface-specific **service-policy**.

Example 7-16 *Changing the ASA Traceroute Behavior Only for Some Flows*

```

! Defining a service-policy that applies only to some specific flows
access-list TRACE extended permit udp 172.16.0.0 255.255.0.0 172.16.0.0
255.255.0.0 range 33434 33534
!
class-map CLASS2
  match access-list TRACE
!
policy-map EXCEPTIONS1
class CLASS2
  set connection decrement-ttl
!
service-policy EXCEPTIONS1 interface dmz
!
! Sample flow that matches the class-map CLASS2 in the interface policy EXCEPTIONS1
ASA2# show service-policy flow udp host 172.16.7.7 host 172.16.40.40 eq 33434
Global policy:
  Service-policy: global_policy
  Class-map: class-default
  Match: any
  Action:
    Output flow:          Input flow:
Interface dmz:
  Service-policy: EXCEPTIONS1
  Class-map: CLASS2
  Match: access-list TRACE
  Access rule: permit udp 172.16.0.0 255.255.0.0 172.16.0.0 255.255.0.0
range 33434 33534
  Action:
    Input flow: set connection decrement-ttl
  Class-map: class-default
  Match: any
  Action:
    Output flow:
!
! Sample flow that does not match CLASS2 in the interface-specific service-policy
ASA2# show service-policy flow udp host 172.20.20.20 host 172.30.30.30 eq 33434
Global policy:
  Service-policy: global_policy
  Class-map: class-default
  Match: any
  Action:
    Output flow:          Input flow:
Interface dmz:
  Service-policy: EXCEPTIONS1
  Class-map: class-default

```

```
Match: any
Action:
Output flow:
```

Note The method documented in Example 7-16 can be used to instruct ASA to display itself as a hop for INTRANET destinations and to remain hidden, for instance, for probes coming from the INTERNET.

TCP Connection Examples

TCP employs a finite state machine that specifies, for example, the expected state transitions for connection setup and termination—this makes TCP a truly stateful protocol. When dealing with TCP connections, stateful firewalls have visibility not only of the key-fields that are used for flow definition but also of the TCP state *flags*.

ASA Flags Associated with TCP Connections

A set of flags, as depicted in Figure 7-6, was added to the ASA algorithm to reflect the state of TCP connections crossing the firewall. The **show conn detail** command reveals the available connectionflags, as shown in Example 7-17.

The following step list refers to the *Outbound* flow represented in Figure 7-6. The interpretation of the *Inbound* ASA flags is completely analogous and left as an exercise for you.

- Step 1.** After the SYN sent by the inside client has crossed the firewall (first step of the three-way handshake), ASA waits for the SYN/ACK from the outside server (second step) and, in the sequence, from the client ACK (third step). ASA represents this first phase with the combination of flags **saA**.
- Step 2.** Having received the SYN/ACK back from the outside server, ASA clears the **sa** indication and keeps only the **A** flag. (Because the inside ACK is still missing.)
- Step 3.** ASA registers the receipt of the inside ACK with the **U** flag, a way to state that the setup phase is over and that the connection is considered **up**.
- Step 4.** The addition of the **I** flag in the fourth step means that the data was received from the outside (*inbound data*).
- Step 5.** The insertion of the **O** flag means that outbound data (data from the inside) has been seen by ASA. The combination **UIO** of the last step tells that the connection is up and that data was already received from both the outside server and the inside client.

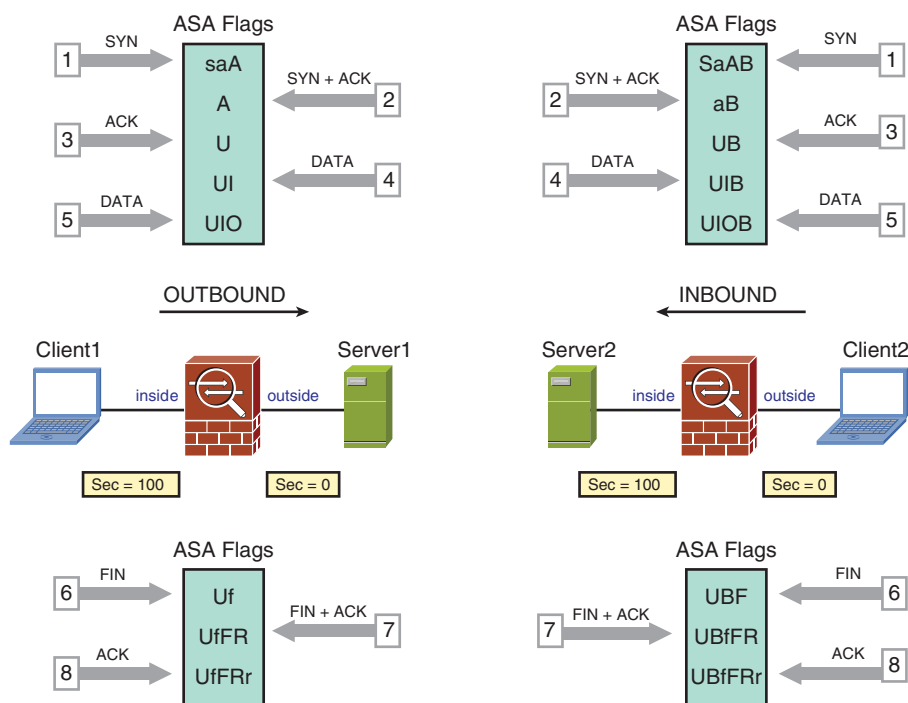


Figure 7-6 ASA Flags for TCP Connections

Example 7-17 TCP Connection Flags

```
%ASA-6-302013: Built outbound TCP connection 478 for out:172.16.4.4/23
(172.16.4.4/23) to dmz:172.16.7.7/17410 (172.16.7.7/17410)
%ASA-6-302013: Built outbound TCP connection 453 for out:172.16.4.4/80
(172.16.4.4/80) to dmz:172.16.7.100/1038 (172.16.7.100/1038)
!
```

ASA2# show conn detail

2 in use, 30 most used

Flags: A - awaiting inside ACK to SYN, a - awaiting outside ACK to SYN,
 B - initial SYN from outside, b - TCP state-bypass or nailed, C - CTIQBE
 media,
 D - DNS, d - dump, E - outside back connection, **F - outside FIN**, f -
 inside FIN,
 G - group, g - MGCP, H - H.323, h - H.225.0, **I - inbound data**,
 i - incomplete, J - GTP, j - GTP data, K - GTP t3-response
 k - Skinny media, M - SMTP data, m - SIP media, n - GUP
O - outbound data, P - inside back connection, p - Phone-proxy TFTP con-
 nection,
 q - SQL*Net data, **R - outside acknowledged FIN**,
 R - UDP SUNRPC, r - inside acknowledged FIN, S - awaiting inside SYN,
 s - awaiting outside SYN, T - SIP, t - SIP transient, **U - up**,

```

V - VPN orphan, W - WAAS,
X - inspected by service module
TCP out:172.16.4.4/23 dmz:172.16.7.7/17410,
    flags UIO, idle 14s, uptime 14s, timeout 1h0m, bytes 119
TCP out:172.16.4.4/80 dmz:172.16.7.100/1038,
    flags UFRIO, idle 18s, uptime 18s, timeout 10m0s, bytes 557
!
! Sample TCP connection attempts denied by ASA
%ASA-6-106015: Deny TCP (no connection) from 172.16.220.110/6006 to
172.16.222.10/80 flags SYN RST URG on interface out1
%ASA-6-106015: Deny TCP (no connection) from 172.16.220.111/6007 to
172.16.222.10/80 flags FIN SYN RST ACK on interface out1
%ASA-6-106015: Deny TCP (no connection) from 172.16.220.112/6011 to
172.16.222.10/80 flags PSH ACK on interface out1

```

Note Only connections that have the SYN flag set in the first packet (value 0x02 in the Flags field) are allowed to start the TCP three-way handshake through ASA. If any other flag combination is seen in the first packet, the connection is denied (as shown in the last part of Example 7-17).

TCP Sequence Number Randomization

The previous discussions in this section pointed out some basic controls that the ASA Stateful Inspection algorithm implements when dealing with TCP:

- The ASA algorithm dynamically opens and closes client and server ports for the pertinent source and destination IP addresses.
- The flags that represent the TCP finite state machine are always considered by ASA, during connection setup, maintenance, or termination.
- The state table keeps timeout information for every existent connection. If the inactivity (or idle) timeout is reached, the connection is automatically closed.

Besides what was just summarized, ASA supports *TCP Sequence Number Randomization*, a feature that certainly adds value to security. With this resource in place, ASA acts as sort of a *proxy* for sequence numbers and does not enable the client and server to know the actual value selected by the other. Two classic attacks that this mechanism helps prevent follow:

- **Initial Sequence Number (ISN) guessing:** A sort of man-in-the-middle attack that tries to discover the sequence numbers involved (supposed to be randomly generated but not that random in practice) and hijack a session.
- **OS Fingerprinting:** Typically used as one the first phases on network reconnaissance procedures, it aims to discover the operating system running on the target machine.

With the OS determined, it is easier to search vulnerability databases and find an available exploit for that specific OS.

Example 7-18 depicts the behavior of ASA when TCP sequence number randomization has been disabled, whereas Example 7-19 illustrates it in action (which is the default setting). Figure 7-7 builds upon Example 7-19 by detailing the sequence numbers seen on each side of ASA (inside client and outside server).

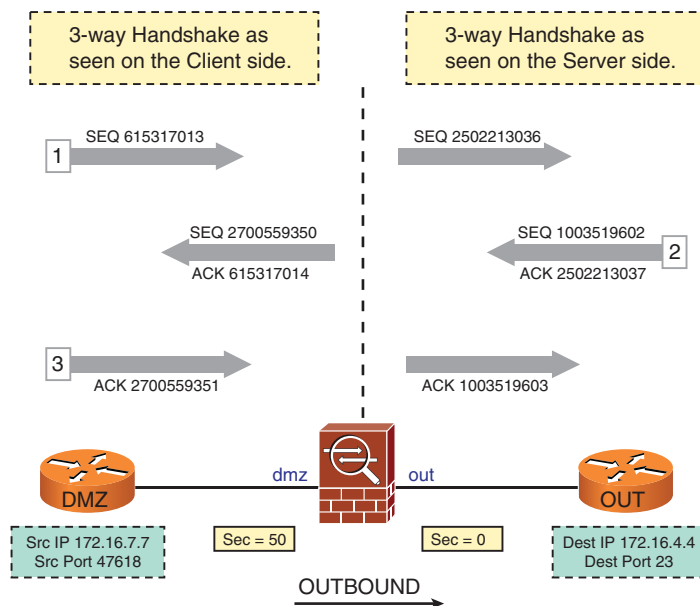


Figure 7-7 Illustrating TCP Sequence Number Randomization

Example 7-18 TCP Connection with No Sequence Number Randomization

```
! Disabling the TCP Sequence Number Randomization (enabled by default)
policy-map global_policy
class class-default
  set connection random-sequence-number disable
!
! Verifying the configuration change
ASA2# show service-policy global set connection detail
Global policy:
  Service-policy: global_policy
  Class-map: class-default
    Set connection policy: random-sequence-number disable
    drop 0
!
```

! Telnet to OUT host (172.16.4.4) from DMZ host (172.16.7.7)

DMZ# telnet 172.16.4.4

Trying 172.16.4.4 ... Open

User Access Verification

Password: *****

[output suppressed]

Reserved port 55810 in Transport Port Agent for TCP IP type 1

TCP: sending SYN, seq **1302332064**, ack 0

TCP2: Connection to 172.16.4.4:23, advertising MSS 536

TCP2: state was CLOSED -> **SYNSENT** [55810 -> 172.16.4.4(23)]

TCP2: state was SYNSENT -> **ESTAB** [55810 -> 172.16.4.4(23)]

TCP: tcb 83E09B2C connection to 172.16.4.4:23, peer MSS 536, MSS is 536

TCB83E09B2C connected to 172.16.4.4.23

OUT>

OUT>show tcp brief

TCB	Local Address	Foreign Address	(state)
84D92CD8	172.16.4.4.23	172.16.7.7.55810	ESTAB

!

! Outbound TCP connection as seen on ASA

%ASA-6-302013: Built outbound TCP connection 660 for out:172.16.4.4/23 (172.16.4.4/23) to dmz:172.16.7.7/55810 (172.16.7.7/55810)

!

! Ingress Capture on ASA ('dmz' interface)

1: 23:20:09.231891 802.1Q vlan#7 P0 172.16.7.7.55810 > 172.16.4.4.23: **S 1302332064**:1302332064(0) win 4128 <mss 536>

2: 23:20:09.234393 802.1Q vlan#7 P0 172.16.4.4.23 > 172.16.7.7.55810: S 3381677038:3381677038(0) **ack 1302332065** win 4128 <mss 536>

3: 23:20:09.235491 802.1Q vlan#7 P0 172.16.7.7.55810 > 172.16.4.4.23: . ack 3381677039 win 4128

!

! Egress Capture on ASA ('out' interface)

1: 23:20:09.232257 802.1Q vlan#4 P0 172.16.7.7.55810 > 172.16.4.4.23: **S 1302332064**:1302332064(0) win 4128 <mss 536>

2: 23:20:09.234362 802.1Q vlan#4 P0 172.16.4.4.23 > 172.16.7.7.55810: S 3381677038:3381677038(0) **ack 1302332065** win 4128 <mss 536>

3: 23:20:09.235522 802.1Q vlan#4 P0 172.16.7.7.55810 > 172.16.4.4.23: . ack 3381677039 win 4128

!

! Connection Setup on OUT host

TCP0: state was LISTEN -> **SYNRCVD** [23 -> 172.16.7.7(55810)]

TCP: tcb 84D92CD8 connection to 172.16.7.7:55810, peer MSS 536, MSS is 516

TCP: sending SYN, seq 3381677038, **ack 1302332065**

TCP0: Connection to 172.16.7.7:55810, advertising MSS 536

TCP0: state was SYNRCVD -> **ESTAB** [23 -> 172.16.7.7(55810)]

Example 7-19 *TCP Connection Involving Sequence Number Randomization*

```

! Telnet to OUT host (172.16.4.4) from DMZ host (172.16.7.7)
DMZ# telnet 172.16.4.4
Trying 172.16.4.4 ... Open
User Access Verification
Password: *****
[ output suppressed ]
Reserved port 47618 in Transport Port Agent for TCP IP type 1
TCP: sending SYN, seq 615317013, ack 0
TCP2: Connection to 172.16.4.4:23, advertising MSS 536
TCP2: state was CLOSED -> SYNSENT [47618 -> 172.16.4.4(23)]
TCP2: state was SYNSENT -> ESTAB [47618 -> 172.16.4.4(23)]
TCP: tcb 84DF58F4 connection to 172.16.4.4:23, peer MSS 536, MSS is 536
TCB84DF58F4 connected to 172.16.4.4.23
OUT>
!
! Outbound TCP connection as seen on ASA
%ASA-6-302013: Built outbound TCP connection 629 for out:172.16.4.4/23
(172.16.4.4/23) to dmz:172.16.7.7/47618 (172.16.7.7/47618)
!
! Ingress Capture on ASA ('dmz' interface)
1: 22:39:42.725258 802.1Q vlan#7 P0 172.16.7.7.47618 > 172.16.4.4.23: S
615317013:615317013(0) win 4128 <mss 536>
2: 22:39:42.727684 802.1Q vlan#7 P0 172.16.4.4.23 > 172.16.7.7.47618: S
2700559350:2700559350(0) ack 615317014 win 4128 <mss 536>
3: 22:39:42.728783 802.1Q vlan#7 P0 172.16.7.7.47618 > 172.16.4.4.23: . ack
2700559351 win 4128
!
! Egress Capture on ASA ('out' interface)
1: 22:39:42.725563 802.1Q vlan#4 P0 172.16.7.7.47618 > 172.16.4.4.23: S
2502213036:2502213036(0) win 4128 <mss 536>
2: 22:39:42.727654 802.1Q vlan#4 P0 172.16.4.4.23 > 172.16.7.7.47618: S
1003519602:1003519602(0) ack 2502213037 win 4128 <mss 536>
3: 22:39:42.728813 802.1Q vlan#4 P0 172.16.7.7.47618 > 172.16.4.4.23: . ack
1003519603 win 4128
!
! Connection Setup on OUT host
TCP0: state was LISTEN -> SYNRCVD [23 -> 172.16.7.7(47618)]
TCP: tcb 83CFFCEC connection to 172.16.7.7:47618, peer MSS 536, MSS is 516
TCP: sending SYN, seq 1003519602, ack 2502213037
TCP0: Connection to 172.16.7.7:47618, advertising MSS 536
TCP0: state was SYNRCVD -> ESTAB [23 -> 172.16.7.7(47618)]

```

Example 7-20 proposes a way to selectively disable the randomization feature by means of the **Modular Policy** structure available in ASA. In this example, the BGP session falls under the interface **service-policy**; therefore, it has the randomization process disabled.

The sample telnet session, on the other hand, matches only the global policy and presents the default behavior.

Example 7-20 *Randomizing the TCP Sequence Number Only for Some Specific Flows*

```

access-list BGP extended permit tcp 172.16.0.0 255.255.0.0 172.16.0.0
255.255.0.0 eq bgp
!
class-map CLASS1
  match access-list BGP
!
policy-map EXCEPTIONS1
  class CLASS1
    set connection random-sequence-number disable
!
service-policy EXCEPTIONS1 interface dmz
!
! This BGP connection matches the class-map CLASS1 in the interface service-policy
ASA2# show service-policy flow tcp host 172.16.7.7 host 172.16.4.4 eq bgp
Global policy:
  Service-policy: global_policy
  Class-map: class-default
  Match: any
  Action:
    Output flow:      Input flow:
Interface dmz:
  Service-policy: EXCEPTIONS1
  Class-map: CLASS1
  Match: access-list BGP
  Access rule: permit tcp 172.16.0.0 255.255.0.0 172.16.0.0 255.255.0.0 eq
bgp
  Action:
    Input flow: set connection random-sequence-number disable
  Class-map: class-default
  Match: any
  Action:
    Output flow:
!
! This telnet flow matches only the global policy
ASA2# show service-policy flow tcp host 172.16.7.7 host 172.16.4.4 eq telnet
Global policy:
  Service-policy: global_policy
  Class-map: class-default
  Match: any

```

```

    Action:
      Output flow:      Input flow:
Interface dmz:
  Service-policy: EXCEPTIONS1
  Class-map: class-default
    Match: any
    Action:
    Output flow:

```

Note At this point, you might be asking, “If the Randomization mechanism adds value to Security, what is the point of disabling it?” In some specific situations, the TCP-based protocol crossing the firewall might use a kind of hash to verify a combination of fields in the TCP header. If the Sequence Number (SN) has been modified in transit, the hash calculated at the receipt of the packet will not match that sent by the peer. This results in a packet drop, which may be undesirable. One classic example that alters the SN is the pseudo header created by BGP when configured for MD5 authentication. Example 7-20 was conceived to reinforce the idea that it is quite convenient to have the possibility to change the default service-policy only when it truly matters.

Same Security Access

Having analyzed the operation of ASA for Inbound and Outbound, it is time to look at the particularities of the Same Security Access model.

By default, the ASA algorithm does not enable this category of access. If two interfaces are configured with the same security level, traffic cannot flow between them (refer to Example 7-6).

In Example 7-21 the interfaces **out** and **dmz** have equal security levels (as previously configured in Example 7-6). The same security **implicit deny** rule holds even if a wide open ACL (**permit ip any any**) has been applied to the participant interfaces.

Example 7-21 Verifying Same Security Access Operation (1)

```

! Creating and applying two wide open ACLs to interfaces 'out' and 'dmz'
access-list EVERYTHING1 extended permit ip any any
access-list EVERYTHING2 extended permit ip any any
!
ASA2# show run access-group
access-group EVERYTHING1 in interface out
access-group EVERYTHING2 in interface dmz
!
! Trying to send a PING to 172.16.4.4 through ASA
DMZ#ping 172.16.4.4 source 172.16.7.7 repeat 1

```

```

Success rate is 0 percent (0/1)
!
! The implicit deny rule takes precedence over the configured ACL
%ASA-3-106014: Deny inbound icmp src dmz:172.16.7.7 dst out:172.16.4.4 (type 8,
code 0)
!
! Packet Tracer confirms the implicit deny behavior for a TCP connection attempt
ASA2# packet-tracer input dmz tcp 172.16.7.7 2000 172.16.4.4 80
[ output suppressed ]
Phase: 3
Type: ACCESS-LIST
Subtype:
Result: DROP
Config:
Implicit Rule
[ output suppressed ]
Action: drop
Drop-reason: (acl-drop) Flow is denied by configured rule

%ASA-2-106001: Inbound TCP connection denied from 172.16.7.7/2000 to 172.16.4.4/80
flags SYN on interface dmz

```

Example 7-22 employs the `same-security-traffic permit` command to enable the flow of traffic between two interfaces configured with equal security levels. The usage of this command inserts an *implicit permit* rule. To restrict the types of connections that might be established, the configuration of an ACL becomes necessary.

Figure 7-8 complements what has been discussed so far by showing the decision criteria for two interfaces configured with the values X and Y for their security levels.

Example 7-22 Verifying Same Security Access Operation (2)

```

! Verifying that Same Security Access is enabled
ASA2# show running-config | include same
same-security-traffic permit inter-interface
!
! Repeating the Packet Tracer simulation of Example 7-19
ASA2# packet-tracer input dmz tcp 172.16.7.7 2000 172.16.4.4 80
[ output suppressed ]
Phase: 4
Type: ACCESS-LIST
Subtype:
Result: ALLOW

```

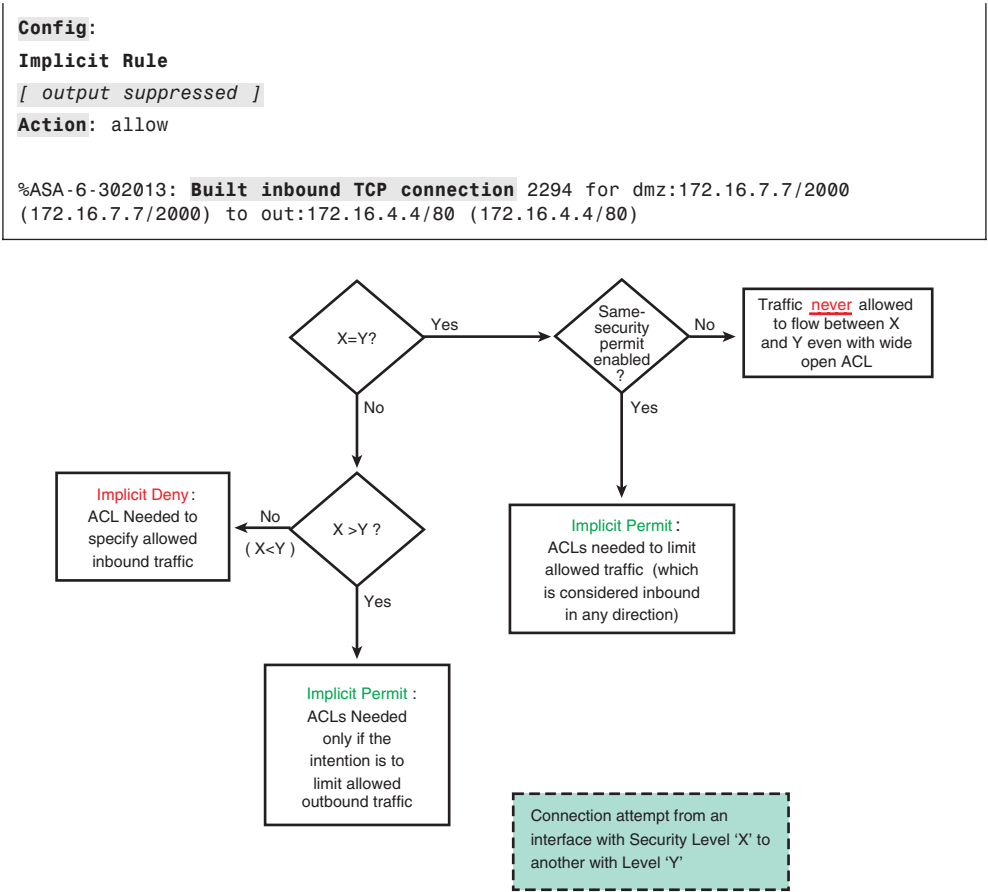


Figure 7-8 Summary of Flow Creation Rules for the no nat-control Model

Handling ACLs and Object-Groups

Object-groups were conceived to make ACL creation and maintenance easier. They become particularly relevant for those security administrators that need to deal with access control lists that are large or subject to frequent changes in their ACE components.

Example 7-23 shows some sample network **object-groups** that define collections of hosts or networks in any combination. Because ACLs have the basic construction model to define permissions *from source to destination*, you need to have this notion of direction in mind when creating the groupings.

Example 7-24 shows some **object-groups** that contain services related to only one of the main IP protocols (TCP, UDP or ICMP) in their definition.

Example 7-23 *Sample Network Object-Groups (Hosts and Networks)*

```

! Sample Network Object-Groups
object-group network OUT1
  network-object host 172.16.4.4
  network-object host 172.16.40.40
!
object-group network DMZ1
  network-object host 172.16.7.7
  network-object 172.20.20.0 255.255.255.0
!
object-group network LOG-HOSTS
  network-object host 172.16.7.200
!
object-group network INSIDE1
  network-object host 192.168.1.100

```

Example 7-24 *Basic Service Object-Groups*

```

! Sample ICMP service group
object-group icmp-type PING
  icmp-object echo
  icmp-object echo-reply
!
! Sample UDP service group
object-group service LOGGING udp
  port-object eq syslog
  port-object eq 2055
!
! Sample TCP service groups
object-group service TERMINAL tcp
  description * Remote Access to CLI (Telnet and SSH)
  port-object eq ssh
  port-object eq telnet
!
object-group service DESKTOP tcp
  description * Remote Desktop Access (VNC and RDP)
  port-object eq 5900
  port-object eq 3389

```

Example 7-25 illustrates the usage of some special types of service **object-groups**:

- **Nested Group:** The group TCP-MGMT refers to the groups named TERMINAL and DESKTOP, previously defined in Example 7-24.
- **Enhanced Group:** In the opposite range of the spectrum from those groups of Example 7-24, this category is not limited to the selection of a single protocol. The

group called AUTH, for instance, contains both TCP and UDP applications and enables the inclusion of the source port in its definition. If a source port is specified, it binds to the source host in the ACL configuration. The way the elements of the group are expanded when applied to an ACL, may be viewed in Example 7-29 (line 2 of the ACL named DMZ).

- **Equal ports for TCP and UDP:** Some application protocols have the same service port number reserved for TCP and UDP. The group SMB uses the service type **tcp-udp** to reflect that the chosen ports apply simultaneously to both transport protocols.
- **Protocol Group:** The protocol group called **tcp-udp** is intended for use with the service groups of the **tcp-udp** type. An illustration of such a combination is found in Example 7-29 (line 1 of the ACL named DMZ).

Example 7-26 just teaches how to locate **object-groups** within the *running-config*. Figure 7-9 shows the ASDM counterpart of the CLI configuration for the service **object-groups**.

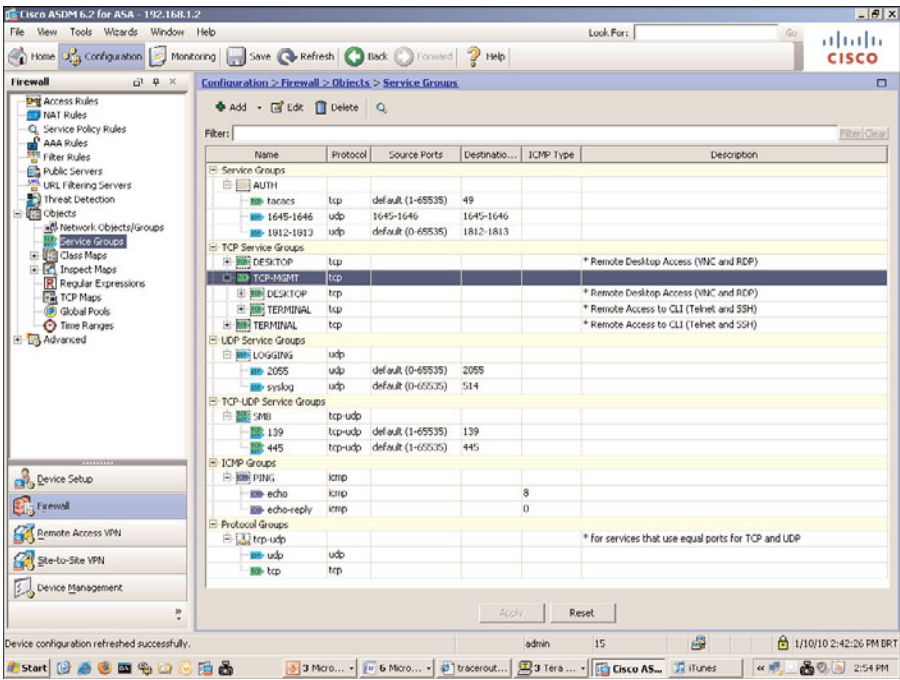


Figure 7-9 Service object-groups in ASDM

Example 7-25 Special Service Object-Groups

! Nesting Service Groups (using previously defined object-groups inside a new group)

```
object-group service TCP-MGMT tcp
group-object TERMINAL
```

```

group-object DESKTOP
!
! Sample Enhanced Service Group
object-group service AUTH
  service-object udp range 1812 1813
  service-object tcp eq tacacs
  service-object udpsource range radius radius-acct range radius radius-acct
!
! Sample Services Group that use equal ports for TCP and UDP
object-group service SMB tcp-udp
  port-object eq 139
  port-object eq 445
!
! Creating a protocol group to specify equal TCP and UDP ports simultaneously
object-group protocol tcp-udp
  description * for services that use equal ports for TCP and UDP
  protocol-object udp
  protocol-object tcp

```

Example 7-26 Finding object-groups in the running-config

```

show running-config object-group protocol
show running-config object-group icmp-type
show running-config object-group service
show running-config object-group network
show running-config object-group id LOGGING

```

Examples 7-27 through 7-29 combine the network, protocol, and service **object-groups** defined in Examples 7-23 to 7-25 to create structured and compact ACLs. Some noteworthy characteristics of these combinations follow:

- In the **running-config**, the ACLs display solely in the summarized manner and do not exhibit line numbers.
- When displayed with the **show access-list** command, an ACL expands all the possible combinations of protocol, source hosts, destination hosts, and services specified by each category of **object-group** employed in the **access-list** creation. The line number for each expanded line bears the same number of the condensed entry.
- Each expanded ACE displays its individual hit counts and is assigned a unique hash value used for later correlation with the logging messages it generates (in cases where logging for the individual permit ACE is explicitly enabled, as illustrated in Example 7-30).
- The first output line of the **show access-list** command shows the number of elements derived from all the possible combinations of **object-groups**.

Example 7-27 *Sample ACL That Uses Object-Groups (1)*

```

! Viewing the configured access-list
ASA2# show running-config access-list OUT
access-list OUT extended permit icmp 172.16.0.0 255.255.0.0 172.16.0.0 255.255.0.0
object-group PING
access-list OUT extended permit udp object-group OUT1 object-group LOG-HOSTS
object-group LOGGING
!

! Viewing the applied access-list and the individual hit counts
ASA2# show access-list OUT
access-list OUT; 6 elements; name hash: 0xcd7d0798
access-list OUT line 1 extended permit icmp 172.16.0.0 255.255.0.0 172.16.0.0
255.255.0.0 object-group PING 0x61f4cfd9
    access-list OUT line 1 extended permit icmp 172.16.0.0 255.255.0.0 172.16.0.0
255.255.0.0 echo (hitcnt=0) 0x7463e142
    access-list OUT line 1 extended permit icmp 172.16.0.0 255.255.0.0 172.16.0.0
255.255.0.0 echo-reply (hitcnt=0) 0x5d620d78
access-list OUT line 2 extended permit udp object-group OUT1 object-group LOG-
HOSTS object-group LOGGING 0x978330ad
    access-list OUT line 2 extended permit udp host 172.16.4.4 host 172.16.7.200 eq
syslog (hitcnt=0) 0xe0c6a264
    access-list OUT line 2 extended permit udp host 172.16.4.4 host 172.16.7.200 eq
2055 (hitcnt=0) 0x485f0680
    access-list OUT line 2 extended permit udp host 172.16.40.40 host 172.16.7.200
eq syslog (hitcnt=0) 0xdc783115
    access-list OUT line 2 extended permit udp host 172.16.40.40 host 172.16.7.200
eq 2055 (hitcnt=0) 0x6880d9ac
!

! Viewing the configured access-groups
ASA2# show running-config access-group
access-group INSIDE in interface inside
access-group OUT in interface out
access-group DMZ in interface dmz

```

Example 7-28 *Sample ACL That Uses Object-Groups (2)*

```

ASA2# show access-list INSIDE
access-list INSIDE; 4 elements; name hash: 0xdedb237a
access-list INSIDE line 1 extended permit tcp object-group INSIDE1 host
192.168.7.7 object-group TCP-MGMT 0x19c4987e
    access-list INSIDE line 1 extended permit tcp host 192.168.1.100 host
192.168.7.7 eq ssh (hitcnt=0) 0x067bbe33
    access-list INSIDE line 1 extended permit tcp host 192.168.1.100 host
192.168.7.7 eq telnet (hitcnt=0) 0xdb5bcd9c
    access-list INSIDE line 1 extended permit tcp host 192.168.1.100 host
192.168.7.7 eq 5900 (hitcnt=0) 0x1ea6e3be
    access-list INSIDE line 1 extended permit tcp host 192.168.1.100 host
192.168.7.7 eq 3389 (hitcnt=0) 0x48539370

```

Example 7-29 *Sample ACL That Uses Object-Groups (3)*

```

ASA2# show access-list DMZ
access-list DMZ; 19 elements; name hash: 0x55d29ba9
access-list DMZ line 1 extended permit object-group tcp-udpobject-group
DMZ1object-group OUT1object-group SMB 0x3529d11d
    access-list DMZ line 1 extended permit udp host 172.16.7.7 host 172.16.4.4 eq
139 (hitcnt=0) 0x03dd090c
    access-list DMZ line 1 extended permit udp host 172.16.7.7 host 172.16.4.4 eq
445 (hitcnt=0) 0xe80bd445
    access-list DMZ line 1 extended permit udp host 172.16.7.7 host 172.16.40.40 eq
139 (hitcnt=0) 0xf5df7e4f
    access-list DMZ line 1 extended permit udp host 172.16.7.7 host 172.16.40.40 eq
445 (hitcnt=0) 0x9477bbf7
    access-list DMZ line 1 extended permit udp172.20.20.0 255.255.255.0 host
172.16.4.4 eq 139 (hitcnt=0) 0xf116c645
    access-list DMZ line 1 extended permit udp172.20.20.0 255.255.255.0 host
172.16.4.4 eq 445 (hitcnt=0) 0x1da5d6e4
    access-list DMZ line 1 extended permit udp172.20.20.0 255.255.255.0 host
172.16.40.40 eq 139 (hitcnt=0) 0xf8187fd8
    access-list DMZ line 1 extended permit udp172.20.20.0 255.255.255.0 host
172.16.40.40 eq 445 (hitcnt=0) 0x1385b14b
    access-list DMZ line 1 extended permit tcp host 172.16.7.7 host 172.16.4.4 eq
netbios-ssn (hitcnt=0) 0x1ad61ef5
    access-list DMZ line 1 extended permit tcp host 172.16.7.7 host 172.16.4.4 eq
445 (hitcnt=0) 0xc77faa9e
    access-list DMZ line 1 extended permit tcp host 172.16.7.7 host 172.16.40.40 eq
netbios-ssn (hitcnt=0) 0x0a4f154b
    access-list DMZ line 1 extended permit tcp host 172.16.7.7 host 172.16.40.40 eq
445 (hitcnt=0) 0xd0426268
    access-list DMZ line 1 extended permit tcp172.20.20.0 255.255.255.0 host
172.16.4.4 eq netbios-ssn (hitcnt=0) 0xb77b1a34
    access-list DMZ line 1 extended permit tcp172.20.20.0 255.255.255.0 host
172.16.4.4 eq 445 (hitcnt=0) 0x721ecb12
    access-list DMZ line 1 extended permit tcp172.20.20.0 255.255.255.0 host
172.16.40.40 eq netbios-ssn (hitcnt=0) 0x7f0b3f36
    access-list DMZ line 1 extended permit tcp172.20.20.0 255.255.255.0 host
172.16.40.40 eq 445 (hitcnt=0) 0xeeb31055
access-list DMZ line 2 extended permit object-group AUTH host 192.168.7.7 host
192.168.1.254 0x740d4d8a
    access-list DMZ line 2 extended permit udp host 192.168.7.7 host 192.168.1.254
range 1812 1813 (hitcnt=0) 0x09f557e8
    access-list DMZ line 2 extended permit tcp host 192.168.7.7 host 192.168.1.254
eq tacacs (hitcnt=0) 0x90180c1b
    access-list DMZ line 2 extended permit udp host 192.168.7.7 range radius
radius-acct host 192.168.1.254 range radius radius-acct (hitcnt=0) 0xb175a591

```

Figure 7-10 shows the Access Rules table in ASDM corresponding to the ACLs created using the **object-groups** of previous examples. The figure shows, on the right pane, that

you can exhibit existent **network object-groups** used for rule definition. It also underlines the possibility of searching for usage of a given **object-group**.

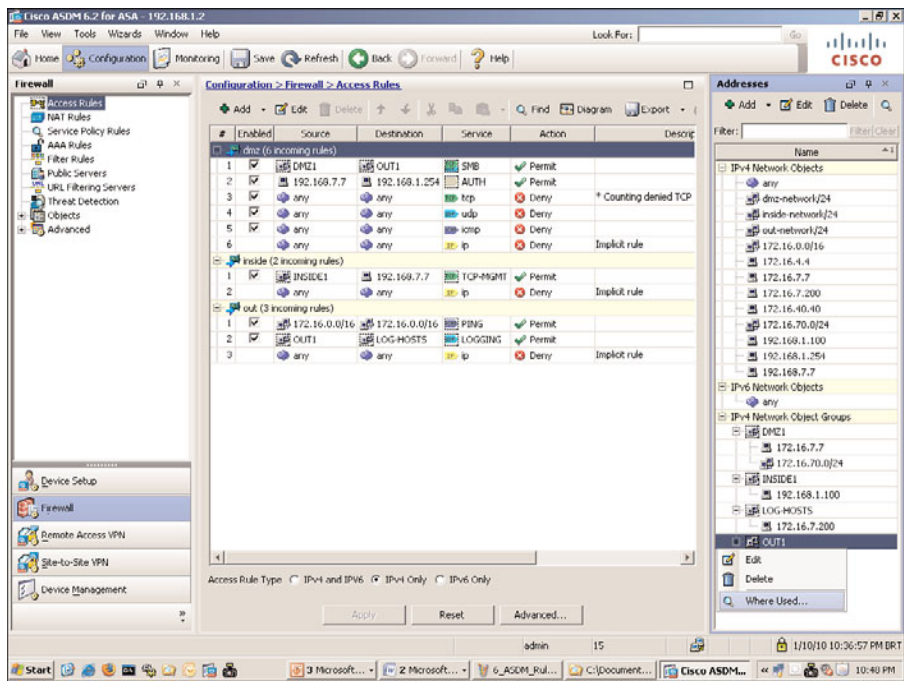


Figure 7-10 Access Rules Table in ASDM

Tip The View menu also enables the **service object-groups** display on the right pane and perform searches (*where used*) in a completely analogous fashion for what was done for network objects.

Figure 7-11 documents how to instruct ASDM to expand the components of each **object-group** in its Rules Table. It also highlights the default update period of 30 seconds for ACL *bit count* update.

Example 7-30 explores a set of important ACL concepts:

- The **implicit deny** rule at the end of each ACL does not include a hash value, a useful resource for correlating logging messages with the ACEs that generated them. Drawing on the approach proposed in Chapter 4, in which one usage of ACLs is for counting packets, four lines were added to the ACL named DMZ (defined earlier in Example 7-29). These **explicit deny** entries not only include the hash but also help to quickly show the distribution of denied connections among the classic protocols (TCP, UDP, ICMP, and generic IP).

- The example shows how the hash value in a logging message might be used to discover the ACL line that was hit.
- The ACL line number might also be used to filter the output of the **show access-list** command.
- A remark might be added to a particular ACE to describe its purpose.

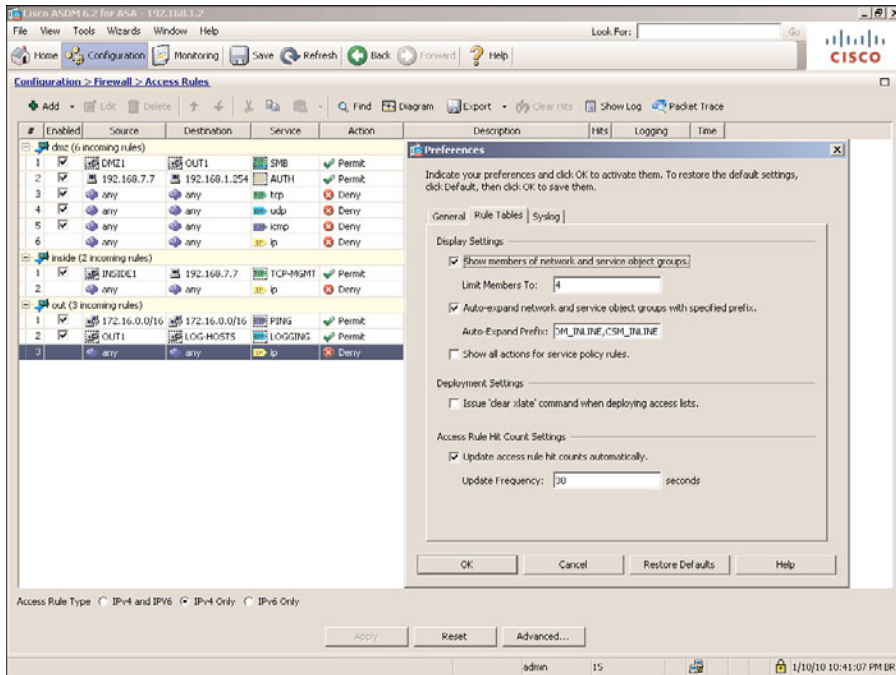


Figure 7-11 Viewing Object-Group Components in ASDM Rules Table

Example 7-30 ACL Maintenance Tasks (1)

! Sample flow denied by the implicit deny rule (no hash present for this ACE)

```
%ASA-4-106023: Deny icmp src dmz:172.16.7.7 dst out:172.16.4.4 (type 8, code 0) by
access-group "DMZ" [0x0, 0x0]
```

!

! Adding explicit rules at the end of the ACL to enable counting of denied flows

```
ASA2(config)# access-list DMZ line 3 extended deny tcp any any
ASA2(config)# access-list DMZ line 4 extended deny udp any any
ASA2(config)# access-list DMZ line 5 extended deny icmp any any
ASA2(config)# access-list DMZ line 6 extended deny ip any any
```

!

! Clearing the counters (hit counts) for an ACL

```
ASA2# clear access-list DMZ counters
```

```

!
! Sample flow denied by explicit rule (hash is now present)
%ASA-4-106023: Deny icmp src dmz:172.16.7.7 dst out:172.16.4.4 (type 8, code 0) by
access-group "DMZ" [0xedd51436, 0x0]
!
! Viewing the hit counts for an ACE using the hash information
ASA2# show access-list DMZ | include 0xedd51436
access-list DMZ line 5 extended deny icmp any any (hitcnt=1) 0xedd51436
!
! Viewing the hit counts for an ACE using the line number
ASA2# show access-list DMZ | include line 5
access-list DMZ line 5 extended deny icmp any any (hitcnt=1) 0xedd51436
!
! Adding a description (remark) for an Access Control Entry
ASA2(config)# access-1 DMZ line 3 remark * Counting denied TCP connection attempts

```

Tip The configuration of a **remark** actually consumes a line number and moves down the previously configured ACEs. After adding the **remark** in Example 7-30, the **permit tcp any any** statement has its line number changed to 4:

```

ASA2# show access-list DMZ | include tcp any any
access-list DMZ line 4 extended deny tcp any any (hitcnt=88) 0x60253772.

```

Logging is enabled by default for ACEs configured with the **deny** action (both implicit and explicit rules). Example 7-31 teaches how to configure logging for an ACE that has **permit** as its action. The logging level (6 by default) can be changed to any of the possible values (0 to 7). The example also registers the occurrence of the *first bit* of a **permit** ACE and the logging message containing the aggregated count of hits sent after the 300 seconds interval.

Example 7-31 ACL Maintenance Tasks (2)

```

! Enabling logging for a given 'permit' ACE
access-list OUT line 1 extended permit icmp 172.16.0.0 255.255.0.0 172.16.0.0
255.255.0.0 object-group PING log 5
access-list OUT line 2 extended permit udp object-group OUT1 object-group LOG-
HOSTS object-group LOGGING log 6
!
! Logs generated by the first occurrence of a 'permit' statement
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.4.41/0 gaddr
172.16.7.7/0 laddr 172.16.7.7/0
%ASA-6-106100: access-list OUT permitted icmp out/172.16.4.41(8) ->
dmz/172.16.7.7(0) hit-cnt 1 first hit [0x61f4cfd9, 0x7463e142]
!
! Log generated by a 'permit' statement after the default 300 seconds interval

```

```
%ASA-6-106100: access-list OUT permitted icmp out/172.16.4.41(8) ->
dmz/172.16.7.7(0) hit-cnt 5300-second interval [0x61f4cfd9, 0x7463e142]
```

Figure 7-12 complements Example 7-31 by showing how to correlate message types 106100 (*explicit permit*) and 106023 (*explicit deny*), as shown on the ASDM log viewer, with the entries in the ASDM Rules Table responsible for their generation.

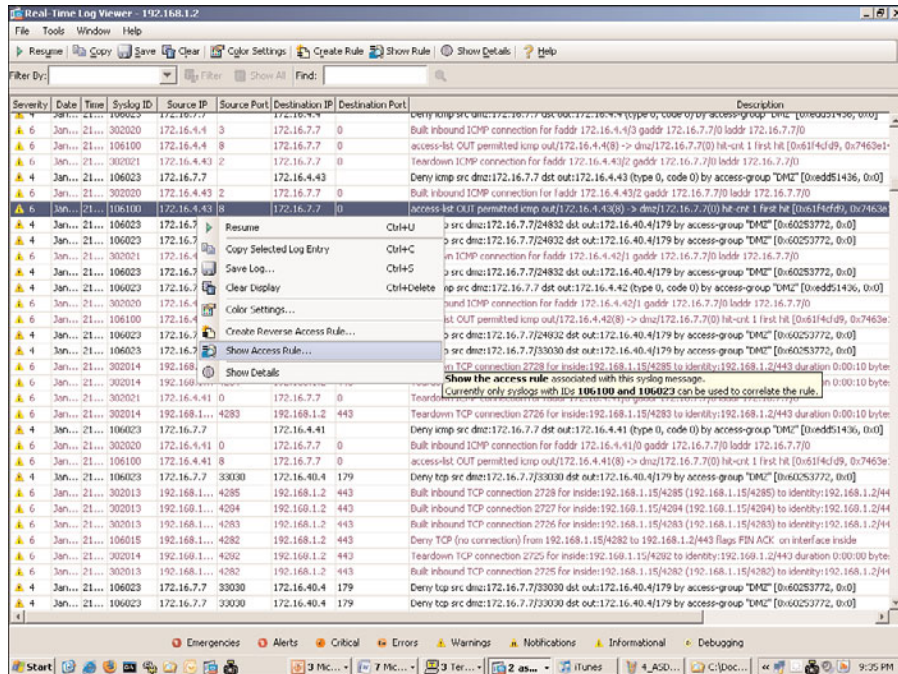


Figure 7-12 From ASDM Log Viewer to Rules Table

Example 7-32 assembles some other sample ACL maintenance tasks:

- Beginning ACL display from a particular line (using the line number information)
- Disabling an ACE (without removing it from the configuration) by adding the inactive keyword
- Removing a given ACE or deleting the entire ACL
- Renaming an existent ACL

Example 7-32 ACL Maintenance Tasks (3)

```
! Beginning ACL display from a particular line
ASA2# show access-list DMZ | begin line 6
```

```

access-list DMZ line 6 extended deny icmp any any (hitcnt=28) 0xedd51436
access-list DMZ line 7 extended deny ip any any (hitcnt=0) 0x93ba46e0
!
! Disabling an ACE with the 'inactive' keyword
ASA2(config)# access-list DMZ line 7 extended deny ip any any inactive
!
ASA2(config)# show access-list | include inactive
access-list DMZ line 7 extended deny ip any any inactive (hitcnt=0) (inactive)
0x93ba46e0
!
! Deleting an specific ACE
ASA2(config)# no access-list DMZ line 7 extended deny ip any any
!
ASA2(config)# show access-list DMZ | begin line 6
access-list DMZ line 6 extended deny icmp any any (hitcnt=28) 0xedd51436
!
! Viewing an ACL in the 'running-config' (before trying to delete it)
ASA2# show running-config access-list EVERYTHING2
access-list EVERYTHING2 extended permit ip any any
!
! Deleting an entire ACL
ASA2(config)# clear configure access-list EVERYTHING2
!
! Confirming the ACL removal
ASA2# show running-config access-list EVERYTHING2
ERROR: access-list <EVERYTHING2> does not exist
!
! Renaming an existent ACL
ASA2(config)# access-list DENY-ALL1 rename DENY-ANY1

```


Summary

This chapter analyzed the creation of connections through ASA-based Firewalls in scenarios where NAT is not in use, which can be referred to as the **no nat-control** model. The concept of *Security Level* lies at the core of the ASA algorithm, reflecting the relative perception of trustworthiness of each firewall interface. According to the relative levels of the two interfaces between which a connection is created, the access type is classified as *outbound*, *inbound*, or *same security* access.

The chapter also covered the following:

- The differences between *implicit* and *explicit* rules.
- The elements of the TCP finite state machine covered by the ASA Stateful Inspection algorithm.
- TCP Sequence Number Randomization.
- The creation of some **set connection** statements to modify the default behavior of service policies was exemplified.
- Types of **object-groups** and how they make work much more structured when dealing with ACLs (mainly the large ones).
- Important ACL maintenance tasks.

The next chapter complements the contents of this chapter by studying connections with translations.

This page intentionally left blank

Through ASA Using NAT

This chapter covers the following topics:

- The Nat-control Model
- Outbound NAT analysis
- Address publishing for inbound access
- Inbound NAT analysis
- Dual NAT
- Disabling TCP sequence number randomization
- Defining connection limits with NAT rules

“Wheresoever we have spoken plainly, there we have spoken nothing. But where we have used riddles and figures, there we have hidden the truth”—Rosarium Philosophorum

Network Address Translation (NAT) is a widespread technique for creating mappings between internal and external address spaces. The simplest NAT mode of operation establishes a one-to-one correspondence between the *real* source address and the *virtual* (or translated) source address. Some other more sophisticated modes enable changing the destination addresses and building many-to-one translations.

The various NAT types might be employed to accomplish several different tasks. Some of the most relevant ones follow:

- **Hiding private addresses from the global Internet:** The private addresses defined by RFC 1918 are commonly used inside companies but cannot be routed on the Internet. NAT lends itself to the task of confining these IPs to the organization boundaries.
- **Mitigating the problem of IPv4 address depletion:** The number of publicly routable addresses assigned to companies is frequently much smaller than the amount of

internal hosts, thus creating the need for a technology that enables many-to-one mappings. The classic solution is called *Dynamic Port Address Translation* (*Dynamic PAT*).

- **Concealing the details of the internal network from the outside world:** Dynamic PAT not only handles the reduction of valid Internet addresses, but also makes it possible to define *unidirectional* translations for hosts that should not be accessible from external sources.
- **Interconnecting networks with overlapping addresses:** Almost every company uses private addresses for internal hosts. For organization acquisitions and mergers, conflicting addresses (that still need to access each other) might appear. NAT is a possible solution in such a scenario.

NAT involves two fundamental (and complementary) processes:

- Replacing the original address by a virtual address and creating an entry that reflects this mapping in a *translation table*. For NAT to work properly, the virtual address should be routable on the destination network.
- Searching the translation table to map the return packets (sent to the virtual address) back to the original address. (This is called *untranslating*.)

This chapter is devoted to the analysis of the numerous NAT categories supported by ASA and to study how the concept of *connections*, which were discussed in Chapter 7, “Through ASA Without NAT,” relate with the connectivity provided by *translations*.

Before presenting the actual contents, this chapter deals with the NAT syntaxes used in ASA pre-8.3 releases. The command and philosophy changes of release 8.3 are summarized in Appendix A, “NAT and ACL Changes in ASA 8.3.” Although 8.3 introduces a new design for the NAT functionality, it does not modify the available NAT categories.

Nat-Control Model

As studied in the previous chapter, the default NAT configuration, after release 7.0 of PIX and ASA, is the model denominated **no nat-control**, meaning that traffic can flow through the firewall with no requirement of any NAT rule. When this model is chosen, NAT becomes an optional attribute for both outbound and inbound classes of access.

This chapter examines the model known as **nat-control**, which represents the single choice of operation mode for PIX Firewalls before the new era brought by release 7.0. Between 7.0 (first ASA release) and 8.2, **nat-control** is optional.

In the **nat-control** model, the ASA algorithm enables connectivity to be established through the firewall only if a clear response for NAT usage is provided during configuration. This task can be accomplished in two basic ways:

- By configuring one of the main categories of NAT (dynamic, static, policy)
- By explicitly choosing a *NAT Bypass* mechanism in scenarios where NAT is not needed (or desired)

At this point, even though **no nat-control** can be considered the generalized NAT Bypass solution, *Identity NAT* and *NAT Exemption* do provide a conditional NAT Bypass functionality for **nat-control** scenarios.

When **nat-control** is enabled, the outbound access rules can be summarized as follows:

- Step 1.** Packets traversing from a higher to a lower sec-lvl interface are required to match some type of NAT rule. This may be considered a *connectivity requirement*.
- Step 2.** The NAT Rule mentioned in Step 1 might be eventually a **nat 0** statement, which is materialized using either the Identity NAT or NAT Exemption bypass mechanisms. The requirement in this case is to give an explicit answer for NAT usage to the ASA algorithm. If the answer provided is *no*, the firewall applies the **implicit deny** rule associated with the situations in which no *translation group* is found.
- Step 3.** If the ASA algorithm finds a translation group, outbound connection initiation is allowed. The implicit permit for outbound connections does not have precedence over the NAT requirement.

On the other hand, the inbound access rules for a **nat-control** environment follow:

- If it is necessary to make an internal address visible for hosts located on a lower sec-lvl interface, some sort of *address publishing* NAT rule must be built. It is convenient to reinforce that this publishing effect arises from the usage of some type of univocal translation rule. Identity NAT and dynamic rules (such as Dynamic PAT) are not suitable for publishing an internal address on the outside.
- The types of translation rules that can be used for publishing internal addresses are Static NAT, Static PAT (*Port Redirection*), Static Policy NAT, and NAT Exemption.
- To allow the initiation of an inbound connection, an explicit *Access Control Entry*, defining the types of traffic allowed through, must be created. The destination address in this ACE should be the published address of the internal host.

Note Despite that NAT typically brings more complexity to the network environment, it might contribute to the *defense-in-depth* security philosophy. When the **nat-control** model is in place, if no translation group is found, an ACL permission is not enough to enable connections to be established through ASA.

Example 8-1 shows how to verify that **nat-control** is enabled and highlights the implicit deny behavior associated with this model when hosts on the *dmz* interface try to connect to hosts reachable through the *out* interface.

Example 8-1 *Enabling NAT-Control*

```

ASA2#show running-config nat-control
nat-control
!
! Implicit deny associated with NAT-Control (from 'dmz' to 'out')
ASA2# show nat dmz out
match ip dmz any out any
no translation group, implicit deny
policy_hits = 0

```

Outbound NAT Analysis

The concepts of inbound and outbound access discussed throughout the previous chapter are used frequently in the current one. The main distinction is that, whenever **nat-control** is configured, connections are allowed to be built only on top of *translations* (which are also referred to as *xlates* in the nomenclature of the ASA algorithm).

The baseline configuration of Example 8-2 is associated with the reference topology of Figure 8-1 and is used to illustrate the various categories of NAT supported by ASA.

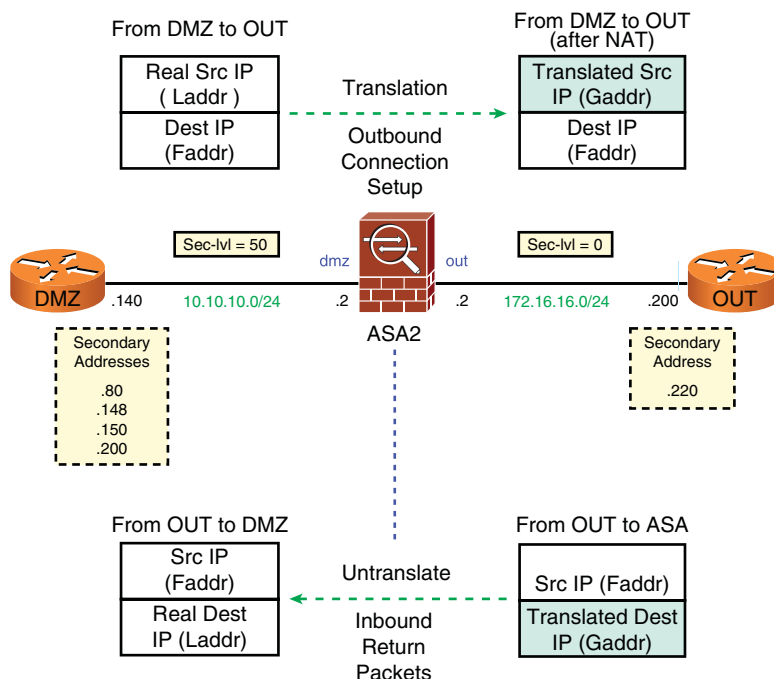


Figure 8-1 *Reference Topology for Examples in This Chapter*

Example 8-2 *Baseline Configuration*

```

object-group icmp-type PING
  icmp-object echo
  icmp-object echo-reply
!
object-group network TRANSLATED
  network-object 172.16.16.0 255.255.255.0
  network-object 10.10.10.0 255.255.255.0
  network-object 172.17.17.0 255.255.255.0
  network-object 172.18.18.0 255.255.255.0
!
object-group network OUTSIDE
  network-object 172.16.16.0 255.255.255.0
!
access-list OUT extended permit icmp object-group OUTSIDE object-group TRANSLATED
object-group PING log notifications interval 120
!
access-group OUT in interface out

```

Note Recall from Chapter 7 that ICMP is inherently stateless, with the echo replies (from the outside) being treated as inbound connections. The outside hosts always reply to the *translated* source address, and this is the rationale for using the **log** option in the ACL named OUT. Whenever an ACE is hit, a log message is sent, thus revealing the address seen on the outside (after translation takes place).

Note The **debug icmp trace** command is useful for the analysis of ICMP translations and connections through the firewall. It is extensively employed in the examples that follow.

Tip Before starting any of the examples presented, the commands **clear access-list OUT counters**, **clear xlate**, **clear nat counters**, and **clear local-host all** were issued.

Dynamic NAT

The first option to be analyzed is *Dynamic NAT*, which maps a group of inside (*real*) source addresses (defined by the **nat** command) to a pool of translated source addresses (defined by the **global** command). The parameter that binds these two pools is a non-zero value for the *pool number*, as shown in Example 8-3.

ASA randomly selects an address from the **global** pool and associates it with the connection initiator address (which belongs to the **nat** pool). After the **xlate** timeout expires, this mapping is removed from the translation table, and there is no guarantee that the original inside host (*laddr*) will be assigned the same global address (*gaddr*) after a new outbound connection attempt.

Two important situations are documented in Example 8-3:

- **A match in the Dynamic NAT Rule:** The source address 10.10.10.150 falls within the 10.10.10.128/25 range and is mapped to a random address in the **global** pool (172.16.16.253, in this case). The outbound ICMP connection (associated with the echo request) is built on top of the dynamic translation. The **show xlate debug** command is useful for displaying details about active translations. It also reveals the pertinent *translation flags* and the *xlate timeout* values.
- **A packet that does not match the NAT Rule:** Originates in the source address 10.10.10.80, which is out of the 10.10.10.128/25 subnet. Considering that there is no other NAT rule in place, a translation is not created, the packet is dropped, and a syslog message is issued, stating that *no translation group was found*.

Example 8-3 Dynamic NAT Configuration and Verification

```
! The pool-number parameter ( 2, in this case) binds the 'global' and 'nat' com-
mands
nat (dmz) 2 10.10.10.128 255.255.255.128
global (out) 2 172.16.16.129-172.16.16.254 netmask 255.255.255.128
!
! 'dmz' host (10.10.10.150) pings host 172.16.16.200 on the 'out' interface
%ASA-6-305009: Built dynamic translation from dmz:10.10.10.150 to out:172.16.16.253
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.200/0
gaddr 172.16.16.253/25 laddr 10.10.10.150/25
ICMP echo request from dmz:10.10.10.150 to out:172.16.16.200 ID=25 seq=0 len=72
ICMP echo request translating dmz:10.10.10.150 to out:172.16.16.253
%ASA-6-106100: access-list OUT permitted icmp out/172.16.16.200(0) ->
dmz/172.16.16.253(0) hit-cnt 1 first hit [0x4f98dc69, 0x24945c78]
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.200/0
gaddr 172.16.16.253/25 laddr 10.10.10.150/25
ICMP echo reply from out:172.16.16.200 to dmz:172.16.16.253 ID=25 seq=0 len=72
ICMP echo reply untranslating out:172.16.16.253 to dmz:10.10.10.150
!
! Viewing details about the active translations (xlates)
ASA2# show xlate debug
2 in use, 5 most used
Flags: D - DNS, d - dump, I - identity, i - dynamic, n - no random,
      r - portmap, s - static
NAT from dmz:10.10.10.150 to out:172.16.16.253 flags i idle 0:00:41 timeout 3:00:00
NAT from dmz:10.10.10.151 to out:172.16.16.232 flags i idle 0:00:11 timeout 3:00:00
!
! Host 10.10.10.80 does not match any NAT Rule and is not allowed to go through
DMZ#ping 172.16.16.200 source 10.10.10.80 repeat 1
Success rate is 0 percent (0/1)
```

```
%ASA-3-305005: No translation group found for icmp src dmz:10.10.10.80 dst
out:172.16.16.200 (type 8, code 0)
```

Note Dynamic NAT is a *unidirectional* type of translation that fits well for private hosts that are not intended to be visible from the outside and need to reach only external networks.

Note Dynamic NAT might be important for unidirectional translation of protocols that, unlike TCP and UDP, do not include the concept of Layer 4 port.

Dynamic PAT

Dynamic PAT maps a group of inside (*real*) source addresses (defined by a **nat** command) to a single translated source address (defined with a **global** statement). To individualize each internal host that matches the PAT Rule and to correctly deliver the return packets, ASA uses a combination of the **global** address and a source port. The intrinsic many-to-one nature of PAT makes it an adequate solution for mitigating the public IPv4 address exhaustion problem.

Figure 8-2 illustrates PAT operation for two internal clients, C1 and C2, that need to reach the outside server S1. It should be noted that the figure depicts a situation in which two hosts select equal source ports for connection setup. At this point, you should answer the question: *If ASA simply preserved the source port (as it does for NAT), what would happen to return packets from the same outside host?*

The **nat** and **global** commands used in Example 8-4 establish that hosts arriving at the *dmz* interface with source addresses belonging to the 10.10.10.0/24 subnet should be translated to the global address 172.16.16.126, when connecting to a destination reachable through the *out* interface.

Example 8-4 illustrates PAT in action:

- It characterizes that the ICMP ID from the real source address (*laddr*) is treated as the source port (similar to UDP and TCP), and mapped to a translated source port, which becomes the ICMP ID after translation. An ICMP packet from host 10.10.10.140 with ID 9 was mapped to the *gaddr* 172.16.16.126 with port 55957.
- The **show xlate debug** command registers sample ICMP and TCP PAT translations. It also shows the *r* flag associated with *portmap* operations. It is an instructive exercise to compare this output with that of Example 8-3.

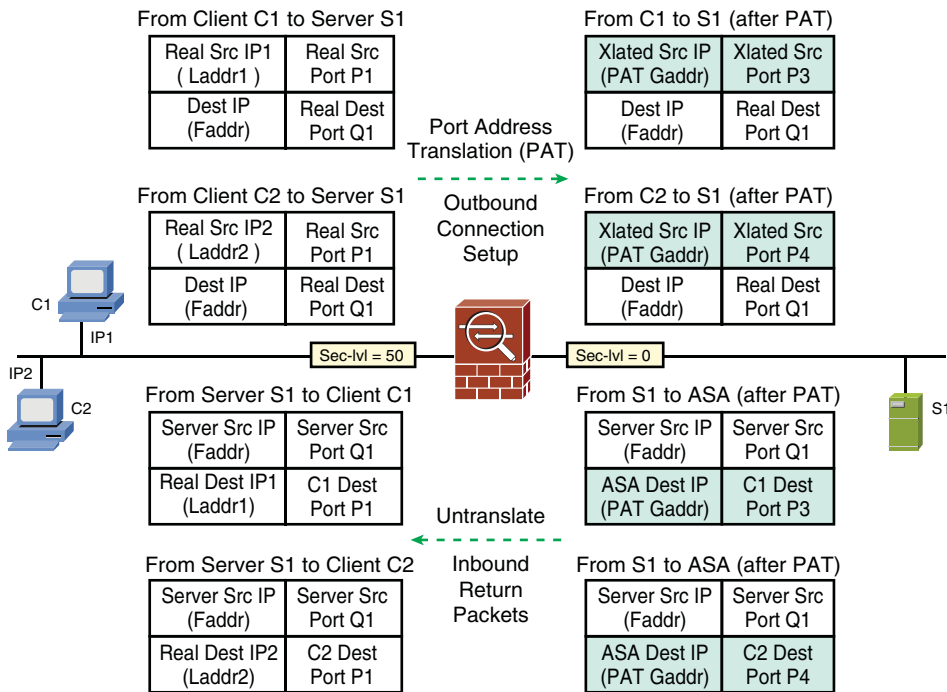


Figure 8-2 Illustrating Dynamic PAT Operation

Example 8-4 Dynamic PAT Configuration and Verification

```
! Configuring PAT using the global address 172.16.16.126
ASA2(config)# nat (dmz) 1 10.10.10.0 255.255.255.0
ASA2(config)# global (out) 1 172.16.16.126
INFO: Global 172.16.16.126 will be Port Address Translated
!
! Host with laddr 10.10.10.140 pings host with faddr 172.16.16.200
%ASA-6-305011: Built dynamic ICMP translation from dmz:10.10.10.140/9
to out:172.16.16.126/55957
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.200/0
gaddr 172.16.16.126/55957 laddr 10.10.10.140/9
ICMP echo request from dmz:10.10.10.140 to out:172.16.16.200 ID=9 seq=0 len=72
ICMP echo request translating dmz:10.10.10.140/9 to out:172.16.16.126/55957
%ASA-6-106100: access-list OUT permitted icmp out/172.16.16.200(0) ->
dmz/172.16.16.126(0) hit-cnt 1 first hit [0x4f98dc69, 0x24945c78]
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.200/0 gaddr
172.16.16.126/55957 laddr 10.10.10.140/9
ICMP echo reply from out:172.16.16.200 to dmz:172.16.16.126 ID=55957 seq=0 len=72
ICMP echo reply untranslating out:172.16.16.126/55957 to dmz:10.10.10.140/9
!
! Viewing details about the active translations (xlates)
ASA2# show xlate debug
3 in use, 5 most used
```

```

Flags: D - DNS, d - dump, I - identity, i - dynamic, n - no random,
r - portmap, s - static
ICMP PAT from dmz:10.10.10.140/9 to out:172.16.16.126/55957 flags ri idle 0:00:07
timeout 0:00:30
ICMP PAT from dmz:10.10.10.80/34 to out:172.16.16.126/45518 flags ri idle 0:00:05
timeout 0:00:30
TCP PAT from dmz:10.10.10.140/21506 to out:172.16.16.126/39442 flags ri idle
0:00:34 timeout 0:00:30

```

Example 8-5 registers the relevant stages of a sample *Packet Tracer* simulation for an UDP connection from *dmz* host 10.10.10.90/4000 to *out* host 172.16.16.220 on port 53. In this case, the source port 4000 is mapped to port 25393 after PAT. Phase 5 unveils lots of data about the matched NAT Rule.

Example 8-5 Using Packet Tracer for UDP

```

! Building an UDP Connection on top of a dynamic PAT Translation
ASA2# packet-tracer input dmz udp 10.10.10.90 4000 172.16.16.220 53
Phase: 1
%ASA-6-305011: Built dynamic UDP translation from dmz:10.10.10.90/4000 to
out:172.16.16.126/25393
%ASA-6-302015: Built outbound UDP connection 142 for out:172.16.16.220/53
(172.16.16.220/53) to dmz:10.10.10.90/4000 (172.16.16.126/25393)
Type: FLOW-LOOKUP
Subtype:
Result: ALLOW
[ output suppressed ]
Phase: 5
Type: NAT
Subtype:
Result: ALLOW
Config:
nat (dmz) 1 10.10.10.0 255.255.255.0
nat-control
match ip dmz 10.10.10.0 255.255.255.0 out any
dynamic translation to pool 1 (172.16.16.126)
translate_hits = 1, untranslate_hits = 0
Additional Information:
Dynamic translate 10.10.10.90/4000 to 172.16.16.126/25393 using netmask
255.255.255.255
[ output suppressed ]
Action: allow

```

Note Applications that embed the IP address information in *upper layer* headers tend not to behave well when subject to NAT. (It is even worse when PAT is enabled.) To properly handle these situations, the device in charge of the translation needs to understand the protocol characteristics and compensate for the IP address mismatch between L3 and upper headers. Some samples are presented in Chapter 12, “Application Inspection.”

In Example 8-6, the NAT rules of Examples 8-3 and 8-4 are configured simultaneously. The `show nat` command clearly reveals that Dynamic NAT has precedence over Dynamic PAT. As a result, in situations in which there is overlapping between the NAT pool definitions, Dynamic NAT will be matched first. For instance, a host belonging to 10.10.10.128/25 (the higher half of subnet 10.10.10.0/24) is translated to an address within the pool **global 2**. Hosts in the 10.10.10.0/25 fall only under the PAT rule (**global 1**), where, as source addresses that are not on 10.10.10.0/24 are blocked by the implicit deny rule.

Example 8-6 *Dynamic NAT Takes Precedence over Dynamic PAT*

```
ASA2# show nat dmz out
match ip dmz 10.10.10.128 255.255.255.128 out any
  dynamic translation to pool 2 (172.16.16.129 - 172.16.16.254)
    translate_hits = 0, untranslate_hits = 0
match ip dmz 10.10.10.0 255.255.255.0 out any
  dynamic translation to pool 1 (172.16.16.126)
    translate_hits = 0, untranslate_hits = 0
match ip dmz any out any
  no translation group, implicit deny
  policy_hits = 0
```

Note A considerable amount of attention is devoted to clarify the **Order of NAT Processing**, a topic that is often misunderstood. I have seen several customer deployments with large ACL and NAT configurations, in which there was no clear knowledge of the behavior of each NAT category and the associated precedence rules. This may result in translation statements that are never used (because they have a lower priority than some other type of NAT) and lead to the incorrect conclusion that *something is not working properly*.

Identity NAT

Identity NAT is a technique used to avoid that a specific set of hosts ever be translated. Although the real (*laddr*) and global (*gaddr*) addresses are identical, this conditional NAT Bypass mechanism is considered a valid response for the translation requirement when the `nat-control` model is being used. The value *zero* for the *pool number* parameter is used to specify that this set of source addresses should not undergo translation.

Example 8-7 shows that Identity NAT is also considered a dynamic translation option. In this example, a telnet and a ping session, from hosts 10.10.10.140 and 10.10.10.150 respectively, are initiated toward host 172.16.16.200. The **debug icmp trace** output highlights that the local (laddr) and global (gaddr) addresses are equal.

Example 8-7 Identity NAT Configuration and Verification

```

! Configuring Identity NAT
ASA2(config)# nat (dmz) 0 10.10.10.128 255.255.255.192
nat 0 10.10.10.128 will be identity translated for outbound
!
! Host 10.10.10.140 telnets to 172.16.16.200
%ASA-6-305009: Built dynamic translation from dmz:10.10.10.140 to out:10.10.10.140
%ASA-6-302013: Built outbound TCP connection 144 for out:172.16.16.200/23
(172.16.16.200/23) to dmz:10.10.10.140/40450 (10.10.10.140/40450)
!
! Host 10.10.10.150 pings host 172.16.16.200
%ASA-6-305009: Built dynamic translation from dmz:10.10.10.150 to out:10.10.10.150
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.200/0
gaddr 10.10.10.150/36 laddr 10.10.10.150/36
ICMP echo request from dmz:10.10.10.150 to out:172.16.16.200 ID=36 seq=0 len=72
%ASA-6-106100: access-list OUT permitted icmp out/172.16.16.200(0) ->
dmz/10.10.10.150(0) hit-cnt 1 first hit [0x4f98dc69, 0xf6ef56b4]
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.200/0 gaddr
10.10.10.150/36 laddr 10.10.10.150/36
ICMP echo reply from out:172.16.16.200 to dmz:10.10.10.150 ID=36 seq=0 len=72
!
! Viewing details about the active translations (xlates)
ASA2# show xlate debug
2 in use, 5 most used
Flags: D - DNS, d - dump, I - identity, i - dynamic, n - no random,
      r - portmap, s - static
NAT from dmz:10.10.10.150 to out:10.10.10.150 flags iI idle 0:00:21 timeout 3:00:00
NAT from dmz:10.10.10.140 to out:10.10.10.140 flags iI idle 0:00:40 timeout 3:00:00

```

Note The **show xlate debug** reveals that the translation flag *I* is marked, thus indicating that Identity NAT is being used.

Note Similarly to other dynamic translation methods, Identity NAT is unidirectional. Despite that the global and local addresses are identical, a connection to *gaddr* from a lower sec-lvl interface is not permitted.

Example 8-8 is based on the rules configured in Examples 8-3 and 8-7 and was crafted to create an overlapping between Dynamic and Identity NAT Rules within the range 10.10.10.128/26. The **show nat** command reveals that hosts with source addresses belonging to this subnet are identity translated, whereas those from the 10.10.10.192/26 are subject to Dynamic NAT (pool 2). Hosts out of the network 10.10.10.128/25 are implicitly denied.

Example 8-8 Identity NAT Takes Precedence over Dynamic NAT

```
ASA2# show nat dmz out
match ip dmz 10.10.10.128 255.255.255.192 out any
  identity NAT translation, pool 0
  translate_hits = 0, untranslate_hits = 0
match ip dmz 10.10.10.128 255.255.255.128 out any
  dynamic translation to pool 2 (172.16.16.129 - 172.16.16.254)
  translate_hits = 0, untranslate_hits = 0
match ip dmz any out any
  no translation group, implicit deny
  policy_hits = 0
```

Static NAT

Static NAT defines a permanent, one-to-one correspondence between private (*laddr*) and public (*gaddr*) addresses residing on two different interfaces of the firewall. Because the mapping remains unchanged, Static NAT is *bidirectional* in nature, enabling connection initiation from the outside. (If this is the intended behavior, an explicit permission needs to be created by means of an ACL.)

Example 8-9 documents two possible configurations for Static NAT:

- **Host-to-host mapping:** Although displayed in the running-config, the /32 netmask is not necessary while entering the **static** command. (It is the default mask.)
- **Range-to-range mapping:** There should be as many mapped addresses as there are real ones. In the example, a /26 netmask is chosen to create the translations between the real addresses 10.10.10.0/26 and the global addresses 172.17.170/26.

Some important points about this NAT category are noteworthy:

- The first interface that appears in the **static** command is the *private* interface (where real addresses exist), and the first address is the *virtual* (or *mapped*) address. This may sound counterintuitive, but the command syntax was built in this way. It works.
- When the **static** command is entered (in *config mode*), the static translation is immediately built. That is why there is no *built translation* syslog message at connection setup time. The **show xlate debug** command reveals the infinite timeout (0:00:00) that characterizes *statics*.

Example 8-9 *Configuring and Verifying the Operation of Static NAT*

```

! Static translation for an individual host (/32 is the default netmask)
ASA2(config)# static (dmz,out) 172.16.16.140 10.10.10.140
%ASA-6-305009: Built static translation from dmz:10.10.10.140 to out:172.16.16.140
!
! Sample static translation for a /26 subnet
ASA2(config)# static (dmz,out) 172.17.17.0 10.10.10.0 netmask 255.255.255.192
%ASA-6-305009: Built static translation from dmz:10.10.10.0 to out:172.17.17.0
!
! Searching for static entries within the running-config
ASA2# show running-config static
static (dmz,out) 172.16.16.140 10.10.10.140 netmask 255.255.255.255
static (dmz,out) 172.17.17.0 10.10.10.0 netmask 255.255.255.192
!
! Verifying the details about the static translations
ASA2# show xlate debug
2 in use, 5 most used
Flags: D - DNS, d - dump, I - identity, i - dynamic, n - no random,
      r - portmap, s - static
NAT from dmz:10.10.10.140 to out:172.16.16.140 flags s idle 0:17:43 timeout 0:00:00
NAT from dmz:10.10.10.0 to out:172.17.17.0 flags s idle 0:03:25 timeout 0:00:00
!
! Statics in action (the connections are built on top of the pre-existent xlates)
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.200/0
gaddr 172.16.16.140/67 laddr 10.10.10.140/67
ICMP echo request from dmz:10.10.10.140 to out:172.16.16.200 ID=67 seq=0 len=72
ICMP echo request translating dmz:10.10.10.140 to out:172.16.16.140
%ASA-6-106100: access-list OUT permitted icmp out/172.16.16.200(0) ->
dmz/172.16.16.140(0) hit-cnt 1 first hit [0x4f98dc69, 0x24945c78]
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.200/0
gaddr 172.16.16.140/67 laddr 10.10.10.140/67
ICMP echo reply from out:172.16.16.200 to dmz:172.16.16.140 ID=67 seq=0 len=72
ICMP echo reply untranslating out:172.16.16.140 to dmz:10.10.10.140

```

Policy NAT

Policy NAT is a kind of *conditional translation*, for it takes into account not only the source (*laddr*) but also the destination addresses (*faddr*) to determine the mapped address (*gaddr*). The criteria for translation are defined by ACLs.

Static Policy NAT

This method is similar to Static NAT in the sense that it creates one-to-one mappings. The particularity in this case relates to the fact that the mapped address may be different, depending on the destination host to which packets from the real address are being sent.

Example 8-10 shows a policy with the following definitions:

- Packets from real address (*laddr*) 10.10.10.148 and destined to *faddr* 172.16.16.200 are translated to 172.18.18.148 on *out* interface.
- Packets from 10.10.10.148 that are destined to a host other than 172.16.16.200 are mapped to 172.16.16.148. The reader should compare the *ping* packet from 10.10.10.148 to host 172.16.16.200 with that destined to 172.16.16.220.

Although the firewall complains about the overlapping between *Static* and *Static Policy* methods, it creates the translation. Because these overlapping rules are checked in order of configuration, the Policy NAT in this example should be defined first. (It is more specific.)

Example 8-10 *Static Policy NAT*

```
! Source 10.10.10.148 when talking to 172.16.16.200 is translated to 172.18.18.148
ASA2(config)# access-list STATIC-POLICY1 extended permit ip host 10.10.10.148 host
172.16.16.200
ASA2(config)# static (dmz,out) 172.18.18.148 access-list STATIC-POLICY1
%ASA-6-305009: Built static translation from dmz:10.10.10.148 to out
(STATIC-POLICY1):172.18.18.148
!
! For other destinations source address 10.10.10.148 is translated to
172.16.16.148
ASA2(config)# static (dmz,out) 172.16.16.148 10.10.10.148 netmask 255.255.255.255
INFO: overlap with existing static
dmz:10.10.10.148 to out:172.18.18.148 netmask 255.255.255.255
%ASA-6-305009: Built static translation from dmz:10.10.10.148 to out:172.16.16.148
!
! Ping from 10.10.10.148 to 172.16.16.200
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.200/0
gaddr 172.18.18.148/3 laddr 10.10.10.148/3
ICMP echo request from dmz:10.10.10.148 to out:172.16.16.200 ID=3 seq=0 len=72
ICMP echo request translating dmz:10.10.10.148 to out:172.18.18.148
%ASA-6-106100: access-list OUT permitted icmp out/172.16.16.200(0) ->
dmz/172.18.18.148(0) hit-cnt 1 first hit [0x4f98dc69, 0x3a72763]
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.200/0 gaddr
172.18.18.148/3 laddr 10.10.10.148/3
ICMP echo reply from out:172.16.16.200 to dmz:172.18.18.148 ID=3 seq=0 len=72
ICMP echo reply untranslating out:172.18.18.148 to dmz:10.10.10.148
!
! Ping from 10.10.10.148 to 172.16.16.220
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.220/0
gaddr 172.16.16.148/1 laddr 10.10.10.148/1
ICMP echo request from dmz:10.10.10.148 to out:172.16.16.220 ID=1 seq=0 len=72
ICMP echo request translating dmz:10.10.10.148 to out:172.16.16.148
```

```
%ASA-6-106100: access-list OUT permitted icmp out/172.16.16.220(0) ->
dmz/172.16.16.148(0) hit-cnt 1 first hit [0x4f98dc69, 0x24945c78]
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.220/0 gaddr
172.16.16.148/1 laddr 10.10.10.148/1
ICMP echo reply from out:172.16.16.220 to dmz:172.16.16.148 ID=1 seq=0 len=72
ICMP echo reply untranslating out:172.16.16.148 to dmz:10.10.10.148
!
ASA2# show xlate debug
2 in use, 5 most used
Flags: D - DNS, d - dump, I - identity, i - dynamic, n - no random,
      r - portmap, s - static
NAT from dmz:10.10.10.148 to out(STATIC-POLICY1):172.18.18.148 flags s idle 1:16:56
timeout 0:00:00
NAT from dmz:10.10.10.148 to out:172.16.16.148 flags s idle 1:09:57 timeout
0:00:00
```

Dynamic Policy NAT

This technique is analogous to Dynamic NAT but takes into consideration the destination address (*faddr*) in addition to *laddr*, before electing the mapped address (*gaddr*).

Example 8-11 shows that hosts in the range 10.10.10.128/27 are translated to the **global** pool 4 when sending packets to *out* host 172.16.16.200.

Example 8-11 *Dynamic Policy NAT*

```
! Packets from 10.10.10.128/27 travelling to host 172.16.16.200 undergo Policy NAT
access-list DYN-POLICY1 extended permit ip 10.10.10.128 255.255.255.224 host
172.16.16.200
!
nat (dmz) 4 access-list DYN-POLICY1
global (out) 4 172.17.17.129-172.17.17.158 netmask 255.255.255.224
!
! Policy NAT in action for a ping from 10.10.10.140 to 172.16.200.200
%ASA-6-305009: Built dynamic translation from dmz:10.10.10.140 to out
(DYN-POLICY1):172.17.17.138
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.200/0
gaddr 172.17.17.138/9 laddr 10.10.10.140/9
ICMP echo request from dmz:10.10.10.140 to out:172.16.16.200 ID=9 seq=0 len=72
ICMP echo request translating dmz:10.10.10.140 to out:172.17.17.138
%ASA-6-106100: access-list OUT permitted icmp out/172.16.16.200(0) ->
dmz/172.17.17.138(0) hit-cnt 1 first hit [0x4f98dc69, 0xe6015120]
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.200/0 gaddr
172.17.17.138/9 laddr 10.10.10.140/9
ICMP echo reply from out:172.16.16.200 to dmz:172.17.17.138 ID=9 seq=0 len=72
ICMP echo reply untranslating out:172.17.17.138 to dmz:10.10.10.140
!
```



```
ASA2# show xlate debug
1 in use, 5 most used
Flags: D - DNS, d - dump, I - identity, i - dynamic, n - no random,
      r - portmap, s - static
NAT from dmz:10.10.10.140 to out(DYN-POLICY1):172.17.17.138 flags i idle 0:00:21
timeout 3:00:00
```

Dynamic Policy PAT

This method is similar to Dynamic PAT but takes into account the destination address (*faddr*) and the *laddr* before choosing the mapped address (*gaddr*).

The translation policy of Example 8-12 states that hosts from the 10.10.10.128/26 range should be mapped to the global address 172.16.16.125, when connecting to the *faddr* 172.16.16.200.

Example 8-12 Dynamic Policy PAT

```
! Policy PAT for source subnet 10.10.10.128/26 going to host 172.16.16.200
access-list DYN-POLICY2 extended permit ip 10.10.10.128 255.255.255.192 host
172.16.16.200
!
ASA2(config)# nat (dmz) 3 access-list DYN-POLICY2
ASA2(config)# global (out) 3 172.16.16.125
INFO: Global 172.16.16.125 will be Port Address Translated
!
! Host 10.10.10.140 pings 172.16.16.200 (Policy PAT comes into play)
%ASA-6-305011: Built dynamic ICMP translation from dmz:10.10.10.140/2 to out
(DYN-POLICY2):172.16.16.125/25839
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.200/0
gaddr 172.16.16.125/25839 laddr 10.10.10.140/2
ICMP echo request from dmz:10.10.10.140 to out:172.16.16.200 ID=2 seq=0 len=72
ICMP echo request translating dmz:10.10.10.140/2 to out:172.16.16.125/25839
%ASA-6-106100: access-list OUT permitted icmp out/172.16.16.200(0) ->
dmz/172.16.16.125(0) hit-cnt 1 first hit [0x6671365f, 0x24945c78]
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.200/0
gaddr 172.16.16.125/25839 laddr 10.10.10.140/2
ICMP echo reply from out:172.16.16.200 to dmz:172.16.16.125 ID=25839 seq=0 len=72
ICMP echo reply untranslating out:172.16.16.125/25839 to dmz:10.10.10.140/2
!
! Verifying details about the dynamic xlate
ASA2# show xlate debug
1 in use, 5 most used
Flags: D - DNS, d - dump, I - identity, i - dynamic, n - no random,
      r - portmap, s - static
```

```
ICMP PAT from dmz:10.10.10.140/2 to out(DYN-POLICY2):172.16.16.125/25839 flags ri
idle 0:00:06 timeout 0:00:30
```

Note The precedence rules for Dynamic Policy PAT and Dynamic Policy NAT depend on the order in which they were defined. Therefore, it is always convenient to configure the most specific source range first.

NAT Exemption

In the opposite range of the spectrum from Identity NAT, this conditional NAT Bypass technique is *bidirectional* in nature. NAT Exemption is configured using the **nat 0 access-list** command and states that translation should not be performed between sources and destinations defined in this **access-list**.

Example 8-13 illustrates the operation of NAT Exemption. It should be observed that

- No translation is created at anytime: Compare this behavior with that of Identity NAT in Example 8-7.
- The **permit** statements in the ACL define the traffic that should be exempted from NAT. Likewise, **deny** statements mean that the traffic should be excluded from exemption (meaning that it should be translated). Because ACLs have an implicit deny at their ends, any explicit deny rules should be configured before explicit permits.

Example 8-14 shows how to perform a static translation for host 10.10.10.140, which falls under the 10.10.10.128/28 NAT Exemption range. A **deny** statement is necessary in this case, because NAT Exemption takes precedence over any other sort of NAT rule.

Example 8-13 NAT Exemption

```
! Connections between 10.10.10.128/28 and 172.16.16.0/24 are exempted from NAT
access-list NONAT extended permit ip 10.10.10.128 255.255.255.240 172.16.16.0
255.255.255.0
!
nat (dmz) 0 access-list NONAT
!
! Host 10.10.10.140 pings host 172.16.200.200 (no translation is performed)
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.200/0
gaddr 10.10.10.140/68 laddr 10.10.10.140/68
ICMP echo request from dmz:10.10.10.140 to out:172.16.16.200 ID=68 seq=0 len=72
%ASA-6-106100: access-list OUT permitted icmp out/172.16.16.200(0) ->
dmz/10.10.10.140(0) hit-cnt 1 first hit [0x4f98dc69, 0xf6ef56b4]
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.200/0 gaddr
10.10.10.140/68 laddr 10.10.10.140/68
ICMP echo reply from out:172.16.16.200 to dmz:10.10.10.140 ID=68 seq=0 len=72
```

Example 8-14 *Avoiding NAT Exemption for a Specific Static Translation*

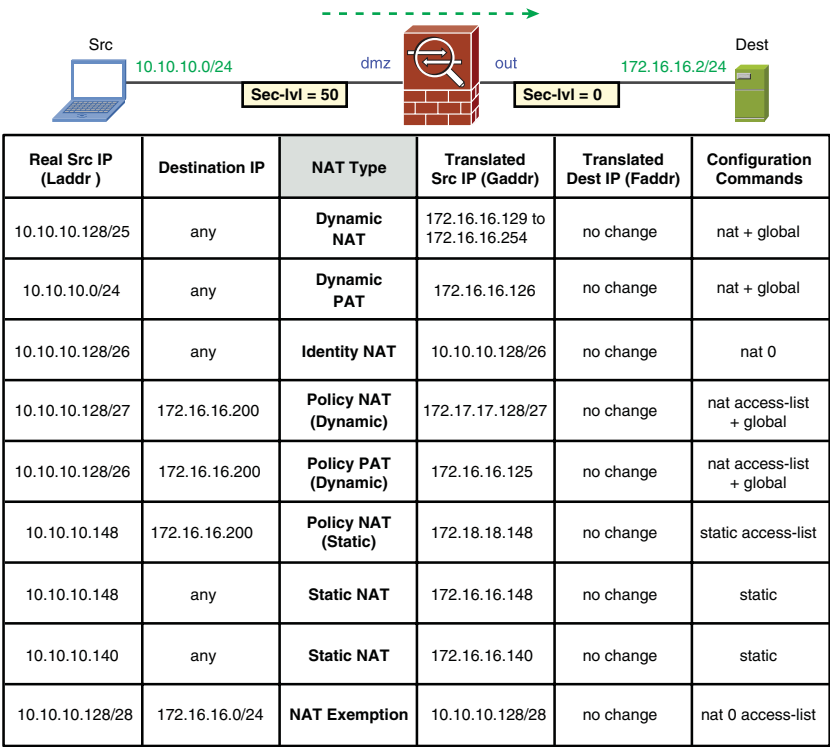
```
! Host 10.10.10.140 is excluded from the definition of NAT exemption traffic
access-list NONAT2 extended deny ip host 10.10.10.140 any
access-list NONAT2 extended permit ip 10.10.10.128 255.255.255.240 172.16.16.0
255.255.255.0
nat (dmz) 0 access-list NONAT2
!
! Host 10.10.10.140 is not subject to NAT Exemption
static (dmz,out) 172.16.16.140 10.10.10.140 netmask 255.255.255.255
```

NAT Precedence Rules

Having promoted a detailed examination of the various NAT types, it is time to describe what is the *Order of NAT Processing* followed by ASA. You need to be acquainted with this topic, for in practical scenarios where several categories of translations are simultaneously in place, there is a huge potential for rules overlapping.

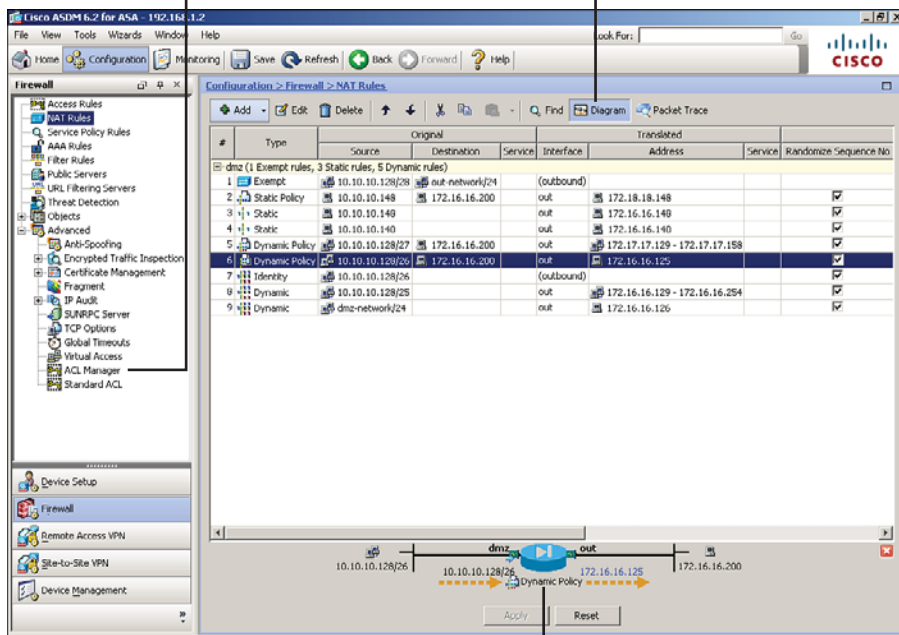
Drawing on the method employed in Examples 8-6 and 8-8, Example 8-15 was conceived to create conflicting rules among the NAT Types studied so far.

Figure 8-3 summarizes the translation statements defined in Examples 8-2 through 8-14 (which are all enabled at the same time in Example 8-15). The analysis of such a scenario is documented in Example 8-16. Figure 8-4 shows the ASDM configuration screen corresponding to the NAT rules of Example 8-15.



This link displays all the ACLs configured on ASA, including those used for NAT Exemption and Policy NAT. (Configuration > Firewall > Advanced > ACL Manager)

Diagram



This appears when the *Diagram* option is selected

Figure 8-4 ASDM View of the NAT Rules Configured in Example 8-15

Example 8-15 Scenario for the Analysis of NAT Precedence for Outbound

```
! ACLs referred to by Policy NAT and NAT Exemption Rules
access-list STATIC-POLICY1 extended permit ip host 10.10.10.148 host 172.16.16.200
access-list DYN-POLICY1 extended permit ip 10.10.10.128 255.255.255.224 host
172.16.16.200
access-list DYN-POLICY2 extended permit ip 10.10.10.128 255.255.255.192 host
172.16.16.200
access-list NONAT extended permit ip 10.10.10.128 255.255.255.240
172.16.16.0 255.255.255.0
!

! Final configuration for nat, global and static commands
ASA2# show running-config nat
nat (dmz) 0 access-list NONAT
nat (dmz) 4 access-list DYN-POLICY1
nat (dmz) 3 access-list DYN-POLICY2
nat (dmz) 0 10.10.10.128 255.255.255.192
```

```

nat (dmz) 2 10.10.10.128 255.255.255.128
nat (dmz) 1 10.10.10.0 255.255.255.0
!
ASA2# show running-config global
global (out) 4 172.17.17.129-172.17.17.158 netmask 255.255.255.224
global (out) 2 172.16.16.129-172.16.16.254 netmask 255.255.255.128
global (out) 3 172.16.16.125
global (out) 1 172.16.16.126
!
ASA2# show running-config static
static (dmz,out) 172.18.18.148 access-list STATIC-POLICY1
static (dmz,out) 172.16.16.148 10.10.10.148 netmask 255.255.255.255
static (dmz,out) 172.16.16.140 10.10.10.140 netmask 255.255.255.255

```

Example 8-16 employs the **show nat dmz** command to register the NAT precedence rules for outbound access when **nat-control** is configured. The order of processing follows:

Example 8-16 *Characterizing NAT Precedence Rules for Outbound Access*

```

! Determining Precedence for the various categories of address translation
ASA2# show nat dmz out
match ip dmz 10.10.10.128 255.255.255.240 out 172.16.16.0 255.255.255.0
    NAT exempt
    translate_hits = 0, untranslate_hits = 0
match ip dmz host 10.10.10.148 out host 172.16.16.200
    static translation to 172.18.18.148
    translate_hits = 0, untranslate_hits = 0
match ip dmz host 10.10.10.148 out any
    static translation to 172.16.16.148
    translate_hits = 0, untranslate_hits = 0
match ip dmz host 10.10.10.140 out any
    static translation to 172.16.16.140
    translate_hits = 0, untranslate_hits = 0
match ip dmz 10.10.10.128 255.255.255.224 out host 172.16.16.200
    dynamic translation to pool 4 (172.17.17.129 - 172.17.17.158)
    translate_hits = 0, untranslate_hits = 0
match ip dmz 10.10.10.128 255.255.255.192 out host 172.16.16.200
    dynamic translation to pool 3 (172.16.16.125)
    translate_hits = 0, untranslate_hits = 0
match ip dmz 10.10.10.128 255.255.255.192 out any
    identity NAT translation, pool 0
    translate_hits = 0, untranslate_hits = 0
match ip dmz 10.10.10.128 255.255.255.128 out any
    dynamic translation to pool 2 (172.16.16.129 - 172.16.16.254)
    translate_hits = 0, untranslate_hits = 0
match ip dmz 10.10.10.0 255.255.255.0 out any

```

```
dynamic translation to pool 1 (172.16.16.126)
  translate_hits = 0, untranslate_hits = 0
match ip dmz any out any
  no translation group, implicit deny
  policy_hits = 0
```

- Step 1.** **NAT Exemption:** This is always the first to be checked and has precedence over any other type of NAT rule that eventually conflicts with it.
- Step 2.** **Static Policy NAT:** The motivation for this type of rule is to allow the selection of distinct global addresses for a given *laddr*, depending on the destination address (*faddr*) being contacted. This can be thought of as an exception to a generic **static** statement and, as such, needs to be configured before regular **statics**.
- Step 3.** **Static NAT:** This is checked before all the Dynamic, Dynamic Policy, and Identity NAT rules. If some hosts that fall within a NAT Exemption range require static translation, the pertinent exceptions in the **nat 0 access-list** command need to be created, as previously illustrated in Example 8-14.
- Step 4.** **Dynamic Policy NAT/PAT:** These rules are checked before the Dynamic and Identity NAT rules. If two rules of this category exist, the precedence is given by the order in which they were configured (does not matter if is a Policy PAT or Policy NAT).
- Step 5.** **Identity NAT:** This unidirectional translation bypass method is evaluated before any Dynamic NAT or Dynamic PAT rule.
- Step 6.** **Dynamic NAT:** This category has precedence over Dynamic PAT only.
- Step 7.** **Dynamic PAT:** If after running through all the previous NAT types, ASA does not find a match, it still searches for a Dynamic PAT. If no matching rule is located, the implicit deny rule that characterizes the NAT-control model is enforced.

To further enforce the concepts just described, a set of tests, in which conflicting rules do appear, is proposed (always referring to Figure 8-3). In this particular scenario, the highest potential for overlapping relates to situations in which the *laddr* tries to connect to the *faddr* 172.16.16.200, the host used as the destination for all the Policy NAT statements. (The suggested source and destination hosts for each of the tests are listed here:

- T1 From 10.10.10.150 to 172.16.16.200:** The most specific range that contains this *laddr* is 10.10.10.128/27, for which Dynamic Policy NAT is configured. Because there is neither a static nor a NAT Exemption rule defined for this source host, an address from the global pool #4 is selected as the *gaddr*.
- T2 From 10.10.10.180 to 172.16.16.200:** The *laddr* belongs to 10.10.10.128/26 and is translated to 172.16.16.125 (Dynamic Policy PAT) when connecting to this *faddr*.

- T3 From 10.10.10.180 to 172.16.16.220:** The *laddr* belongs to 10.10.10.128/26 but does not meet any Policy NAT rule, thus allowing Identity translation to take place.
- T4 From 10.10.10.148 to 172.16.16.200:** The *laddr* matches the policy defined by ACL STATIC-POLICY1 and does not compete with NAT Exemption. The *gaddr* 172.18.18.148 is then selected.
- T5 From 10.10.10.148 to 172.16.16.220:** Similar to T4, but using the regular **static** entry for this *laddr*, yielding the *gaddr* 172.16.16.148.
- T6 From 10.10.10.240 to 172.16.16.200:** The most specific range for this *laddr* is 10.10.10.128/25 and a *gaddr* from pool 2 (Dynamic NAT) is chosen.
- T7 From 10.10.10.25 to 172.16.16.200:** Because there is no conflict of the *laddr* with any of the rules for subnet 10.10.10.128/25, Dynamic PAT (pool 1) is the choice in this case.
- T8 From 10.10.10.140 to 172.16.16.200:** This combination of *laddr* and *faddr* matches the NAT Exemption settings and the connection is accepted without any translation.

Tip Packet Tracer is useful for performing tests T1 through T8. For instance, T6 could be simply `packet-tracer input dmz tcp 10.10.10.240 6000 172.16.16.200 443`. You are strongly encouraged to reproduce these testing activities.

Address Publishing for Inbound Access

The term *Address Publishing* is employed in the context of the **nat-control** model, whenever an internal address needs to be univocally identifiable on the outside. To meet such a requirement, a bidirectional method of translation is necessary. An explicit **permit** statement (within an ACL) should then be configured so that inbound access to the *gaddr* (published address) can take place.

Publishing with the static Command

Static NAT is the classic method for address publishing. As studied in Example 8-9, when the **static** command is entered in the CLI, a permanent translation is immediately added to the **xlate** table.

Example 8-17 assembles some commands that allow Telnet from *out* hosts that reside on the 172.16.16.0/24 subnet to the *dmz* address 10.10.10.140.

Example 8-17 Using the static Command for Address Publishing

! Defining object-groups and an ACL that uses them
 object-group network OUTSIDE

```

network-object 172.16.16.0 255.255.255.0
!
object-group network TRANSLATED1
  network-object 172.16.16.0 255.255.255.0
  network-object 10.10.10.0 255.255.255.0
!
access-list OUT extended permit tcp object-group OUTSIDE object-group TRANSLATED1 eq 23
access-group OUT in interface out
!
! Making the host 10.10.10.140 visible as 172.16.16.140 in the 'out' interface
static (dmz,out) 172.16.16.140 10.10.10.140 netmask 255.255.255.255

```

Note A common configuration is to use a `static (dmz,out) X X` command to publish the *dmz* address *X* unchanged on the *out* interface. This might be called *Identity Static*.

Note Figure 8-6, at the end of this chapter, shows a practical scenario that illustrates not only address publishing but also the configuration of the pertinent ACL to enable inbound connections.

Publishing with Port Redirection

Port Redirection, a special case of static translation, is sometimes referred to as Static PAT and enables the specification of the L4 port in the mapping between the *laddr* and the *gaddr*. Some situations in which this method may be useful follow:

- When mapping multiple inside servers to a single global address.
- When the service port of a given inside host needs to be published as a different port.

Example 8-18 illustrates a situation in which the telnet port on the *dmz* host 10.10.10.150 is statically mapped to port TCP/10023 on the *out* host 172.16.16.150. The `show xlate debug` command reveals a combination of the *s* and *r* flags for this NAT Type.

Example 8-18 Using Port Redirection for Address Publishing

```

! Creating a port redirection rule
ASA2(config)# static (dmz,out) tcp 172.16.16.150 10023 10.10.10.150 telnet
%ASA-6-305011: Built static TCP translation from dmz:10.10.10.150/23 to
out:172.16.16.150/10023
!
! ACL for allowing inbound connections to TCP port 10023
access-list OUT extended permit tcp object-group OUTSIDE host 172.16.16.150 eq
10023 log notifications interval 120
!

```



```
ASA2# show xlate debug
1 in use, 5 most used
Flags: D - DNS, d - dump, I - identity, i - dynamic, n - no random,
      r - portmap, s - static
TCP PAT from dmz:10.10.10.150/23 to out:172.16.16.150/10023 flags sr idle 0:03:44
timeout 0:00:00
```

Note Given that this chapter deals with pre-8.3 releases, an ACE aimed to permit access in a port redirection scenario must refer to the *gaddr/mapped_port* pair.

Publishing with NAT Exemption

NAT Exemption is a bidirectional form of translation that may be used to produce the address publishing effect for connections between the source and destination addresses defined with the **nat 0 access-list** command.

Example 8-19 shows that NAT Exemption accomplishes the publishing of the *dmz* host 10.10.10.80 on the *out* interface. The **access-list** named *OUT* (defined in Example 8-17) accepts the connection to the *gaddr* 10.10.10.80, as shown in the **show conn** command. The **show nat** command reveals that there was an *untranslate* hit in the NAT exemption rule.

Example 8-19 Using NAT Exemption for Address Publishing

```
! Connections between 10.10.10.0/25 and 172.16.16.0/24 are exempted from NAT
access-list NONAT extended permit ip 10.10.10.0 255.255.255.128 172.16.16.0
255.255.255.0
nat (dmz) 0 access-list NONAT
!
! Host 172.16.16.200 telnets to 10.10.10.80 (located on the 'dmz')
%ASA-6-106100: access-list OUT permitted tcp out/172.16.16.200(61954) ->
dmz/10.10.10.80(23) hit-cnt 1 first hit [0x2b839c8d, 0x36521d7]
%ASA-6-302013: Built inbound TCP connection 372 for out:172.16.16.200/61954
(172.16.16.200/61954) to dmz:10.10.10.80/23 (10.10.10.80/23)
!
ASA2# show conn
1 in use, 8 most used
TCP out 172.16.16.200:61954 dmz 10.10.10.80:23, idle 0:01:10, bytes 163, flags UIOB
!
ASA2# show nat dmz out
match ip dmz 10.10.10.0 255.255.255.128 out 172.16.16.0 255.255.255.0
NAT exempt
translate_hits = 0, untranslate_hits = 1
match ip dmz any out any
no translation group, implicit deny
policy_hits = 0
```

Figure 8-5 brings a simplified version of the combined processing of NAT and ACL rules, for both inbound and outbound classes of access. A scenario in which a packet, that does not match any pre-existent flow, crosses the firewall from an interface with security level X to another with sec-lvl Y, is depicted.

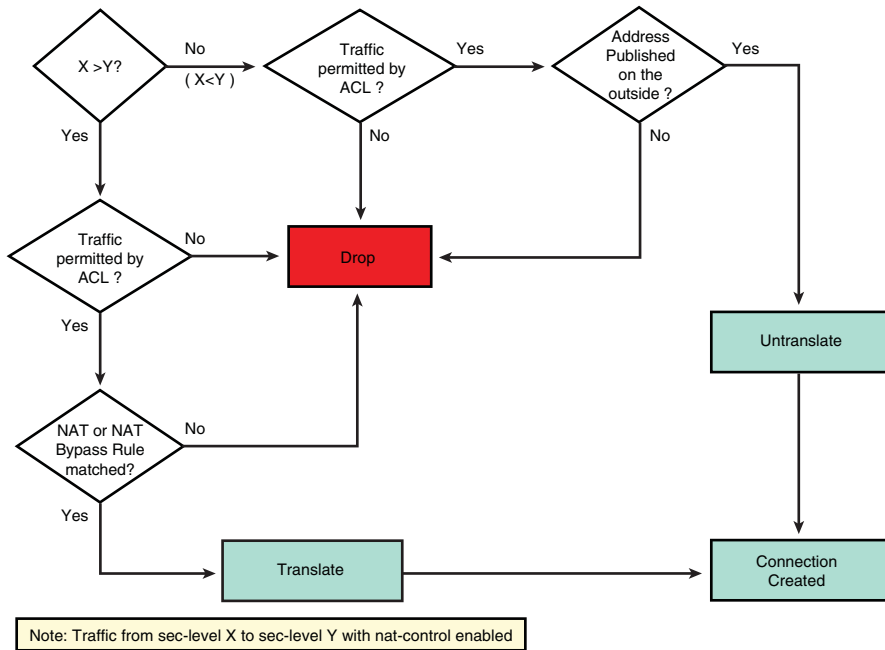


Figure 8-5 *Simplified Analysis of NAT and ACL Processing*

Inbound NAT Analysis

The previous sections provided not only a detailed treatment of Outbound NAT but also the requirements for address publishing on a lower sec-lvl interface. Outbound NAT was the only mode available in the early days of the ASA algorithm (implemented by PIX Firewalls at that time). To reverse the original behavior and permit Inbound NAT, Cisco introduced the **outside** keyword for the dynamic translation rules. All the categories of NAT exemplified in this section are completely analogous to the respective inbound type.

Dynamic PAT for Inbound

Example 8-20 assembles the commands to configure an Inbound Dynamic PAT rule. Hosts in the range 172.16.16.0/24 are translated to address 10.10.10.126 when connecting from the *out* to the *dmz* interface. This example also emphasizes the need for the keyword **outside** at the end of the **nat** statement.

Example 8-20 *Configuring and Verifying Dynamic PAT for Inbound*

```

! The keyword 'outside' is indispensable for inbound NAT/PAT
ASA2(config)# nat (out) 1 172.16.16.0 255.255.255.0
WARNING: Binding inside nat statement to outermost interface.
WARNING: Keyword "outside" is probably missing.
!
! Configuring inbound PAT
ASA2(config)# nat (out) 1 172.16.16.0 255.255.255.0 outside
ASA2(config)# global (dmz) 1 10.10.10.126
INFO: Global 10.10.10.126 will be Port Address Translated
!
! Sample Packet Tracer simulation for Inbound NAT
ASA2# packet-tracer input out icmp 172.16.16.10 8 0 1 10.10.10.140
Phase: 1
Type: FLOW-LOOKUP
Subtype:
Result: ALLOW
Config:
Additional Information:
Found no matching flow, creating a new flow
%ASA-6-106100: access-list OUT1 permitted icmp out/172.16.16.10(8) ->
dmz/10.10.10.140(0) hit-cnt 1 first hit [0x110e0bc6, 0x32ebf4dc]
%ASA-6-305011: Built dynamic ICMP translation from out:172.16.16.10/1 to
dmz:10.10.10.126/55499
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.10/1 gaddr
10.10.10.140/0 laddr 10.10.10.140/0
[ output suppressed ]
Phase: 3
Type: ACCESS-LIST
Subtype: log
Result: ALLOW
Config:
access-group OUT1 in interface out
access-list OUT1 extended permit icmp object-group OUTSIDE object-group DMZ1
object-group PING log notifications interval 120
[ output suppressed ]
Phase: 7
Type: NAT
Subtype:
Result: ALLOW
Config:
nat (out) 1 172.16.16.0 255.255.255.0 outside
match ip out 172.16.16.0 255.255.255.0 dmz any
dynamic translation to pool 1 (10.10.10.126)
translate_hits = 1, untranslate_hits = 0

```

Additional Information:

Dynamic translate 172.16.16.10/1 **to 10.10.10.126/55499** using netmask 255.255.255.255

[output suppressed]

Action: allow

Identity NAT for Inbound

Example 8-21 illustrates Identity NAT in action for source hosts that belong to the 172.16.16.128/26 range. This category of NAT is always unidirectional, independently of being applied for inbound or outbound access.

Example 8-21 *Configuring and Verifying Identity NAT for Inbound*

! The keyword 'outside' is indispensable for inbound Identity NAT

```
ASA2(config)# nat (out) 0 172.16.16.128 255.255.255.192
```

nat 0 172.16.16.128 will be identity translated for outbound

WARNING: Binding inside nat statement to outermost interface.

WARNING: Keyword "outside" is probably missing.

!

```
ASA2(config)# nat (out) 0 172.16.16.128 255.255.255.192 outside
```

nat 0 172.16.16.128 will be identity translated for outbound

!

```
ASA2# packet-tracer input out icmp 172.16.16.180 8 0 1 10.10.10.140
```

[output suppressed]

Found no matching flow, **creating a new flow**

```
%ASA-6-106100: access-list OUT1 permitted icmp out/172.16.16.180(8) ->
dmz/10.10.10.140(0) hit-cnt 1 first hit [0x110e0bc6, 0x32ebf4dc]
```

```
%ASA-6-305009: Built dynamic translation from out:172.16.16.180 to
dmz:172.16.16.180
```

```
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.180/1 gaddr
10.10.10.140/0 laddr 10.10.10.140/0
```

[output suppressed]

Phase: 3

Type: ACCESS-LIST

Subtype: log

Result: ALLOW

Config:

```
access-group OUT1 in interface out
```

```
access-list OUT1 extended permit icmp object-group OUTSIDE object-group DMZ1
object-group PING log notifications interval 120
```

[output suppressed]

Phase: 7

Type: NAT

Subtype:

Result: ALLOW

Config:

```
nat (out) 0 172.16.16.128 255.255.255.192 outside
  match ip out 172.16.16.128 255.255.255.192 dmz any
  identity NAT translation, pool 0
  translate_hits = 1, untranslate_hits = 0
```

Additional Information:

```
Dynamic translate 172.16.16.180/0 to 172.16.16.180/0 using netmask 255.255.255.255
[ output suppressed ]
Action: allow
```

NAT Exemption for Inbound

Example 8-22 shows how to implement NAT Exemption for inbound and highlights that the precedence rules are completely analogous to those established for outbound.

Example 8-22 *Configuring and Verifying NAT Exemption for Inbound*

```
! Connections between 172.16.16.128/28 and 10.10.10.0/24 are exempted from NAT
access-list NONAT1 extended permit ip 172.16.16.128 255.255.255.240
10.10.10.0 255.255.255.0
nat (out) 0 access-list NONAT1 outside
!
! Analogous Precedence Rules are valid for Inbound
ASA2# show nat out dmz
  match ip out 172.16.16.128 255.255.255.240 dmz 10.10.10.0 255.255.255.0
    NAT exempt
    translate_hits = 0, untranslate_hits = 0
  match ip out 172.16.16.128 255.255.255.192 dmz any
    identity NAT translation, pool 0
    translate_hits = 0, untranslate_hits = 0
  match ip out 172.16.16.0 255.255.255.0 dmz any
    dynamic translation to pool 1 (10.10.10.126)
    translate_hits = 0, untranslate_hits = 0
```

Static NAT for Inbound

Static NAT for inbound access does not require the **outside** keyword. Example 8-23 illustrates the configuration of a static rule from the *out* to the *dmz* interface.

Example 8-23 *Configuring Static NAT for Inbound*

```
! Static Translation for 'out' host 172.16.16.150
ASA2(config)# static (out,dmz) 10.10.10.150 172.16.16.150
%ASA-6-305009: Built static translation from out:172.16.16.150 to dmz:10.10.10.150
```

Dual NAT

Dual NAT is supported by ASA in pre-8.3 releases through the combination of inbound and outbound **static** rules.

Example 8-24 assembles two **static** commands used to produce the Dual NAT effect:

- *dmz* host 10.10.10.140 is published as 172.16.16.140 on the *out* interface.
- *out* host 172.16.16.200 is seen as 10.10.10.200 on the *dmz*.

This example goes a bit further by illustrating the connection and translation setup processes for a telnet session from 172.16.16.200 to 172.16.16.140:

- The visibility of the out host 172.16.16.200 is limited to the published address 172.16.16.140.
- The ACE that permits inbound telnet specifies 172.16.16.140 as the destination IP.
- The **show conn** command displays the real addresses.
- The *dmz* host sees the connection as if it had been initiated from 10.10.10.200.

Example 8-24 Configuring Dual NAT

```
! 'dmz' real address 10.10.10.140 seen as 172.16.16.140 on 'out' interface
static (dmz,out) 172.16.16.140 10.10.10.140 netmask 255.255.255.255
!
! 'out' real address 172.16.16.200 becomes visible as 10.10.10.200 on 'dmz'
interface
static (out,dmz) 10.10.10.200 172.16.16.200 netmask 255.255.255.255
!
ASA2# show xlate debug
2 in use, 2 most used
Flags: D - DNS, d - dump, I - identity, i - dynamic, n - no random,
       r - portmap, s - static
NAT from out:172.16.16.200 to dmz:10.10.10.200 flags s idle 0:09:41 timeout 0:00:00
NAT from dmz:10.10.10.140 to out:172.16.16.140 flags s idle 0:10:11 timeout 0:00:00
!
! Host 172.16.16.200 telnets to published address 172.16.16.140
OUT#telnet 172.16.16.140
Trying 172.16.16.140 ... Open
Reserved port 27138 in Transport Port Agent for TCP IP type 1
TCP: sending SYN, seq 3999413401, ack 0
TCP2: Connection to 172.16.16.140:23, advertising MSS 1460
```

```

TCP2: state was CLOSED -> SYNSENT [27138 -> 172.16.16.140(23)]
TCP2: state was SYNSENT -> ESTAB [27138 -> 172.16.16.140(23)]
TCP: tcb 84AED73C connection to 172.16.16.140:23, peer MSS 1380, MSS is 1380
TCB84AED73C connected to 172.16.16.140.23
DMZ>
! ACL match and connection building on ASA
%ASA-6-106100: access-list OUT permitted tcp out/172.16.16.200(27138) ->
dmz/172.16.16.140(23) hit-cnt 1 first hit [0xccc24816, 0x987b42bc]
%ASA-6-302013: Built inbound TCP connection 38 for out:172.16.16.200/27138
(10.10.10.200/27138) to dmz:10.10.10.140/23 (172.16.16.140/23)
!
! ASA displays real addresses in the connections table
ASA2# show conn detail | begin out:
TCP out:172.16.16.200/27138 dmz:10.10.10.140/23,
      flags UIOB, idle 4s, uptime 8s, timeout 1h0m, bytes 146
!
! Host 10.10.10.140 sees the connection as if it came from 10.10.10.200
Reserved port 0 in Transport Port Agent for TCP IP type 0
TCP0: state was LISTEN -> SYNRCVD [23 -> 10.10.10.200(27138)]
TCP: tcb 8414768C connection to 10.10.10.200:27138, peer MSS 1380, MSS is 516
TCP: sending SYN, seq 26169904, ack 4093011897
TCP0: Connection to 10.10.10.200:27138, advertising MSS 1380
TCP0: state was SYNRCVD -> ESTAB [23 -> 10.10.10.200(27138)]
DMZ#show tcp brief
TCB          Local Address          Foreign Address          (state)
8414768C    10.10.10.140.23          10.10.10.200.27138      ESTAB

```

Example 8-25 registers a sample Packet Tracer simulation for an outbound connection in a Dual NAT environment. It is instructive to observe that there is an *untranslate* hit for the *out* real address (172.16.16.200) and a *translate* hit for the *dmz* real IP (10.10.10.140).

Example 8-25 Analyzing Dual NAT with Packet Tracer

```

ASA2# packet-tracer input dmz tcp 10.10.10.140 4500 10.10.10.200 80
%ASA-6-302013: Built outbound TCP connection 35 for out:172.16.16.200/80
(10.10.10.200/80) to dmz:10.10.10.140/4500 (172.16.16.140/4500)
[ output suppressed ]
Phase: 3
Type: UN-NAT
Subtype: static
Result: ALLOW
Config:
static (out,dmz) 10.10.10.200 172.16.16.200 netmask 255.255.255.255
nat-control

```

```

match ip out host 172.16.16.200 dmz any
static translation to 10.10.10.200
translate_hits = 0, untranslate_hits = 1
Additional Information:
NAT divert to egress interface out
Untranslate 10.10.10.200/0 to 172.16.16.200/0 using netmask 255.255.255.255
[ output suppressed ]
Phase: 5
Type: NAT
Subtype:
Result: ALLOW
Config:
static (dmz,out) 172.16.16.140 10.10.10.140 netmask 255.255.255.255
nat-control
match ip dmz host 10.10.10.140 out any
static translation to 172.16.16.140
translate_hits = 1, untranslate_hits = 0
Additional Information:
Static translate 10.10.10.140/0 to 172.16.16.140/0 using netmask 255.255.255.255
[ output suppressed ]
Action: allow

```

Disabling TCP Sequence Number Randomization

This topic was initially investigated in Chapter 7. The justification for revisiting it is the possibility of using the **norandomseq** option while performing address translation. This can be useful to restrict the situations in which randomization is disabled.

Example 8-26 shows sample configurations of NAT Types that support the **norandomseq** parameter. The **show xlate debug** command dedicates the flag *n* to signal that no randomization is taking place for a given translation entry.

Example 8-26 *Disabling Sequence Number Randomization During Translations*

```

! Summary configuration of NAT Types
access-list DYN-POLICY2 extended permit ip 10.10.10.128 255.255.255.192 host
172.16.16.200
!
ASA2# show running-config | include norandomseq
nat (dmz) 3 access-list DYN-POLICY2 norandomseq
nat (dmz) 0 10.10.10.150 255.255.255.255 norandomseq
nat (dmz) 1 10.10.10.0 255.255.255.0 norandomseq

```



```

static (dmz,out) 172.16.16.180 10.10.10.180 netmask 255.255.255.255 norandomseq
!
ASA2# show running-config global
global (out) 1 172.18.18.18
global (out) 3 172.16.16.125
!
! NAT with 'norandomseq' in action
ASA2# show xlate debug
4 in use, 5 most used
Flags: D - DNS, d - dump, I - identity, i - dynamic, n - no random,
      r - portmap, s - static
NAT from dmz:10.10.10.180 to out:172.16.16.180 flags sn idle 0:11:13 timeout
0:00:00
NAT from dmz:10.10.10.150 to out:10.10.10.150 flags niI idle 0:00:06 timeout
3:00:00
ICMP PAT from dmz:10.10.10.80/11 to out:172.18.18.18/53205 flags rni idle 0:00:18
timeout 0:00:30
ICMP PAT from dmz:10.10.10.140/12 to out(DYN-POLICY2):172.16.16.125/4571 flags rni
idle 0:00:13 timeout 0:00:30

```

Note The *NAT Exemption* resource does not support disabling the TCP sequence number randomization feature:

```

ASA2(config)# nat (dmz) 0 access-list NONAT norandomseq
WARNING: norandomseq is not allowed with NAT 0.

```

Tip The possibility of using the **norandomseq** option is one of the reasons for preferring the *Identity Static* address publishing method over NAT Exemption. (Refer to Examples 8-17 and 8-19).

Defining Connection Limits with NAT Rules

The ASA algorithm supports the definition of acceptable connection limits within address translation statements. One key usage of this feature is the contention of Denial of Service (DoS) attacks to specific hosts (typically servers). DoS mitigation for client side addresses is analyzed in Chapter 11, “Additional Protection Mechanisms.”

Example 8-27 demonstrates how a limit for the number of simultaneous UDP connections can be defined for a host whose IP is being published with the **static** command. The example emphasizes that, after the configured number of 25 connections is reached, ASA stops accepting new requests.

After that, as some connections are torn down, new ones are allowed through (but never exceeding the configured upper bound of 25).

For TCP, there are two configurable parameters: The first option specifies the number of simultaneous connections, and the second defines the tolerable number of embryonic connections. (Those that have not completed the TCP three-way handshake and that many times relate to attack packets.) In Example 8-28, an upper bound of 20 concurrent half-open (embryonic) connections is set up for the published address 172.16.222.40.

The topology associated with Examples 8-27 and 8-28 is portrayed in Figure 8-6, which also highlights the relationship of *Access Control Entries* with published addresses. The permit statements should include the published addresses (rather than the real ones).

Example 8-27 *Defining the Limit of UDP Connections for a Host*

```
! Defining a limit of 25 simultaneous UDP connections for host 172.16.222.60
static (dmz,out) 172.16.222.60 172.16.222.60 netmask 255.255.255.255 udp 25
!
! Hosts on subnet 172.16.220.0/24 create connections to 172.16.220.60 on UDP/53
%ASA-6-302015: Built inbound UDP connection 46926 for out:172.16.220.90/7000
(172.16.220.90/7000) to dmz:172.16.222.60/53 (172.16.222.60/53)
%ASA-6-302015: Built inbound UDP connection 46927 for out:172.16.220.91/7001
(172.16.220.91/7001) to dmz:172.16.222.60/53 (172.16.222.60/53)
[ output suppressed ]
%ASA-6-302015: Built inbound UDP connection 46950 for out:172.16.220.94/7024
(172.16.220.94/7024) to dmz:172.16.222.60/53 (172.16.222.60/53)
%ASA-3-201011: Connection limit exceeded 25/25 for inbound packet from
172.16.220.95/7025 to 172.16.222.60/53 on interface out
[ output suppressed ]
%ASA-3-201011: Connection limit exceeded 25/25 for inbound packet from
172.16.220.92/7122 to 172.16.222.60/53 on interface out
%ASA-6-302016: Teardown UDP connection 46926 for out:172.16.220.90/7000 to
dmz:172.16.222.60/53 duration 0:02:02 bytes 70
%ASA-6-302016: Teardown UDP connection 46927 for out:172.16.220.91/7001 to
dmz:172.16.222.60/53 duration 0:02:01 bytes 70
%ASA-6-302015: Built inbound UDP connection 46952 for out:172.16.220.93/7123
(172.16.220.93/7123) to dmz:172.16.222.60/53 (172.16.222.60/53)
```

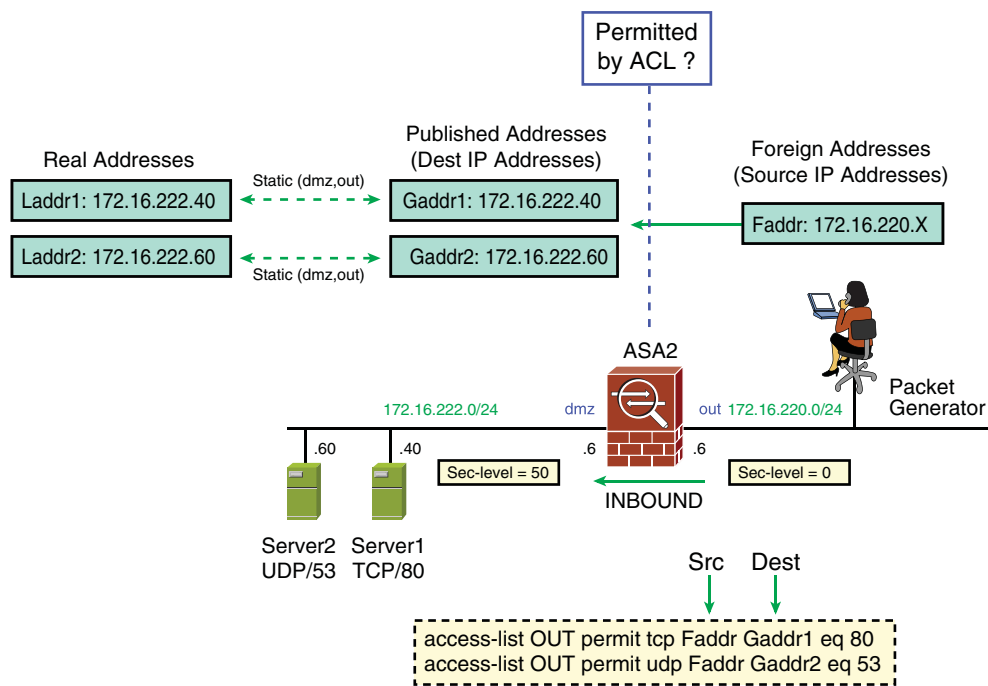


Figure 8-6 Reference Topology for the Analysis of Connection Limits

Example 8-28 Defining the Limit of Embryonic TCP Connections for a Host

```
! Defining a limit of 20 embryonic TCP connections for host 172.16.222.40
static (dmz,out) 172.16.222.40 172.16.222.40 netmask 255.255.255.255 tcp 0 20
!
! Hosts on subnet 172.16.220.0/24 create connections to 172.16.222.40 on port
TCP/80
%ASA-6-302013: Built inbound TCP connection 46977 for out:172.16.220.100/20000
(172.16.220.100/20000) to dmz:172.16.222.40/80 (172.16.222.40/80)
%ASA-6-302013: Built inbound TCP connection 46978 for out:172.16.220.101/20001
(172.16.220.101/20001) to dmz:172.16.222.40/80 (172.16.222.40/80)
[ output suppressed ]
%ASA-6-302013: Built inbound TCP connection 46996 for out:172.16.220.119/20019
(172.16.220.119/20019) to dmz:172.16.222.40/80 (172.16.222.40/80)
%ASA-6-201010: Embryonic connection limit exceeded 20/20 for inbound packet from
172.16.220.120/20020 to 172.16.222.40/80 on interface out
[ output suppressed ]
%ASA-6-201010: Embryonic connection limit exceeded 20/20 for inbound packet from
172.16.220.130/20030 to 172.16.222.40/80 on interface out
%ASA-6-302014: Teardown TCP connection 46977 for out:172.16.220.100/20000 to
dmz:172.16.222.40/80 duration 0:00:30 bytes 100 SYN Timeout
%ASA-6-302013: Built inbound TCP connection 46997 for out:172.16.220.131/20031
(172.16.220.131/20031) to dmz:172.16.222.40/80 (172.16.222.40/80)
```

Tip A value of *zero* for any of the connection limits means that an unlimited number of connections is enabled. The configurable values belong to the range 1 to 65535.

Summary

This chapter presented in a great level of detail the **nat-control** model and the various NAT types available on ASA appliances. Particular emphasis was placed on establishing the *NAT Precedence Rules*, a frequently misunderstood topic, thus leading to poor design and ineffective implementations.

Having studied this chapter, you should be able to do the following:

- Understand and configure each NAT type defined in the ASA algorithm.
- Define how connections depend on translations. (Or *conditional NAT bypass* options when **nat-control** is active.)
- Extract valuable information from the **show xlate debug** and **show nat** commands.
- Describe the NAT Precedence Rules and foresee the result of overlapping translation statements.
- Describe and configure Dual NAT.
- Configure connection limits for a given host using translation statements.
- Understand the usage of the **norandomseq** parameter.

This page intentionally left blank

Chapter 9

Classic IOS Firewall Overview

This chapter covers the following topics:

- **Motivations for CBAC:** Presents the motivations for the *Classic IOS Firewall* Stateful Inspection solution, originally known as *Context Based Access Control* (CBAC).
- **CBAC basics:** Introduces the generic Stateful Inspection of TCP, UDP and ICMP using CBAC. It also promotes a quick comparison between ASA and IOS Firewall philosophies.
- **ICMP connection examples:** Registers how the Classic IOS Firewall can treat ICMP as a *pseudo* Stateful protocol.
- **UDP connection examples:** Explores some sample UDP connections created through the *Classic IOS Firewall*.
- **TCP connection examples:** Deals with TCP, the truly stateful transport protocol. It also illustrates some advanced resources that relate to the knowledge that CBAC has of the *TCP State Machine* and goes a bit further by showing how to limit the number of connections for an individual host, which may be useful for containing DoS attacks.
- **Handling ACLs and Object-Groups:** Presents a quick review of IOS Access Control Lists. It also covers the handling of IOS object-groups and how they apply to the creation of more structured ACLs. Finally, it documents the interaction of ACLs with CBAC.
- **IOS NAT review:** Revisits the main NAT resources available within Cisco IOS. It assumes a basic knowledge of *Network Address Translation* theory, which was covered in the previous chapter. The effects of enabling NAT on Flow Accounting tasks are also examined.
- **CBAC and NAT:** Analyzes the simultaneous operation of CBAC, NAT and ACLs, a combination that may frequently appear in practical usage scenarios with more advanced Security requirements.

“Those who cannot remember the past are condemned to repeat it.”—George Santayana

The previous two chapters detailed the main concepts of the Adaptive Security algorithm. Its theory of operations for environments that employ either the **nat-control** or **no nat-control** models was thoroughly examined. Topics such as connection setup, maintenance, and teardown were largely exemplified, enabling you to become acquainted with the baseline firewall functionality within the ASA family.

Instead of diving deeper into the ASA domains, this chapter starts building your knowledge of IOS firewall fundamentals (connections, translations, ACL handling, and so on).

This chapter covers the *Classic IOS Firewall*, historically referred to as *Context-Based Access Control (CBAC)*, whereas the next one focuses on the newer *IOS Zone-Based Policy Firewall (ZFW)* approach. Despite the tendency of concentrating the development of new features on ZFW, there are some reasons for dedicating a chapter to the analysis of CBAC operation:

- The installed base of routers running older IOS versions that do not support ZFW is certainly not neglectable. Although CBAC's configuration is a bit more complex than ZFW's, it can still prove helpful on several practical deployments.
- When studying science, it is generally instructive to look at the development of new ideas throughout time. Likewise, an insight into how the implementation philosophy of IOS has evolved may reinforce relevant theoretical concepts.

If you do not agree with the second justification or are lucky enough to own (or manage) brand new routers, you can skip this chapter.

Motivations for CBAC

Network environments such as the corporate Data Centers or the Internet Edge typically present a strong demand for performance and availability, when security design is under consideration. This usually leads to the selection of dedicated devices for each role (VPN, Firewall, and IPS, for instance).

Other *places in the network*, such as the remote branches, have an appeal for flexibility and integration capabilities (rather than throughput), and normally are more subject to budget constraints. Cisco IOS Firewall fits quite well in these scenarios as part of the *all-in-one* solution provided by *Integrated Services Routers (ISRs)*. Among the services available in ISRs, some deserve specific reference:

- **Multiprotocol routing:** This is doubtlessly a recognized domain of excellence for Cisco IOS Software. Besides the variety of supported unicast routing protocols, IP Multicast, QoS handling, *Generic Routing Encapsulation (GRE)*, and VRF support are resources that deserve special mention. *First Hop Redundancy Protocols* (HSRP, VRRP, and GLBP), IP SLAs, object tracking, and powerful backup techniques (using several access technologies) are just a small sample of the features included in the flexible Cisco IOS toolbox.

- **Stateful inspection with high level of application awareness:** IOS Firewall goes far beyond regular packet filtering, offering true stateful Inspection functionality, which can include parameters from L3 to L7 in the analysis. L7 inspection is the subject of Chapter 12, “Application Inspection.”
- **Identity services:** The *Authentication Proxy* feature can be used with IOS Firewall’s stateful inspection functionality to provide security services on a per-user basis.
- **Wide spectrum of secure connectivity models:** Spanning from classic IPSEC and EasyVPN to DMVPN and GET VPN, IOS also offers SSL VPN termination. These VPN options bring unparalleled flexibility for the network and security architects.
- **Integrated intrusion prevention:** Ranging from IOS IPS to dedicated IPS service modules (in AIM or NM format), this extra layer of security can be added to traffic inspected by the IOS Firewall.
- **Convergence services:** Cisco always sponsored network convergence, supporting VoIP, VoFR, and VoATM since the early days and later including not only the Voice gateway and gatekeeper features but also *Call Manager Express* functionality in IOS. Because of the importance of Unified Communications (UC) for companies, Chapter 13, “Inspection of Voice Protocols,” is dedicated to the analysis of Firewalls in the UC World.

CBAC Basics

CBAC understands the characteristics of many application layer protocols and uses this knowledge to provide intelligent inspection services at higher OSI layers. CBAC, in its simplest mode of operation, may also be used to provide *generic stateful inspection* of TCP and UDP traffic, which is adequate for the analysis of well-behaved applications that employ *single channel* sessions.

When performing generic stateful inspection, CBAC focuses on L3 and L4 parameters, keeping track of state information for each connection established through it. The attributes held in the state table enable CBAC to dynamically create *temporary openings* to enable the pertinent *return traffic*.

This chapter is devoted to generic stateful inspection, and the decision to follow this approach is based on two underlying goals:

- It promotes an overview of CBAC operations without delving into specific behaviors of more complex application protocols.
- It facilitates comparison with the ASA algorithm (refer to Chapter 7, “Through ASA Without NAT,” and Chapter 8, “Through ASA Using NAT”).

Later in the book (Chapters 12 and 13), FTP (which negotiates a data connection via a control session) and more complex protocols (such as those related to voice and video transport) are examined in more detail. Further, the analysis of how to enhance the security provided by a stateful firewall by means of identity verification is provided in Chapter 14, “Identity on Cisco Firewalls.”

Before starting the analysis of any connection examples, it is instructive to review some key aspects of IOS and ASA philosophies and history so that similarities and differences about the implementation of stateful inspection can be more easily grasped. Some of the most significant aspects follow:

- The PIX Firewall, ancestor of the ASA-family, was launched in the marketplace as a purpose-built firewall appliance, inherently stateful for TCP and UDP. As studied in Chapters 7 and 8, the ASA algorithm has an implicit deny for inbound connections, and whenever **nat-control** is enabled, another layer of defense (the need for a match for translation) is added. Application knowledge was substantially increased over time while other functionalities (VPN for instance) were sequentially made available for customers. This natural evolution of the product culminated in the launch of the ASA family of multifunctional security appliances.
- As the Internet Protocol (IP) became almost omnipresent, attacks on it proliferated. Cisco then realized that its routers, already processing IP packets (for routing and forwarding purposes), could have their functionality expanded to network security. The stateful inspection solution known as *CBAC* was born.
- Although the stateful firewall capabilities of CBAC are similar to those of ASA, there is no implicit deny configuration in IOS. This is because IOS is the heart of routers, a class of network equipments conceived with the underlying objective of providing connectivity. If the goal is to turn this device into a security *policy enforcement point*, more configuration effort is needed.
- Although the ASA algorithm is intrinsically stateful, the explicit configuration of CBAC is required to render IOS stateful, even for a generic inspection of TCP, UDP, and ICMP.
- There is no concept such as **nat-control** in the Classic IOS Firewall. CBAC does support Network Address Translation (NAT) techniques, but everything needs to be defined manually.

Figure 9-1 portrays the basic operation of CBAC to provide generic stateful inspection on a simple *Internet Access* firewall topology. In this scenario

- TCP connections initiated by inside clients are visible to the CBAC software module (the inspection rule named INSPECT1 is applied to incoming packets).
- For each TCP packet arriving at interface *F1*, CBAC analyzes the basic flow attributes (source /destination addresses and protocol ports) and verifies if it belongs to a pre-existent connection. If this is a new flow, CBAC dynamically creates an entry in its state table. A correspondent temporary opening, with *reversed flow parameters*, is built to bypass the deny all settings of ACL 100.
- As return packets arrive at interface *F0*, their *flow* attributes are compared with the *temporary openings* and, if a match is found, they are allowed back to the originating client.

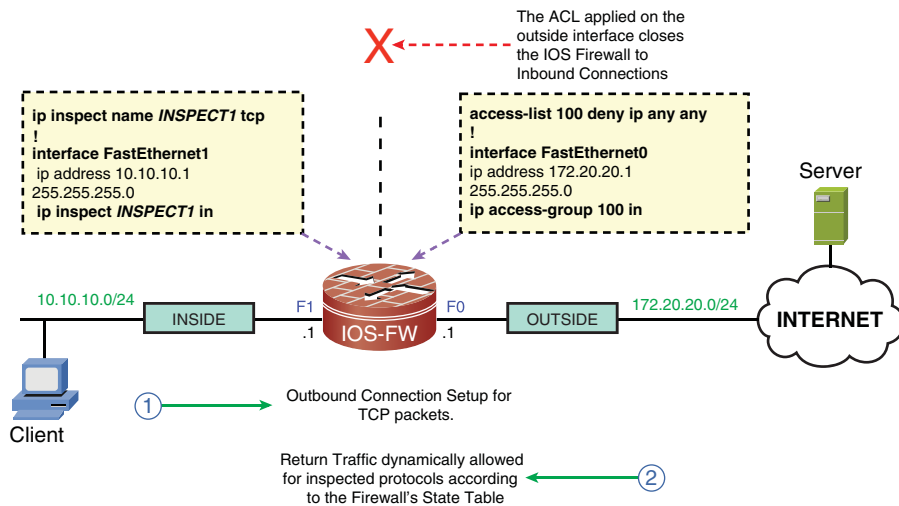


Figure 9-1 Sample Internet Access Topology Using the Classic IOS Firewall (CBAC)

- If the *inactivity timeout* for TCP inspection is reached for a given connection, the entry in the *state table* is deleted and the *temporary opening* is automatically removed.
- Packets that arrive at interface *F0* and do not fit as *return traffic* for any *outbound connection*, are dropped by *ACL 100*.
- Non-TCP outgoing packets are allowed to flow through the IOS-FW, but because they do not trigger the creation of a *temporary opening*, the corresponding return packets are dropped by *ACL 100*.

Note As will be seen on Examples 9-8 and 9-9, CBAC tracks other parameters such as TCP flags and sequence numbers. It also contributes to the mitigation of Denial of Service (DoS) attacks, by enforcing connection limits on a per-host basis.

Note The terms *outbound* and *inbound* are not so strict for the Classic IOS Firewall as they are for the ASA algorithm (which derives this reference by comparing security levels). If NAT is not used with CBAC, inside and outside are simply relative terms defined by the firewall administrator in association with the internal and external networks, respectively (and not enforced). When NAT is in place, the **ip nat inside** and **ip nat outside** commands naturally establish the reference of internal and external address spaces, as studied later in this chapter.

ICMP Connection Examples

Although ICMP does not define a *state machine*, an IOS Firewall can be instructed to handle connections associated with this protocol as if they were natively stateful.

This section uses the reference topology depicted in Figure 9-2 and the corresponding baseline configuration of Example 9-1 to cover some examples of ICMP flows crossing an IOS router enabled for CBAC.

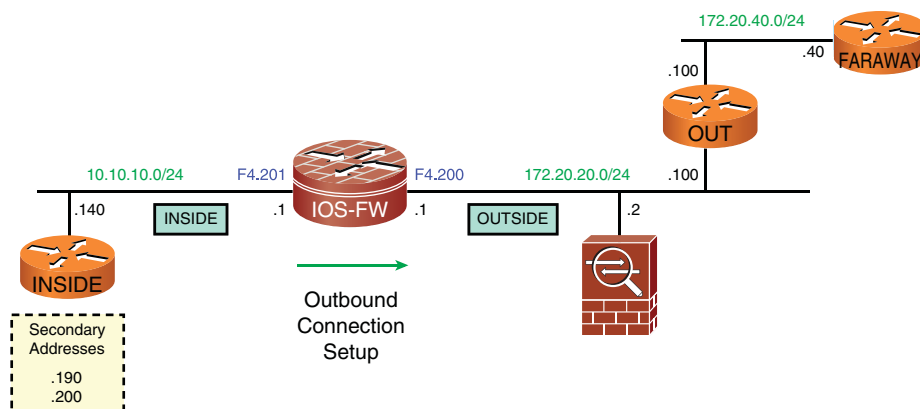


Figure 9-2 Reference Topology for Initial Examples

As stated before, CBAC is enabled on a per-protocol basis, even when performing solely *generic stateful inspection*. The most basic requirement for defining a CBAC rule, is the use of the **ip inspect name** command. More specifically, as documented in Example 9-1:

- The **ip inspect name INSPECT1 icmp** command instructs CBAC to inspect ICMP connection requests.
- When set to **on**, the **audit-trail** parameter instructs CBAC to issue syslog messages related to session creation and termination. As is shown in several examples, the **audit-trail stop session** message also reveals the number of bytes transmitted.
- When set to **on**, the **alert** parameter logs events such as the maximum number of TCP connections being reached.
- The **router-traffic** parameter includes the inspection of packets destined to one of the interface IP addresses of the CBAC-enabled router.
- The definitions of the **ip inspect name INSPECT1** command become effective only after the command **ip inspect INSPECT1** is applied to a router interface in the **in** or **out** direction. In the examples that follow, CBAC inspection is always performed for incoming packets.

- Besides restricting inbound connection initiation, the purpose of applying a restrictive ACL (**access-list 100**) on the outside interface is to make it more easily visible how CBAC inserts the temporary opening for return traffic.

The **permit** statement for ICMP *unreachable* and *time-exceeded* messages on ACL 100 is aimed to enable the use of the UDP traceroute mechanism (employed by systems such as UNIX and IOS) through the Classic IOS Firewall. (Example 9-5 shows sample IOS traceroute tests.) Example 9-2 shows the behavior of CBAC when an echo request packet sent by the inside host 10.10.10.200 to the outside destination 172.20.20.2 arrives at interface F4.201. The **debug ip inspect object-creation**, used to unveil CBAC operation, demonstrates that ICMP inspection creates openings not only for the echo reply (Type 0/Code 0) but also for the other two ICMP messages used for the *traceroute* functionality. As studied in Chapter 7, these messages follow:

- (Type 3/Code 3) : Port unreachable
- (Type 11/Code 11): Time exceeded

Example 9-3 illustrates the CBAC operation for a *Ping* packet destined to the router interface address, 10.10.10.1.

Example 9-1 Baseline Configuration

```
! ACL to be applied on the OUTSIDE (no inbound connection setup)
access-list 100 permit icmp any 10.10.10.0 0.0.0.255 unreachable log
access-list 100 permit icmp any 10.10.10.0 0.0.0.255 time-exceeded log
access-list 100 deny ip any any log
!
! Policy INSPECT1 provides generic inspection of ICMP, UDP and TCP (bound to f4.201)
ip inspect name INSPECT1 icmp alert on audit-trail on router-traffic
ip inspect name INSPECT1 udp alert on audit-trail on router-traffic
ip inspect name INSPECT1 tcp alert on audit-trail on router-traffic
!
! OUTSIDE interface (no connection initiation from hosts on the outside)
interface FastEthernet4.200
description *** OUTSIDE
encapsulation dot1Q 200
ip address 172.20.20.1 255.255.255.0
ip access-group 100 in
ip virtual-reassembly
!
! INSIDE interface ( stateful policy INSPECT1 handles the return traffic)
interface FastEthernet4.201
description *** INSIDE
encapsulation dot1Q 201
```

```
ip address 10.10.10.1 255.255.255.0
ip inspect INSPECT1 in
ip virtual-reassembly
```

Example 9-2 Sample Ping Session

```
! Ping from INSIDE host (10.10.10.200) to host 172.20.20.2
INSIDE# ping 172.20.20.2 source 10.10.10.200 repeat 1 size 300
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
INSIDE# ICMP: echo reply rcvd, src 172.20.20.2, dst 10.10.10.200
!
! CBAC creates the temporary opening for return traffic
FIREWALL OBJ_CREATE: Pak 84030844 sis 8469F268
initiator_addr (10.10.10.200:8) responder_addr (172.20.20.2:0)
initiator_alt_addr (10.10.10.200:8) responder_alt_addr (172.20.20.2:0)
%FW-6-SESS_AUDIT_TRAIL_START: Start icmp session: initiator (10.10.10.200:8)
— responder (172.20.20.2:0)
FIREWALL icmp_info created: 0x846A0BB4
FIREWALL OBJ_CREATE: sid 846B0AFC acl 100 Prot: icmp
  Src 172.20.20.2 Port [0:0]
  Dst 10.10.10.200 Port [0:0]
FIREWALL OBJ_CREATE: sid 846B0B50 acl 100 Prot: icmp
  Src 0.0.0.0 Port [0:0]
  Dst 10.10.10.200 Port [3:3]
FIREWALL OBJ_CREATE: sid 846B0BA4 acl 100 Prot: icmp
  Src 0.0.0.0 Port [0:0]
  Dst 10.10.10.200 Port [11:11]
!
IOS-FW# show ip inspect sessions
Established Sessions
  Session 8469F268 (10.10.10.200:8)=>(172.20.20.2:0) icmp SIS_OPEN
!
! This command shows the temporary opening in ACL 100
IOS-FW# show ip inspect sis detail
Established Sessions
  Session 8469F268 (10.10.10.200:8)=>(172.20.20.2:0) icmp SIS_OPEN
    Created 00:00:07, Last heard 00:00:07
    ECHO request
    Bytes sent (initiator:responder) [272:272]
      Initiator->Responder Window size 0 Scale factor 0
      Responder->Initiator Window size 0 Scale factor 0
    In  SID 172.20.20.2[0:0]=>10.10.10.200[0:0] on ACL 100 (1 matches)
    In  SID 0.0.0.0[0:0]=>10.10.10.200[3:3] on ACL 100
    In  SID 0.0.0.0[0:0]=>10.10.10.200[11:11] on ACL 100
```

```

!
! Connection teardown and audit-trail
FIREWALL OBJ_DELETE: delete sis 8469F268
%FW-6-SESS_AUDIT_TRAIL: Stop icmp session: initiator (10.10.10.200:8) sent 272 bytes
— responder (172.20.20.2:0) sent 272 bytes

```

Example 9-3 Sample Ping Session to the Router Interface Address

```

! Ping from INSIDE host (10.10.10.200) to the IOS-FW interface address
(10.10.10.1)
INSIDE# ping 10.10.10.1 source 10.10.10.200 repeat 1 size 300
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
INSIDE# ICMP: echo reply rcvd, src 10.10.10.1, dst 10.10.10.200
!
! CBAC creates the session
FIREWALL OBJ_CREATE: Pak 840313C0 sis 8469F268
initiator_addr (10.10.10.200:8) responder_addr (10.10.10.1:0)
initiator_alt_addr (10.10.10.200:8) responder_alt_addr (10.10.10.1:0)
%FW-6-SESS_AUDIT_TRAIL_START: Start icmp session: initiator (10.10.10.200:8)
— responder (10.10.10.1:0)
FIREWALL icmp_info created: 0x846A0BB4
!
IOS-FW# show ip inspect sessions
Half-open Sessions
  Session 8469F268 (10.10.10.200:8)=>(10.10.10.1:0) icmp SIS_OPENING

```

UDP Connection Examples

The inspection of UDP packets intended to cross the *Classic IOS Firewall* tracks six of the *seven key fields* associated with *flows*. (Only the ToS is not analyzed.) Because UDP does not define a finite state machine and has no concept of connection flags, CBAC relies on timing information (*inactivity timeouts*) to dynamically close the *temporary openings* built for UDP return traffic.

This section uses the reference topology of Figure 9-2 to study the creation of some UDP connections through a CBAC-based firewall.

In Example 9-4, the NTP Client 10.10.10.140 requests timing information from the NTP Server 172.20.20.1 (outside address of the CBAC router).

Example 9-5 exhibits two **traceroute** tasks started from the INSIDE router and destined to the OUT and FARAWAY routers respectively (refer to Figure 9-2). As discussed in Chapter 7, the IOS **traceroute** utility employs UDP probes and relies on the ICMP *time exceeded*

and *port unreachable* messages to determine the hops along the path from source to destination. The example also shows the hits in the **permit** statement of ACL 100:

- An ICMP Type3/Code3 message issued by the last hop 172.20.20.100 when tracing the route to the OUT router.
- An ICMP Type 3/Code3 message sent by the last hop 172.20.40.40 (FARAWAY) and a time exceeded message (Type 11/Code0) issued by the intermediary hop 172.20.20.100 in the second **traceroute** test.

The UDP probes used for IOS traceroute are decoupled from the ICMP error messages sent by hosts along the path. As a result, the additional openings automatically created for ping sessions (as shown in Example 9-2) are not enough to deal with UDP-based tracing activities. In this type of scenario, ACLs explicitly permitting the ICMP error messages (3/3 and 11/0) become necessary.

Example 9-4 Sample NTP Session

```

! Connection Establishment
FIREWALL OBJ_CREATE: Pak 83B23074 sis 84AF8458
initiator_addr (10.10.10.140:123) responder_addr (172.20.20.1:123)
initiator_alt_addr (10.10.10.140:123) responder_alt_addr (172.20.20.1:123)
!
%FW-6-SESS_AUDIT_TRAIL_START: Start udp session: initiator (10.10.10.140:123)
— responder (172.20.20.1:123)
!
! Viewing established sessions
IOS-FW# show ip inspect sessions
Established Sessions
  Session 84AF8458 (10.10.10.140:123)=>(172.20.20.1:123) udp SIS_OPEN
!
IOS-FW# show ip inspect sis detail
Established Sessions
  Session 84AF8458 (10.10.10.140:123)=>(172.20.20.1:123) udp SIS_OPEN
    Created 00:00:26, Last heard 00:00:23
    Bytes sent (initiator:responder) [96:48]
    Initiator->Responder Window size 0 Scale factor 0
    Responder->Initiator Window size 0 Scale factor 0
!
! Connection teardown as registered by the audit-trail
%FW-6-SESS_AUDIT_TRAIL: Stop udp session: initiator (10.10.10.140:123) sent 96 bytes
— responder (172.20.20.1:123) sent 48 bytes

```

Example 9-5 *Sample IOS Traceroute Through CBAC*

```

! Tracing the route to 172.20.20.100 (from IOS host 10.10.10.190)
INSIDE# traceroute 172.20.20.100 source 10.10.10.190 probe 1
Tracing the route to 172.20.20.100
  1 10.10.10.1 0 msec
  2 172.20.20.100 4 msec
ICMP: time exceeded rcvd from 10.10.10.1
ICMP: dst (10.10.10.190) port unreachable rcv from 172.20.20.100
!
! IOS Firewall sees UDP connection and auxiliary ICMP packets
FIREWALL: ICMP Time Exceeded pkt 10.10.10.1 => 10.10.10.190
%FW-6-SESS_AUDIT_TRAIL_START: Start udp session: initiator (10.10.10.190:49200)
  - responder (172.20.20.100:33435)
FIREWALL* UDP: sis 84AF8458 pak 83E92E70 SIS_CLOSED UDP packet
(10.10.10.190:49200) => (172.20.20.100:33435) datalen 0
FIREWALL: ICMP Unreachable pkt 172.20.20.100 => 10.10.10.190
!
! ICMP Type 3/Code 3 message needs to be explicitly permitted for IOS traceroute
%SEC-6-IPACCESSLOGDP: list 100 permitted icmp 172.20.20.100 -> 10.10.10.190 (3/3),
1 packet
!
! Second traceroute test: tracing the route to 172.20.40.40 (FARAWAY router)

INSIDE# traceroute 172.20.40.40 source 10.10.10.190 probe 1
Tracing the route to 172.20.40.40
  1 10.10.10.1 4 msec
  2 172.20.20.100 4 msec
  3 172.20.40.40 4 msec
ICMP: time exceeded rcvd from 10.10.10.1
ICMP: time exceeded rcvd from 172.20.20.100
ICMP: dst (10.10.10.190) port unreachable rcv from 172.20.40.40

! IOS-FW sees the setup of UDP sessions and the ICMP auxiliary messages
ICMP: time exceeded (time to live) sent to 10.10.10.190 (dest was 172.20.40.40)
FIREWALL* OBJ_CREATE: Pak 47CE3220 sis 481FDF68 initiator_addr
(10.10.10.190:49175) responder_addr (172.20.40.40:33435)
initiator_alt_addr (10.10.10.190:49175) responder_alt_addr (172.20.40.40:33435)
%FW-6-SESS_AUDIT_TRAIL_START: Start udp session: initiator (10.10.10.190:49175) -
responder (172.20.40.40:33435)
FIREWALL OBJ-CREATE: sid 48A58C64 acl 100 Prot: udp
  Src 172.20.40.40 Port [33435:33435]
  Dst 10.10.10.190 Port [49175:49175]
%SEC-6-IPACCESSLOGDP: list 100 permitted icmp 172.20.20.100 -> 10.10.10.190
(11/0), 1 packet
FIREWALL* OBJ_CREATE: Pak 47CE3220 sis 481FE230 initiator_addr
(10.10.10.190:49176) responder_addr (172.20.40.40:33436)
initiator_alt_addr (10.10.10.190:49176) responder_alt_addr (172.20.40.40:33436)

```



```
%FW-6-SESS_AUDIT_TRAIL_START: Start udp session: initiator (10.10.10.190:49176) —
responder (172.20.40.40:33436)
FIREWALL OBJ-CREATE: sid 48A58CB8 acl 100 Prot: udp
  Src 172.20.40.40 Port [33436:33436]
  Dst 10.10.10.190 Port [49176:49176]
%SEC-6-IPACCESSLOGDP: list 100 permitted icmp 172.20.40.40 -> 10.10.10.190
(3/3), 1 packet
```

TCP Connection Examples

TCP defines a *finite state machine* that specifies, for instance, what are the expected state transitions associated with *connection setup* and *termination*. When dealing with TCP connections, the *Classic IOS firewall* has visibility not only of the protocol, IP address, and port information embedded in the flow, but also of the *TCP flags* and *sequence numbers* (SEQ).

Example 9-6 illustrates the setup of a Telnet session from inside host 10.10.10.140 to the destination 172.20.20.2. The **debug ip inspect protocol tcp** was turned on with the intent of registering CBAC's awareness of the *sequence numbers* and the corresponding Acknowledgment values (ACK) involved in the TCP three-way handshake.

Example 9-6 Sample Telnet Session Through CBAC

```
! Host 10.10.10.140 telnets to 172.20.20.2 (ASA management address)
INSIDE# telnet 172.20.20.2
Trying 172.20.20.2 ... Open
Reserved port 14338 in Transport Port Agent for TCP IP type 1
TCP: sending SYN, seq 3637270155, ack 0
TCP2: Connection to 172.20.20.2:23, advertising MSS 536
TCP2: state was CLOSED -> SYNSENT [14338 -> 172.20.20.2(23)]
TCP2: state was SYNSENT -> ESTAB [14338 -> 172.20.20.2(23)]
TCP: tcb 83D5B8B8 connection to 172.20.20.2:23, peer MSS 1380, MSS is 536
TCB83D5B8B8 connected to 172.20.20.2.23
User Access Verification
Password: *****
ASA2> enable
Password:*****
ASA2# show conn all
1 in use, 6 most used
TCP dmz 10.10.10.140:14338 NP Identity Ifc 172.20.20.2:23, idle 0:00:00, bytes
309, flags UOB
!
! Connection Establishment through CBAC
%FW-6-SESS_AUDIT_TRAIL_START: Start tcp session: initiator (10.10.10.140:14338)
— responder (172.20.20.2:23)
FIREWALL* sis 84AF8458 pak 83E92E70 SIS_CLOSED/LISTEN TCP SYN SEQ 3637270155
```

```

LEN 0 (10.10.10.140:14338) => (172.20.20.2:23)
FIREWALL* sis 84AF8458 pak 83E92E70 SIS_OPENING/SYNSENT TCP SYN ACK 3637270156
SEQ 739284105 LEN 0 (172.20.20.2:23) <= (10.10.10.140:14338)
FIREWALL* sis 84AF8458 pak 83E92E70 SIS_OPENING/SYNRCVD TCP ACK 739284106
SEQ 3637270156 LEN 0 (10.10.10.140:14338) => (172.20.20.2:23)

```

Figure 9-3 depicts the reference topology that serves as the base for the study of some additional TCP security features available in the Classic IOS Firewall. Example 9-7 shows the configuration commands that correspond to the scenario represented in the figure. Some points that deserve special notice follow:

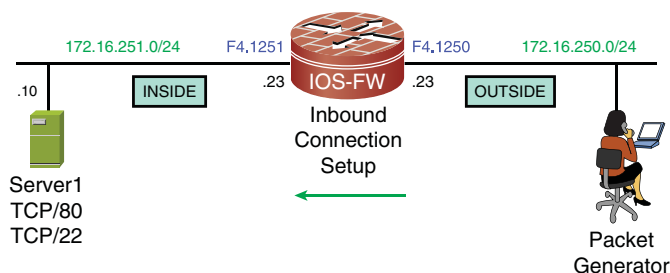


Figure 9-3 *Reference Topology for Advanced TCP Examples*

- Overall, this environment is similar to that in Figure 9.2. Besides the addressing differences, the current topology employs CBAC for inbound connections rather than for outbound.
- The new CBAC rule is named **INSPECT2** and dedicated to TCP inspection.
- The purpose of **ACL 101** is to close the firewall for outbound connections.
- The **ip inspect log drop-pkt** command was added with the goal of revealing the cause of CBAC packet drops (by means of syslog messages).
- The **ip inspect tcp block-non-session** command is in charge of dropping connection attempts that do not belong to any of the existing TCP sessions. For instance, if a TCP segment containing flags other than the SYN is received and does not have an associated entry in the state table, it will not be allowed through the firewall.

Note It is a common practice to avoid that internal servers initiate outbound connections, particularly when the outside world means the global Internet. This makes sense because, in the event a server becomes compromised, it will be more difficult for the attacker to establish outbound connections and download tools to the internal hosts (or illegally uploading internal content to an external repository).

Example 9-7 *Baseline Configuration for Advanced TCP Examples*

```

! ACL to be applied on the INSIDE interface (no outbound connection setup)
access-list 101 deny ip any any log
!
! Logging the packets dropped by the IOS Firewall
ip inspect log drop-pkt
!
! Blocking initial TCP packets with flags other than the SYN
ip inspect tcp block-non-session
!
! Inspection Policy for TCP
ip inspect name INSPECT2 tcp alert on audit-trail off router-traffic
!
interface FastEthernet4.1250
 encapsulation dot1Q 1250
 ip address 172.16.250.23 255.255.255.0
 ip inspect INSPECT2 in
!
interface FastEthernet4.1251
 encapsulation dot1Q 1251
 ip address 172.16.251.23 255.255.255.0
 ip access-group 101 in

```

Example 9-8 relates to the topology in Figure 9-3 and portrays some TCP enforcement mechanisms, provided by CBAC, in action:

- An SSH connection attempt with no corresponding entry in the state table is blocked because it contains inappropriate flags (0x21 = 33, meaning that the URG and FIN flags were set). If a TCP connection is built, the initial packet should contain only the SYN flag marked.
- An HTTP packet with the SYN flag (0x02) is received, which normally represents the beginning of the three-way-handshake process. The problem in this case relates to the fact that this initial packet contained data, thus not complying with the behavior described in the TCP RFC.
- An SSH packet bearing illegal TCP flags (0x03 = 3, which means that the SYN and FIN bits are set) is dropped.

These were just a few practical examples of CBAC capabilities for generic inspection of TCP. Other noteworthy available protection resources follow:

- Blocking invalid TCP segments and segments that contain invalid TCP options.
- Blocking *out-of-order* TCP segments.

- Dropping segments with inconsistent values either in the sequence number or in the Acknowledgement fields.
- Terminating a connection if one of the parties involved proposes an illegal *Window* scaling.
- Dropping IP packets whose length value is not adequate for carrying a standard TCP segment.

Example 9-8 Sample TCP Enforcement Performed by CBAC

```

! Dropping a TCP packet that is not part of a connection
%FW-6-DROP_PKT: Dropping tcp session 172.16.250.110:10030
172.16.251.10:22 due to Segment matching no TCP connection with ip
ident 0 tcpflags0x5021 seq.no 6113711 ack 0
!
! Dropping a SYN Packet that contains data
%FW-6-DROP_PKT: Dropping tcp session 172.16.250.100:10000
172.16.251.10:80 due to SYN with data or with PSH/URG flags with ip
ident 0 tcpflags 0x5002 seq.no9763214 ack 0
!
! Dropping a TCP packet with an illegal combination of flags
%FW-6-DROP_PKT: Dropping tcp session 172.16.250.100:10000
172.16.251.10:22 due to SYN pkt with illegal flags with ip ident 0
tcpflags0x5003 seq.no 7903414 ack 0

```

Example 9-9 shows how CBAC can be configured to enforce an upper bound for the number of half-open connections for individual hosts, thus contributing for DoS attacks mitigation. In the first case, a limit of 20 simultaneous incomplete connections was set.

Another practical application was to define a limit of 600 embryonic connections with a **block-time** of 1 minute upon crossing of this threshold. The example shows this value being reached, the host being kept from handling new requests during 1 minute and how it comes back to service after this interval ends.

CBAC offers the possibility of setting the time (in seconds) that the software will wait for a connection to be complete (**ip inspect tcp synwait-time**). Likewise, the **ip inspect tcp finwait-time** is configurable and enables the specification of how long an existent connection will keep being managed after the firewall detects a FIN exchange.

Example 9-9 Limiting the Number of Incomplete TCP Connections

```

! Setting a maximum of 20 incomplete (half-open) TCP connections for a host
ip inspect tcp max-incomplete host 20 block-time 0
!
! After the limit is reached, an ALERT is generated by CBAC

```

```
%FW-6-SESS_AUDIT_TRAIL_START: Start tcp session: initiator (172.16.250.100:10000)
— responder (172.16.251.10:80)
%FW-6-SESS_AUDIT_TRAIL_START: Start tcp session: initiator (172.16.250.101:10001)
— responder (172.16.251.10:80)
[ output suppressed ]
%FW-6-SESS_AUDIT_TRAIL_START: Start tcp session: initiator (172.16.250.109:10019)
— responder (172.16.251.10:80)
%FW-6-SESS_AUDIT_TRAIL_START: Start tcp session: initiator (172.16.250.100:10020)
— responder (172.16.251.10:80)
%FW-6-SESS_AUDIT_TRAIL: Stop tcp session: initiator (172.16.250.100:10020) sent
4284588298 bytes — responder (172.16.251.10:80) sent 0 bytes
%FW-4-HOST_TCP_ALERT_ON: Max tcp half-open connections (20) exceeded for host
172.16.251.10
!
! Blocking new connections in case the limit of incomplete connections is reached
ip inspect tcp max-incomplete host 600 block-time 1
!
! New connections to host 172.16.251.10 are blocked for 1 minute
%FW-4-HOST_TCP_ALERT_ON: Max tcp half-open connections (600) exceeded for host
172.16.251.10
%FW-2-BLOCK_HOST: Blocking new TCP connections to host 172.16.251.10 for 1 minute
(half-open count 600 exceeded).
!
! Unblocking the host for new TCP connections after 1 minute
%FW-4-UNBLOCK_HOST: New TCP connections to host 172.16.251.10 no longer blocked
```

Note Another DoS protection provided by CBAC is the possibility of defining an acceptable rate for the admission of half-open connections. The pertinent commands for this task are **ip inspect one-minute high** and **ip inspect one-minute low**, whose default values are respectively 500 and 400 incomplete connections per minute.

Tip For UDP the concept of half-open means that CBAC has seen traffic only in one direction. The **ip inspect one-minute** commands involve the aggregate number of half-open (UDP and TCP) connections for an individual host.

Handling ACLs and Object-Groups

Access Control Lists (ACL) are an important building block of firewall policies. IOS support for ACLs has greatly increased through time and today several categories of lists are available. The main type of ACL to be familiar with are the so called *extended ACLs*, which include not only source and destination addresses but also service ports among their filtering criteria.

Example 9-10 highlights some important aspects of IOS extended ACLs:

- IOS relies on a *wildcard mask* rather than on the *network mask* used by ASA for source and destination address specification. This approach renders IOS more flexible for IP filtering. The network/wildcard pair 10.1.1.0/0.128.0.255 exemplifies a summarized way of defining hosts on subnets 10.1.1.0/24 and 10.129.0.0/24. In an analogous fashion, the pair 10.20.20.0/0.67.255 is a condensed manner of representing the hosts on the 10.20.[20-23].0/24 and 10.20.[84-87].0/24 subnets.
- The basic filtering structure for ACLs is *from source to destination*. This is further detailed in Figure 9-4.

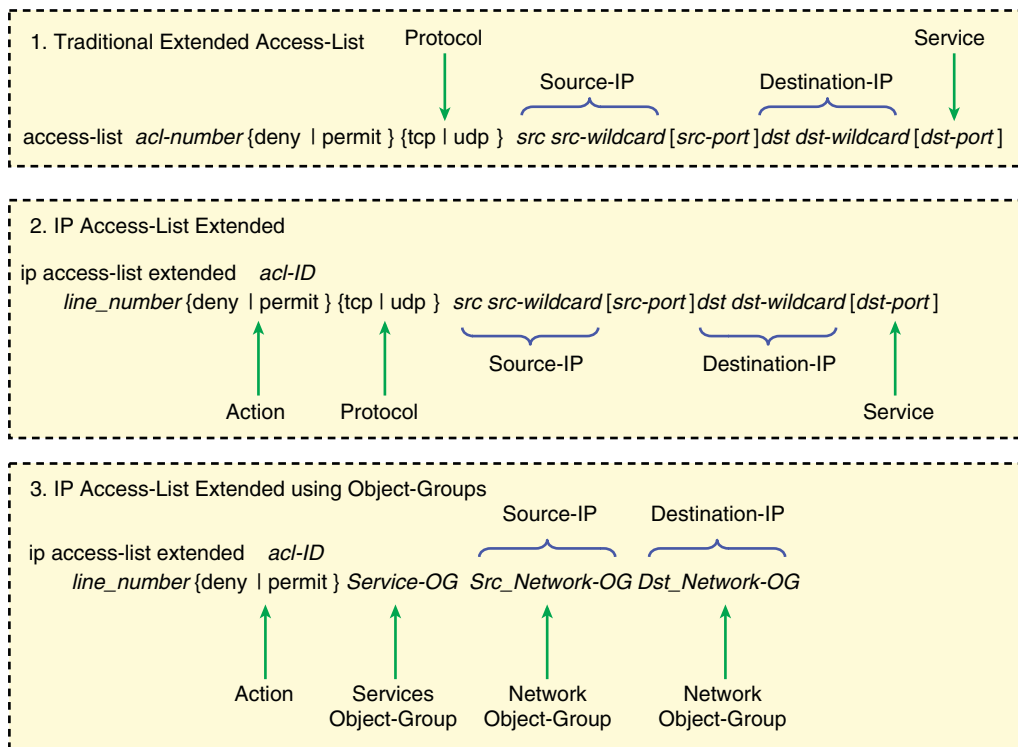


Figure 9-4 Basic Structure of IOS ACLs

Example 9-10 Traditional Way of Defining IOS Extended Access Control Lists

```
! Extended Access-list defined in the traditional way
access-list 120 permit tcp 10.10.10.0 0.0.0.255 host 172.20.20.2 eq tacacs log
access-list 120 remark * Source Networks 10.10.10.0/24 and 10.10.11.0/24
access-list 120 permit udp 10.10.10.0 0.0.1.255 host 172.20.20.1 range 1812 1813
access-list 120 permit icmp host 10.10.10.190 host 172.20.20.1 unreachable
```

```

!
IOS-FW# show access-list 120
Extended IP access list 120
    10 permit tcp 10.10.10.0 0.0.0.255 host 172.20.20.2 eq tacacs log
    20 permit udp 10.10.10.0 0.0.1.255 host 172.20.20.1 range 1812 1813
    30 permit icmp host 10.10.10.190 host 172.20.20.1 unreachable
!
! Inserting lines in the ACL and displaying its updated version
IOS-FW(config)# ip access-list extended 120
IOS-FW(config-ext-nacl)# 9 permit esp 10.1.1.0 0.128.0.255 host 172.20.20.2
IOS-FW(config-ext-nacl)# 19 permit gre 10.20.20.0 0.0.67.255 host 172.20.20.20
!
IOS-FW# show access-list 120
Extended IP access list 120
    9 permit esp 10.1.1.0 0.128.0.255 host 172.20.20.2
    10 permit tcp 10.10.10.0 0.0.0.255 host 172.20.20.2 eq tacacs log
    19 permit gre 10.20.20.0 0.0.67.255 host 172.20.20.20
    20 permit udp 10.10.10.0 0.0.1.255 host 172.20.20.1 range 1812 1813
    30 permit icmp host 10.10.10.190 host 172.20.20.1 unreachable

```

Using Object-Groups with ACLs

The purpose of **object-groups** is to facilitate the creation and maintenance of ACLs. They are of value for security administrators who need to deal with large or frequently changing access control lists.

Example 9-11 shows some sample network **object-groups** used to define sets of hosts or networks in any combination. Because the basic structure of ACLs involves the definition of permissions *from source to destination*, you need to have this notion of direction in mind when building the groupings.

Example 9-12 illustrates some service **object-groups** for IOS. Although using different syntaxes, these groups are directly comparable to the enhanced service object-groups already presented for ASA (refer to the “Handling ACLs and Object-Groups” section in Chapter 7, Through ASA Without NAT”).

Example 9-11 Sample Network Object-Groups

```

object-group network INSIDE
  10.10.10.0 255.255.255.0
!
object-group network OUT1
  172.20.20.0 255.255.255.0
  10.10.10.0 255.255.255.0
!
object-group network OUT2

```

```
description * Mixed Network Object-Group
range 172.21.21.21 172.21.21.22
host 172.21.21.29
group-object OUT1
```

Note The mask used in the definition of network **object-groups** is a network mask (instead of the wildcard mask used for ACLs).

Example 9-12 *Sample Service Object-Groups*

```
! Simple service group that contains only TCP ports
object-group service TCP-MGMT
  tcp range 22 telnet
  tcp eq www
!
! Simple service group that contains only UDP ports
object-group service UDP1
  udp eq ntp
!
! Simple service group that contains only the ICMP messages used for PING
object-group service PING
  icmp echo
  icmp echo-reply
!
! Sample Services Group that uses equal ports for TCP and UDP
object-group service SMB
  tcp-udp eq 139
  tcp-udp eq 445
!
! Sample Services Group that refers to previously defined groups
object-group service NESTED1
  group-object TCP-MGMT
  group-object UDP1
  group-object SMB
```

Example 9-13 shows sample ACLs constructed employing the **object-group** concept. The contents of this example might be further enhanced by the summary of IOS ACL structures presented in Figure 9-4.

Example 9-13 *Using Object-Groups to Define ACLs*

```
! Using only a Services Object-group ( and defining addresses directly in the ACL )
IOS-FW# show access-list OUTBOUND2
```



```

Extended IP access list OUTBOUND2
  10 permit object-group TCP-MGMT 10.41.30.0 0.6.0.255 host 172.20.20.2
!
! Using two object-groups in the ACL definition
IOS-FW# show access-list OUTBOUND3
Extended IP access list OUTBOUND3
  10 permit object-group NESTED1 10.40.72.0 0.0.7.255 object-group OUT2
!
! Using three object-groups in the ACL definition
IOS-FW# show access-list OUTBOUND
Extended IP access list OUTBOUND
  10 permit object-group TCP-MGMT object-group INSIDE object-group OUT1 log
  20 permit object-group PING object-group INSIDE object-group OUT1 log

```

Tip The pair 10.41.30.0/0.6.0.255 is a way to represent the 04 subnets with an odd value (from 41 up to 47) for the second octet. For the first and third octets, 10 and 30 are the values chosen, respectively. The last byte might vary freely from 0 to 255.

Note This section presented a quick review of IOS ACLs. If you need to become familiar with more parameters or want to know the formal description for every option, Cisco IOS *Security Command Reference* is the right place to visit.

CBAC and Access Control Lists

Now that you understand the main characteristics of CBAC and ACL resources available in IOS, it is time to mix inspection and packet filtering to build more sophisticated fire-wall policies.

Example 9-14 uses the ACL named OUTBOUND defined in Example 9-13 with the CBAC policy INSPECT1, which focuses on TCP and ICMP inspection. Some key points related to the integration of these security mechanisms follow:

- The OUTBOUND ACL is applied to filter *incoming packets* on the interface F4.201, which is the same direction in which stateful inspection takes place.
- The access-list OUTBOUND is checked before CBAC, therefore establishing that only traffic that it explicitly permits is later subject to inspection.
- When CBAC comes into play, it creates the *temporary openings for return traffic* on the ACL 100 (applied for incoming packets on interface F4.200, much like as on the initial examples of this chapter).
- When used for filtering purposes, IOS ACLs count packets and not connections. This is another remarkable difference for ASA ACLs.

Example 9-14 *ACLs Used with CBAC*

```

! Relevant configurations on the IOS Firewall
ip inspect name INSPECT1 tcp alert on audit-trail on router-traffic
ip inspect name INSPECT1 icmp alert on audit-trail on router-traffic
!
interface FastEthernet4.200
 ip address 172.20.20.1 255.255.255.0
 ip access-group 100 in
!
interface FastEthernet4.201
 ip address 10.10.10.1 255.255.255.0
 ip access-group OUTBOUND in
 ip inspect INSPECT1 in
!
! Host 10.10.10.140 opens an SSH session to 172.20.20.2
INSIDE# ssh -l admin 172.20.20.2
Password:*****
ASA2>
!
! ACL named 'OUTBOUND' permits traffic and CBAC creates connection
%SEC-6-IPACCESSLOGP: list OUTBOUND permitted tcp 10.10.10.140(23065) ->
172.20.20.2(22), 1 packet
FIREWALL OBJ_CREATE: Pak 8402298C sis 844C1840
initiator_addr (10.10.10.140:23065) responder_addr (172.20.20.2:22)
initiator_alt_addr (10.10.10.140:23065) responder_alt_addr (172.20.20.2:22)
%FW-6-SESS_AUDIT_TRAIL_START: Start tcp session: initiator (10.10.10.140:23065)
— responder (172.20.20.2:22)
FIREWALL OBJ_CREATE: sid 847AA53C acl 100 Prot: tcp
Src 172.20.20.2 Port [22:22]
Dst 10.10.10.140 Port [23065:23065]
!
! IOS ACLs count packets (instead of connections)
%SEC-6-IPACCESSLOGP: list OUTBOUND permitted tcp 10.10.10.140(23065) ->
172.20.20.2(22), 55 packets
!
IOS-FW# show access-list OUTBOUND ! include match
 10 permit object-group TCP-MGMT object-group INSIDE object-group OUT1 log
(56 matches)

```

IOS NAT Review

This section assumes you are familiar with the theoretical concepts pertaining to NAT. The purpose is to quickly revisit the NAT options available in Cisco IOS Software and later apply them with CBAC.

In the context of the NAT functionality implemented by IOS Software, *inside* and *outside* are two fundamental terms. The former refers to networks owned by a given organization (typically using private addresses), whereas the latter relates to networks not under the organization's control. For instance, whenever an inside host (with a real address) needs to contact outside hosts (which are assumed to use public addresses), it requires the NAT services provided by the IOS device.

IOS employs some specific terminology for the IP addresses involved in the translation process:

- **Inside Local:** The address assigned to a host residing on the inside network, which most of the time is private (from the RFC 1918 standpoint).
- **Inside Global:** The IP address of an inside host as seen on the outside domain.
- **Outside Local:** The IP address of an outside host as it appears to the inside domain.
- **Outside Global:** The IP address assigned to a host on the outside domain.

Local addresses are visible (and routable) on the inside network. Conversely, *global* addresses are visible and routable on the outside domain. The topologies drawn in Figures 9-5 and 9-6 might be helpful to clarify these concepts.

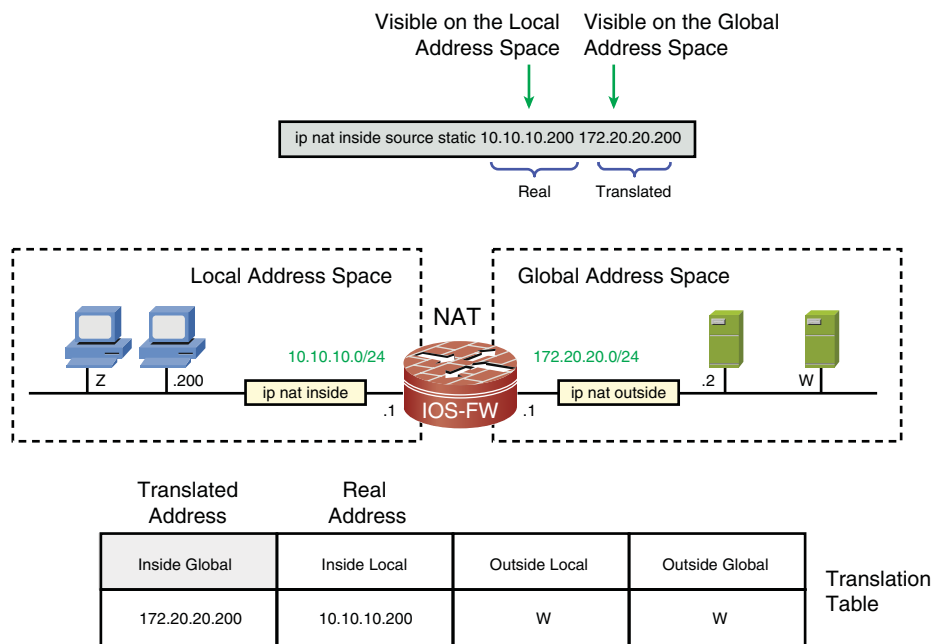


Figure 9-5 Sample Scenario for Inside Static NAT

Example 9-15 assembles the basic commands to establish the separation between the local and global address spaces on the topology represented in Figure 9-2. The `ip nat`

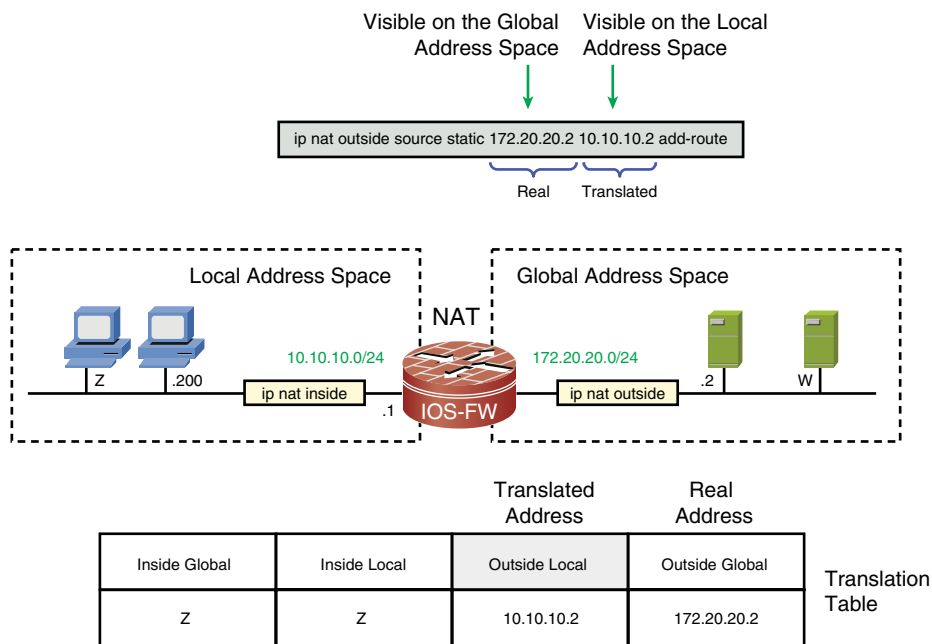


Figure 9-6 Sample Scenario for Outside Static NAT

inside command is applied to F4.201, therefore making this interface a member of the inside domain. In a similar way, the **ip nat outside** command is associated with F4.200, meaning that this interface belongs to the outside domain. The example also teaches how to instruct IOS to issue Syslog messages when a new translation is created. (This is accomplished using the **ip nat log translations syslog** command.)

Example 9-15 Baseline Configuration for NAT Analysis

```
! Relevant connectivity configurations for all NAT tests
interface FastEthernet4.200
ip address 172.20.20.1 255.255.255.0
ip nat outside
!
interface FastEthernet4.201
ip address 10.10.10.1 255.255.255.0
ip nat inside
!
! Instructing IOS to log address translations
ip nat log translations syslog
```

Note The reference (inside/outside) created in Example 9-15 will be used for all the examples presented in this section.

Static NAT

As previously studied in Chapter 8, “Through ASA Using NAT,” static NAT defines a permanent, one-to-one mapping between private (*local*) and public (*global*) addresses residing on two different interfaces of the firewall. Static NAT is bidirectional in nature, enabling connection initiation either from the inside or the outside domain.

Example 9-16 shows inside static NAT in action. The real host 10.10.10.200 is statically mapped to the *inside global* address 172.20.20.200 using the `ip nat inside source static` command. Two situations are illustrated:

- **Outbound ping:** The real host 10.10.10.200 sends an echo request to 172.20.20.2. This outside host sees packets sent by the real host (*inside local* address) as if they had been generated by 172.20.20.200 and therefore sends the echo reply to the latter (*inside global*).
- **Inbound ping:** The real host 172.20.20.2 resides in the outside domain and, as such, has connectivity only to the global address space. When this host sends a packet to the *inside global* address 172.20.20.200 (for which an inside static translation exists), IOS *untranslates* the destination address in the echo request to the real address 10.10.10.200.

Figure 9-5 details the scenario discussed in Example 9-16, clearly establishing the frontier between the *local* and *global* address spaces. Figure 9-6 achieves a similar result for the outside static NAT scenario analyzed in Example 9-17.

Example 9-16 Inside Static NAT

```
! Statically mapping the inside local address 10.10.10.200 to 172.20.20.200
ip nat inside source static 10.10.10.200 172.20.20.200
!
IOS-FW# show ip nat translations
Pro Inside global      Inside local      Outside local      Outside
global
--- 172.20.20.200      10.10.10.200      ---               ---
!
! Outbound ping from the inside local host 10.10.10.200 to 172.20.20.2
INSIDE# ping 172.20.20.2 source 10.10.10.200 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 1/1/1 ms
ICMP: echo reply rcvd, src 172.20.20.2, dst 10.10.10.200
!
! An ICMP specific translation is created after the PING
```

```
%IPNAT-6-CREATED: icmp 10.10.10.200:24 172.20.20.200:24 172.20.20.2:24
172.20.20.2:24
NAT*: s=10.10.10.200->172.20.20.200, d=172.20.20.2 [58]
NAT*: s=172.20.20.2, d=172.20.20.200->10.10.10.200 [2819]
!
IOS-FW# show ip nat translations
Pro Inside global      Inside local      Outside local      Outside
global
icmp 172.20.20.200:24  10.10.10.200:24   172.20.20.2:24
172.20.20.2:24
--- 172.20.20.200      10.10.10.200     ---               ---
!
! The destination host sees the translated (inside global) address
ICMP echo request from 172.20.20.200 to 172.20.20.2 ID=24 seq=0 len=72
ICMP echo reply from 172.20.20.2 to 172.20.20.200 ID=24 seq=0 len=72
!
! Inbound Ping to the inside global address 172.20.20.200 (from 172.20.20.2)
%IPNAT-6-CREATED: icmp 10.10.10.200:4388 172.20.20.200:4388 172.20.20.2:4388
172.20.20.2:4388
NAT*: s=172.20.20.2, d=172.20.20.200->10.10.10.200 [6966]
NAT*: s=10.10.10.200->172.20.20.200, d=172.20.20.2 [6966]
!
IOS-FW# show ip nat translations
Pro Inside global      Inside local      Outside local      Outside
global
icmp 172.20.20.200:4388 10.10.10.200:4388 172.20.20.2:4388
172.20.20.2:4388
--- 172.20.20.200      10.10.10.200     ---               ---
```

Example 9-17 registers the operation of outside static NAT. The real host 172.20.20.2 is statically mapped to the *outside local* address 10.10.10.2 by means of the **ip nat outside source static** command. Two sample ICMP connections are demonstrated:

- **Outbound ping:** The real host 10.10.10.140 is located in the inside domain and therefore has connectivity limited to the local address space. When this host sends an echo request to the *outside local* address 10.10.10.2 (for which an outside static translation exists), IOS *untranslates* the destination address in the echo request to the real address 172.20.20.2 (*outside global*).
- **Inbound ping:** The real host 172.20.20.2 sends an echo request to 10.10.10.190. This inside host sees packets sent by the real host (*outside global* address) as if they had been originated on 10.10.10.2 and therefore sends the echo reply to the latter (*outside local*).

The **add-route** option in the **ip nat outside source static** command instructs IOS to add a static route to the *outside local* address. This is also documented in Example 9-17.

Example 9-17 *Outside Static NAT*

```

! Static mapping for the outside host 172.20.20.2
ip nat outside source static 172.20.20.2 10.10.10.2 add-route
!
! IOS automatically adds a host route to the translated address
IOS-FW# show ip route | begin Gateway
Gateway of last resort is not set
    172.20.0.0/24 is subnetted, 1 subnets
C       172.20.20.0 is directly connected, FastEthernet4.200
    10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S       10.10.10.2/32 [1/0] via 172.20.20.2
C       10.10.10.0/24 is directly connected, FastEthernet4.201
!
! Outbound ping from host 10.10.10.140 to 10.10.10.2 (translated address)
%IPNAT-6-CREATED: icmp 10.10.10.140:34 10.10.10.140:34 10.10.10.2:34
172.20.20.2:34
NAT: s=10.10.10.140, d=10.10.10.2->172.20.20.2 [73]
NAT*: s=172.20.20.2->10.10.10.2, d=10.10.10.140 [13103]
!
IOS-FW# show ip nat translations
Pro Inside global      Inside local      Outside local      Outside
global
-- --
172.20.20.2
icmp 10.10.10.140:34    10.10.10.140:34    10.10.10.2:34
172.20.20.2:34
!
! Inbound ping from host 172.20.20.2 to 10.10.10.190
ICMP echo request from 172.20.20.2 to 10.10.10.190 ID=4388 seq=7724 len=472
ICMP echo reply from 10.10.10.190 to 172.20.20.2 ID=4388 seq=7724 len=472
!
%IPNAT-6-CREATED: icmp 10.10.10.190:4388 10.10.10.190:4388
10.10.10.2:4388 172.20.20.2:4388
NAT: s=172.20.20.2->10.10.10.2, d=10.10.10.190 [27731]
NAT: s=10.10.10.190, d=10.10.10.2->172.20.20.2 [27731]
!
IOS-FW# show ip nat translations
Pro Inside global      Inside local      Outside local      Outside
global
-- --
172.20.20.2
icmp 10.10.10.190:4388    10.10.10.190:4388    10.10.10.2:4388
172.20.20.2:4388

```

Dynamic NAT

Dynamic NAT is a technique that maps a group of inside source addresses (*local*) to a pool of translated source addresses. In most cases, IOS randomly selects an address from the global pool and associates it with the connection initiator address on the inside. After the translation timeout expires, this mapping is removed from the translation table, and there is no guarantee that the original inside host will be assigned the same global address (if a new outbound connection attempt).

Example 9-18 illustrates dynamic NAT for IOS. The relevant components follow:

- A *standard* access-list to determine the inside local source addresses that must be translated. The **permit** entries in this ACL define interesting traffic for NAT.
- An address pool (**ip nat pool**) to allocate inside global addresses (virtual addresses visible on the outside domain).
- An **ip nat inside source list** statement to create a binding between inside local and inside global addresses.

In this example, the inside global address 172.20.20.129 (belonging to the pool named OUT1) is dynamically allocated to the inside local host 10.10.10.190.

Example 9-18 Dynamic NAT

```
! Pool 'OUT1' is reserved for inside local addresses defined by ACL 10
access-list 10 permit 10.10.10.0 0.0.0.255
ip nat pool OUT1 172.20.20.129 172.20.20.254 prefix-length 25
ip nat inside source list 10 pool OUT1
!
! Translation created after outbound ping from 10.10.10.190 to 172.20.20.2
%IPNAT-6-CREATED: ? 10.10.10.190:0 172.20.20.129:0 0.0.0.0:0 0.0.0.0:0
%IPNAT-6-CREATED: icmp 10.10.10.190:25 172.20.20.129:25 172.20.20.2:25
172.20.20.2:25
NAT*: s=10.10.10.190->172.20.20.129, d=172.20.20.2 [60]
NAT*: s=172.20.20.2, d=172.20.20.129->10.10.10.190 [485]
!
IOS-FW# show ip nat translations
Pro Inside global      Inside local      Outside local      Outside
global
icmp 172.20.20.129:25  10.10.10.190:25  172.20.20.2:25
172.20.20.2:25
-- 172.20.20.129      10.10.10.190      --                --
```

Tip The default behavior of IOS, when using Dynamic NAT, is to randomly select an address from the global pool and associate it with the host address that needs translation.

The **match host** parameter for the **ip nat pool** command enables the host portion of the IP address to be preserved between the local and global pools. This special option is presented in Example 9-21.

Note IOS supports dynamic PAT using the **overload** option after the pool definition within the **ip nat inside source list** command.

Policy NAT

Now that static and dynamic NAT concepts have been revisited, it is time to devote some attention to a more sophisticated translation technique known as *Policy NAT*.

Policy NAT is a sort of *conditional translation* because it takes into consideration both the source and destination addresses when deciding which virtual address to associate with a given real address. The translation criteria are defined by ACLs.

Example 9-19 illustrates dynamic Policy NAT and dynamic Policy PAT (*Port Address Translation*) for IOS. Some aspects that deserve specific mention in this scenario follow:

- An *extended* ACL (rather than the standard ACL used for regular dynamic NAT) is necessary so that source and destination addresses are simultaneously analyzed in the NAT decision process.
- In the sample dynamic Policy NAT configuration, ACL 101 establishes that global addresses in the pool OUT1 should be assigned when inside hosts on subnet 10.10.10.0/24 attempt connection with the outside address 172.20.20.2.
- In a similar fashion, ACL 102 defines that hosts on the 10.10.10.0/24 subnet should receive an address from the **global** pool OUT2, when communicating with the outside address 172.20.20.20. The **overload** option in the **ip nat inside source list** command is what tells IOS to perform Dynamic Policy PAT (instead of Dynamic Policy NAT).

Example 9-19 Dynamic Policy NAT/PAT

```
! Dynamic Policy NAT
access-list 101 permit ip 10.10.10.0 0.0.0.255 host 172.20.20.2
ip nat pool OUT1 172.20.20.129 172.20.20.254 prefix-length 25
ip nat inside source list 101 pool OUT1
!

! Dynamic Policy PAT
access-list 102 permit ip 10.10.10.0 0.0.0.255 host 172.20.20.20
ip nat pool OUT2 172.20.20.125 172.20.20.125 prefix-length 30
ip nat inside source list 102 pool OUT2 overload
!

! Ping from 10.10.10.190 to 172.20.20.2 (dynamic Policy NAT)
```

```
%IPNAT-6-CREATED: ? 10.10.10.190:0 172.20.20.129:0 0.0.0.0:0 0.0.0.0:0
%IPNAT-6-CREATED: icmp 10.10.10.190:31 172.20.20.129:31 172.20.20.2:31
172.20.20.2:31
NAT*: s=10.10.10.190->172.20.20.129, d=172.20.20.2 [66]
NAT*: s=172.20.20.2, d=172.20.20.129->10.10.10.190 [20375]
!
! Ping from 10.10.10.190 to 172.20.20.20 (dynamic Policy PAT)
%IPNAT-6-CREATED: icmp 10.10.10.190:30 172.20.20.125:30 172.20.20.20:30
172.20.20.20:30
NAT*: s=10.10.10.190->172.20.20.125, d=172.20.20.20 [65]
NAT*: s=172.20.20.20, d=172.20.20.125->10.10.10.190 [199]
```

Dual NAT

The concurrent use of the `ip nat inside source static` and `ip nat outside source static` commands, in a given IOS configuration, is a possible way to produce the *Dual NAT* behavior (translating both source and destination addresses when interconnecting the local and global address spaces). Example 9-20 brings a sample configuration of this category of NAT:

- The inside host 10.10.10.200 is published as 172.20.20.200 on the outside.
- The outside host 172.20.20.2 is visible as 10.10.10.2 on the inside.

The following sequence of events is associated with Example 9-20:

- Step 1.** The inside local host 10.10.10.200 pings the outside local address 10.10.10.2 (which appears to live on the same subnet). The source address 10.10.10.200 is translated to 172.20.20.200 (inside global).
- Step 2.** IOS untranslates the destination address 10.10.10.2 before delivering the ICMP echo request packet to the outside global host (real address 172.20.20.2).
- Step 3.** The real host 172.20.20.2 replies to the echo request as if it had been originated by the 172.20.20.200 address (inside global).
- Step 4.** IOS translates the source address 172.20.20.2 (of the echo reply packet) to 10.10.10.2 before delivery to the inside local host 10.10.10.200.

The operation of Dual NAT for an outbound connection is generalized in Figure 9-7, which depicts an *inside local* host C1 being mapped to the *inside global* address K1. The scenario is complemented by the binding between the *outside global* address S1 to its counterpart Z1 (*outside local*).

Example 9-20 also documents the opposite situation for an inbound ping (starting on 172.20.20.2 and destined to 172.20.20.200). The explanation of this scenario is proposed as an exercise to the reader.

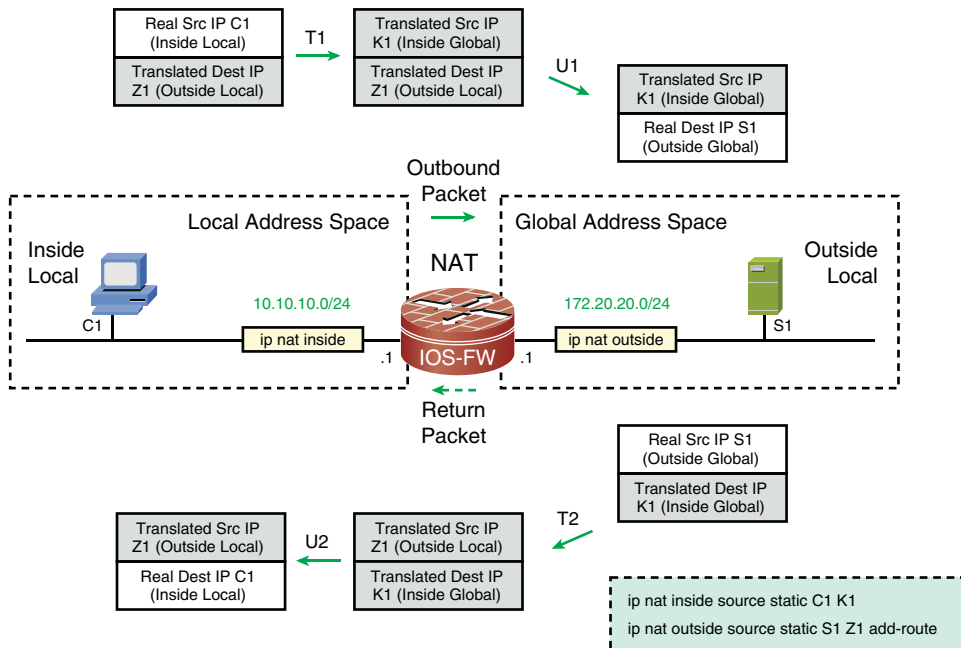


Figure 9-7 Sample Dual-NAT Topology

Example 9-20 Dual NAT

```
! One inside static and one outside static NAT statements
ip nat inside source static 10.10.10.200 172.20.20.200
ip nat outside source static 172.20.20.2 10.10.10.2 add-route
!

! Outbound ping from 10.10.10.200 to the translated address 10.10.10.2
%IPNAT-6-CREATED: icmp 10.10.10.200:35 172.20.20.200:35 10.10.10.2:35
172.20.20.2:35
NAT: s=10.10.10.200->172.20.20.200, d=10.10.10.2 [74]
NAT: s=172.20.20.200, d=10.10.10.2->172.20.20.2 [74]
NAT*: s=172.20.20.2->10.10.10.2, d=172.20.20.200 [6690]
NAT*: s=10.10.10.2, d=172.20.20.200->10.10.10.200 [6690]
!
IOS-FW# show ip nat translations | include icmp
Pro Inside global      Inside local      Outside local      Outside
global
icmp 172.20.20.200:35  10.10.10.200:35   10.10.10.2:35
172.20.20.2:35
!

! Inbound Ping from 172.20.20.2 to the translated address 172.20.20.200
%IPNAT-6-CREATED: icmp 10.10.10.200:4388 172.20.20.200:4388 10.10.10.2:4388
172.20.20.2:4388
```

```

NAT*: s=172.20.20.2->10.10.10.2, d=172.20.20.200 [18680]
NAT*: s=10.10.10.2, d=172.20.20.200->10.10.10.200 [18680]
NAT: s=10.10.10.200->172.20.20.200, d=10.10.10.2 [18680]
NAT: s=172.20.20.200, d=10.10.10.2->172.20.20.2 [18680]
!
! More detailed view of the translation table
IOS-FW# show ip nat translations verbose
Pro Inside global          Inside local          Outside local          Outside
global
- - - - -                - - - - -                10.10.10.2
172.20.20.2
    create 00:23:56, use 00:00:15 timeout:0,
    flags:static, outside, route, use_count: 1, entry-id: 70, lc_entries: 0
icmp 172.20.20.200:4388    10.10.10.200:4388      10.10.10.2:4388
172.20.20.2:4388
    create 00:00:15, use 00:00:15 timeout:60000, left 00:00:44,
    flags:extended, outside, use_count: 0, entry-id: 77, lc_entries: 0
- - 172.20.20.200          10.10.10.200          - - - - -
    create 00:07:49, use 00:00:15 timeout:0,
    flags:static, use_count: 1, entry-id: 75, lc_entries: 0

```

NAT and Flow Accounting

Chapter 4, “Learn the Tools. Know the Firewall,” not only introduces Netflow as a *flow accounting* resource but also proposes some other possible usages for this tool. The current section employs this important instrumentation to characterize how flows are counted in an environment in which NAT is enabled. (This study naturally reflects the order in which translation and accounting operations are performed by IOS).

In Example 9-21, Netflow *ingress accounting* is configured for interface F4.200 (which plays the role of outside for the network diagram in Figure 9-2). Two tests are explored in this example:

- **Outbound pings from hosts on the 10.10.10.0/24 subnet to 172.20.20.100:** The inside local hosts are dynamically assigned an IP address that belongs to pool OUT1. The outside global address sends echo replies as if the requests had been originated by hosts on the 172.20.20.0/24 subnet. As a result, the flows are counted on the outside interface (F4.200) as going from 172.20.20.100 to 172.20.20.X, meaning that Netflow acts before the *untranslation* process takes place (inside global > inside local).
- **Outbound pings from hosts on the 10.10.10.0/24 subnet to the outside local address 10.10.10.2 (translated address):** The inside local hosts are dynamically assigned an IP address selected from pool OUT1. In this case, the destination address 10.10.10.2 is *untranslated* to 172.20.20.2, before delivery of the ICMP echo requests. 172.20.20.2 then replies to these packets as if they have been sent by hosts on its local subnet, 172.20.20.0/24. Given that Netflow is enabled only on interface F4.200, flows are counted for as sent by 172.20.20.2 and destined to 172.20.20.X.

Example 9-21 NAT and Ingress Flow Accounting

```
! Dynamic inside NAT and static outside NAT definitions
access-list 10 permit 10.10.10.0 0.0.0.255
ip nat pool OUT1 172.20.20.129 172.20.20.254 prefix-length 25 type match-host
ip nat inside source list 10 pool OUT1
!
ip nat outside source static 172.20.20.2 10.10.10.2 add-route
!
interface FastEthernet4.201
 ip address 10.10.10.1 255.255.255.0
 ip nat inside
!
! Adding ingress flow accounting to the outside interface
interface FastEthernet4.200
 ip address 172.20.20.1 255.255.255.0
 ip flow ingress
 ip nat outside
!
! Outbound pings to the real address 172.20.20.100
INSIDE# ping 172.20.20.100 source 10.10.10.190 repeat 12
!
IOS-FW# show ip cache flow | begin Src
SrcIf      SrcIPAddress  DstIf      DstIPAddress  Pr SrcP DstP  Pkts
Fa4.200    172.20.20.100    Fa4.201    172.20.20.190 01 0000 0000 12
!
! Outbound pings to the translated address 10.10.10.2
INSIDE# ping 10.10.10.2 source 10.10.10.200 repeat 30
!
IOS-FW# show ip cache flow | begin Src
SrcIf      SrcIPAddress  DstIf      DstIPAddress  Pr SrcP DstP  Pkts
Fa4.200    172.20.20.2    Fa4.201    172.20.20.200 01 0000 0000 30
```

Example 9-22 extends 9-21 by adding Netflow *egress accounting* to the inside interface, F4.201 (refer to Figure 9-2). The following tests are performed:

- **Outbound pings to the real host 172.20.20.100:** Because this destination address is not translated, it always appears as *SrcIPAddress* in the flow accounting (either ingress or egress). The *DstIPAddress* registered in the flows, on the other hand, depends on the interface where Netflow is acting. Although it appears as the inside global 172.20.20.190 in F4.200, it is seen as 10.10.10.190 (inside local) for F4.201. (As a consequence of egress accounting happening after the untranslation inside global > inside local).
- **Outbound pings to the translated address 10.10.10.2:** This test underlines that the ingress accounting process (enabled on the outside) sees the flows as going from the

outside global (172.20.20.2) to the inside global address (172.20.20.190). On the contrary, the flows are visible with source 10.10.10.2 (outside local) and destination 10.10.10.190 (inside local) from the perspective of the egress accounting on Fa4.201

Example 9-22 NAT and Egress Flow Accounting

```
! Adding egress flow accounting to the inside interface
interface FastEthernet4.201
 ip address 10.10.10.1 255.255.255.0
 ip flow egress
 ip nat inside
!
```

! Outbound pings to the real address 172.20.20.100 (from 10.10.10.190)

```
INSIDE# ping 172.20.20.100 source 10.10.10.190 repeat 28
!
```

SrcIf	SrcIPaddress	DstIf	DstIPaddress	Pr	SrcP	DstP	Pkts
Fa4.200	172.20.20.100	Fa4.201	172.20.20.190	01	0000	0000	28
Fa4.200	172.20.20.100	Fa4.201*	10.10.10.190	01	0000	0000	28

! Outbound pings to the translated address 10.10.10.2

```
INSIDE# ping 10.10.10.2 source 10.10.10.190 repeat 15
!
```

SrcIf	SrcIPaddress	DstIf	DstIPaddress	Pr	SrcP	DstP	Pkts
Fa4.200	172.20.20.2	Fa4.201	172.20.20.190	01	0000	0000	15
Fa4.200	10.10.10.2	Fa4.201*	10.10.10.190	01	0000	0000	15

When taken together, the two examples just presented demonstrate the following facts about *flow accounting* for inbound traffic (*outside* to *inside*):

- Ingress Netflow happens before NAT and sees the *global* addresses.
- Egress Netflow comes into play after NAT (either translation or untranslation) and is aware of local addresses.

At this point, you are strongly encouraged to study what happens with the NAT and Netflow interaction when flow accounting is configured for outbound (*inside* to *outside*).

CBAC and NAT

This chapter presented a detailed analysis of CBAC working as an isolated feature or with input ACLs. It also covered the main categories of address translation available in IOS.

Building upon this knowledge, this section proposes a typical situation in which CBAC

and ACLs are combined with NAT to provide security for an environment with more sophisticated demands. The requirements for the scenario follow:

- Two real servers living on the inside of the IOS Firewall need to have their addresses published on the outside.
- An inbound ACL should limit the types of traffic allowed for each of the published addresses.
- No outbound connection initiation is permitted. This is achieved with a restrictive ACL (implementing a *deny all* model).
- CBAC is in charge of inspecting the incoming traffic and, in the sequence, creating the pertinent openings for the return traffic.

Example 9-23 assembles the commands for a sample deployment that meets these requirements. Figure 9-8 shows the companion reference topology:

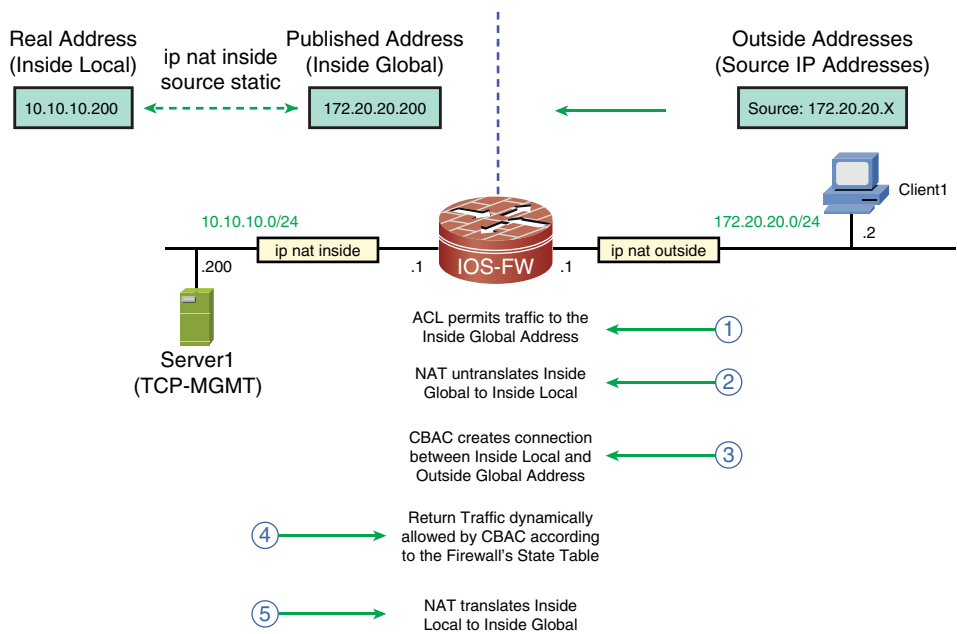


Figure 9-8 Reference Topology for Inbound CBAC with NAT and ACL

- The real servers, residing on network 10.10.10.0/24, are assigned public addresses on the 172.20.20.0/24 subnet.
- The INBOUND ACL is applied in the incoming direction for interface F4.200 (outside), with the goal to define what protocol connections can be directed to the servers (pointing to their virtual addresses). The **object-groups** contained in the ACL are those previously defined in Examples 9-11 and 9-12.

- ACL 110 is applied in the incoming direction for interface F4.201 and is responsible for blocking any outbound connection attempts.
- CBAC is defined by the inspection rules named INBOUND1 and applied to F4.200. In this case, its tasks are focused on the generic stateful inspection of TCP and ICMP.

Example 9-23 *Relevant Configurations Inbound CBAC, NAT, and ACL*

```

! Publishing the internal addresses on the outside interface
ip nat inside source static 10.10.10.190 172.20.20.190
ip nat inside source static 10.10.10.200 172.20.20.200
!
! ACL applied to the outside interface (defining allowed inbound traffic)
ip access-list extended INBOUND
 permit object-group TCP-MGMT object-group OUT1 host 172.20.20.200 log
 permit icmp any host 172.20.20.190 echo log
 deny ip any any log
!
! ACL used to block outbound connection initiation (applied on the inside inter-
face)
access-list 110 deny ip any any log
!
! Inspection Rules to be applied on the outside interface
ip inspect name INBOUND1 tcp alert on router-traffic
ip inspect name INBOUND1 icmp alert on router-traffic
!
! Outside interface ( where connection requests arrive)
interface FastEthernet4.200
 ip address 172.20.20.1 255.255.255.0
 ip access-group INBOUND in
 ip nat outside
 ip inspect INBOUND1 in
!
! Inside interface (return traffic)
interface FastEthernet4.201
 ip address 10.10.10.1 255.255.255.0
 ip access-group 110 in
 ip nat inside

```

Example 9-24 relates to Example 9-23 and shows TCP Inspection at work for a Telnet session from 172.20.20.50 (outside global) to the inside global address of Server1 (172.20.20.200). The sequence of operations is further documented in Figure 9-8:

- Step 1.** The Telnet traffic is permitted by the access-list called INBOUND.
- Step 2.** Upon receipt of the traffic destined to 172.20.20.200 (inside global), IOS untranslates this address to 10.10.10.200 (inside local).

- Step 3.** CBAC inspects the post-NAT traffic and treats 10.10.10.200 as the responder address (*responder_addr*). CBAC is completely aware of the entire NAT process. This is promptly spotted by the existence of the alternative addresses (*initiator_alt_addr* and *responder_alt_addr*) in the **debug ip inspect object-creation** output. Further, the **show ip inspect sessions** command displays a connection between the real addresses involved.
- Step 4.** The temporary opening created by CBAC on ACL 110 permits the return traffic from the real server (10.10.10.200).
- Step 5.** IOS translates the inside local address (10.10.10.200) back to the inside global (172.20.20.200).

Example 9-24 Inbound CBAC, NAT, and ACL for TCP

```

! ACL permits Telnet from 172.20.20.50 to the translated address 172.20.20.200
%SEC-6-IPACCESSLOGP: list INBOUND permitted tcp 172.20.20.50(1592) ->
172.20.20.200(23), 1 packet
!
! IOS untranslates the published address (inside global > inside local)
%IPNAT-6-CREATED: tcp 10.10.10.200:23 172.20.20.200:23 172.20.20.50:1592
172.20.20.50:1592
!
! CBAC inspects incoming connection and creates opening for return traffic
FIREWALL OBJ_CREATE: Pak 839CEB04 sis 844C12B0
initiator_addr (172.20.20.50:1592) responder_addr (10.10.10.200:23)
initiator_alt_addr (172.20.20.50:1592) responder_alt_addr (172.20.20.200:23)
FIREWALL OBJ-CREATE: sid 847AA494 acl 110 Prot: tcp
  Src 10.10.10.200 Port [23:23]
  Dst 172.20.20.50 Port [1592:1592]
!
IOS-FW# show ip inspect sessions
Established Sessions
  Session 844C12B0 (172.20.20.50:1592)=>(10.10.10.200:23) tcp SIS_OPEN
!
IOS-FW# show ip nat translations | include tcp
Pro Inside global      Inside local      Outside local      Outside
global
tcp 172.20.20.200:23    10.10.10.200:23    172.20.20.50:1592
172.20.20.50:1592

```

Example 9-25 is completely analogous to its antecedent. The main distinction is that it deals with ICMP rather than TCP inspection. The point that deserves special notice here is that the first echo request creates the CBAC flow just after *untranslation*. The **debug ip nat** output makes it clear that the echo reply is translated back by IOS after the CBAC process. Subsequent pings (within the same session) do not create any new flow because the opening is already in place and valid for a certain amount of time.

Example 9-25 *Inbound CBAC, NAT, and ACL for ICMP*

```

! Ping from 172.20.20.2 to the translated address 172.20.20.190
ASA2# ping 172.20.20.190 repeat 2
Success rate is 100 percent (2/2), round-trip min/avg/max = 1/5/10 ms
!
! ACL 'INBOUND' permits traffic to the published address 172.20.20.190
%SEC-6-IPACCESSLOGDP: list INBOUND permitted icmp 172.20.20.2 -> 172.20.20.190
(8/0),
  2 packets
!
! The first packet is untranslated (inside global > inside local)
%IPNAT-6-CREATED: icmp 10.10.10.190:4388 172.20.20.190:4388 172.20.20.2:4388
172.20.20.2:4388
NAT: s=172.20.20.2, d=172.20.20.190->10.10.10.190 [8376]
!
! CBAC inspects incoming connection and creates opening for return traffic
FIREWALL OBJ_CREATE: Pak 84025B50 sis 844C12B0
initiator_addr (172.20.20.2:8) responder_addr (10.10.10.190:0)
initiator_alt_addr (172.20.20.2:8) responder_alt_addr (172.20.20.190:0)
FIREWALL icmp_info created: 0x84137A80
FIREWALL OBJ_CREATE: sid 847AA53C acl 110 Prot: icmp
  Src 10.10.10.190 Port [0:0]
  Dst 172.20.20.2 Port [0:0]
!
! Translation for the first packet (Inside Local > Inside Global)
NAT*: s=10.10.10.190->172.20.20.190, d=172.20.20.2 [8376]
!
! Untranslation and Translation for the subsequent packets in the flow
NAT: s=172.20.20.2, d=172.20.20.190->10.10.10.190 [3613]
NAT*: s=10.10.10.190->172.20.20.190, d=172.20.20.2 [3613]

```

Summary

This chapter described how an IOS router, historically dedicated to the routing and forwarding functions, may be configured to operate as a true stateful firewall device. The chapter dealt specifically with the generic stateful inspection of TCP, UDP, and ICMP, using the *Classic IOS Firewall* feature set, a solution originally named CBAC.

The chapter also presented some other important topics:

- How to establish connection limits on a per-host basis and mitigate DoS attacks.
- The handling of ACLs within IOS, which use a *wildcard mask* instead of the *network mask*, employed by ASA. This approach makes the definition of source and destination addresses on IOS ACLs more flexible than on their ASA counterparts.
- The handling of object-groups within IOS and how they can produce more structured ACLs. It is important to get acquainted with these resources because they will be frequently necessary going forward.
- How ACLs complement CBAC, by filtering the traffic types allowed through before inspection starts.
- A quick review of the main NAT resources available in IOS.
- More complex practical usage scenarios in which CBAC, NAT, and ACLs are used simultaneously to deliver the desired network and security functionalities.

Drawing on the knowledge acquired within the current chapter, Chapter 10 covers the *IOS Zone Based Policy Firewall*, which represents a new approach for stateful inspection for Cisco IOS software.

IOS Zone Policy Firewall Overview

This chapter covers the following topics:

- Motivations for the Zone-based Policy Firewall
- Building blocks for Zone-based Firewall policies
- ICMP connection examples
- UDP connection examples
- TCP connection examples
- ZFW and ACLs
- ZFW and NAT
- ZFW in Transparent mode
- Defining connection limits
- Inspection of router traffic
- Intrazone firewall policies in IOS 15.X

“Every gain made by individuals or society is almost instantly taken for granted.”
—Aldous Huxley

Chapter 9, “Classic IOS Firewall Overview,” devotes a great deal of attention to the analysis of *Context Based Access Control (CBAC)*, the original implementation of *stateful inspection* functionality in Cisco IOS.

This chapter builds on the knowledge acquired in Chapter 9 to explore a new IOS methodology to deal with stateful inspection, known as the *Zone-based Policy Firewall* or *Zone Policy Firewall (ZFW or ZPF)*. Remember that the development efforts of new IOS Firewall features are directed to the ZFW.

In this chapter, ZFW focuses on generic Layer 4 inspection (basically UDP and TCP and, by relaxing some rigorous OSI definitions, also ICMP). Chapter 12, “Application Inspection,” extends the study of its capabilities to the specific handling of some application protocols.

IOS resources such as Network Address Translation (NAT), Access Control Lists (ACL), **object-groups** and Netflow, already covered in Chapter 9, are revisited in some of the sections so that their interactions with the ZFW can be analyzed.

Concentrate not only on understanding how zone-based concepts affect the normal behavior of routers, but also on the modularity philosophy inherent to ZFW.

Motivations for the ZFW

The development of Chapter 9 was centered on the usage of the **ip inspect** family of commands to implement stateful firewall functionality. More precisely, it has shown the following:

- Each **ip inspect name** statement instructs IOS to watch connection initiation requests for a particular (L4 or L7) protocol that arrives on a given IOS interface. After processing these requests, CBAC decides about the need of inserting an entry in its state table and creates the associated temporary opening for return traffic.
- CBAC inspection on an interface takes place after input ACL processing. This ACL limits the combination of source/destination addresses allowed to initiate connections through the firewall.
- Any packet for a protocol not explicitly defined to be inspected by CBAC is enabled to pass through the firewall, but it probably face problems with the return traffic. (Likely to be blocked by interface ACLs because the temporary openings were not created.)
- CBAC relies on an interface-based model for firewall policies building. For any input interface where **ip inspect** is configured, there should be a worry with all the possible output interfaces (and the corresponding ACLs) for these packets.
- Although dedicated firewalls (such as ASA) are intrinsically closed, a router is regarded as a connectivity provider, and therefore, normally, does not impose restrictions by means of any inherent packet filters. CBAC is employed to transform the router into a true stateful firewall but always does that on a per-interface basis. (There is no provision for globally closing the router at once).

The ZFW modifies IOS stateful inspection operations by introducing the concept of *security zones*, which enables easier definition of the degree of trustworthiness of a given interface. This new approach also brings more scalability and flexibility to security design activities. The basic aspects of ZFW behavior to be aware of follow:

- Router interfaces are placed in *security zones*, and inspection is applied to packets crossing the firewall between two given *zones*.

- One interface residing on a certain security zone is forbidden from passing packets to interfaces that are members of different zones, unless an *inter-zone policy* is explicitly defined. Further, no traffic is enabled to flow between zone and *nonzone* interfaces. These two rules result in a default blocking between zones, suggesting a simpler way to *close* the router than that provided by CBAC.
- An interzone-policy requires the definition of the source and destination zones that it connects to. The policies are unidirectional in nature and, after inspecting traffic from the source zone, they handle traffic returning from the destination zone.
- An interface cannot be a member of more than one security zone.
- A security zone might include multiple router interfaces.
- The ZFW employs the Class-Based Policy Language (CPL) to build structured and hierarchical policies.
- Interface ACLs are not suitable for use with the ZFW. To add address filtering to the inspection policies, ACLs will be used inside **class-maps**, rather than in the form of **ip access-groups** applied to router interfaces. This paradigm shift from CBAC is so critical for ZFW operation, that it will be dedicated a specific section (ZFW and ACLs).

Note Two router interfaces that are not members of any security zone are allowed to initiate connections to each other.

Figure 10-1 was conceived to emphasize some of the concepts just discussed. Scenario 1 corresponds to the well-known *Internet access* firewall topology:

- A client that reaches the router through Interface Int1 (a member of the TRUSTED zone) is allowed to initiate connections to the UNTRUSTED zone and, for instance, to access a server on the Internet. The ZFW inspects packets arriving on Int1 and takes care of traffic returning through any interface that belongs to the UNTRUSTED zone.
- The OUTBOUND policy is *unidirectional* and does not permit any connection initiation from the UNTRUSTED zone.
- The Interface Int3 is not assigned to any zone and, as such, cannot connect to any of the zones shown in Figure 10-1. Conversely, no client belonging to a zone can send packets to this nonzone interface.

Scenario 2, portrayed in Figure 10-1, represents a situation in which the *Internet access* and *Internet presence* firewall topologies are implemented simultaneously:

- Policy OUTBOUND1 is aimed to inspect packets traveling from the INSIDE to the INTERNET zone. Client1, for instance, may connect to Server2.
- Policy INBOUND1 is designed to handle packets coming from the INTERNET zone and destined to the DMZ zone. Server1, for example, might be in charge of a service needed by Client2.

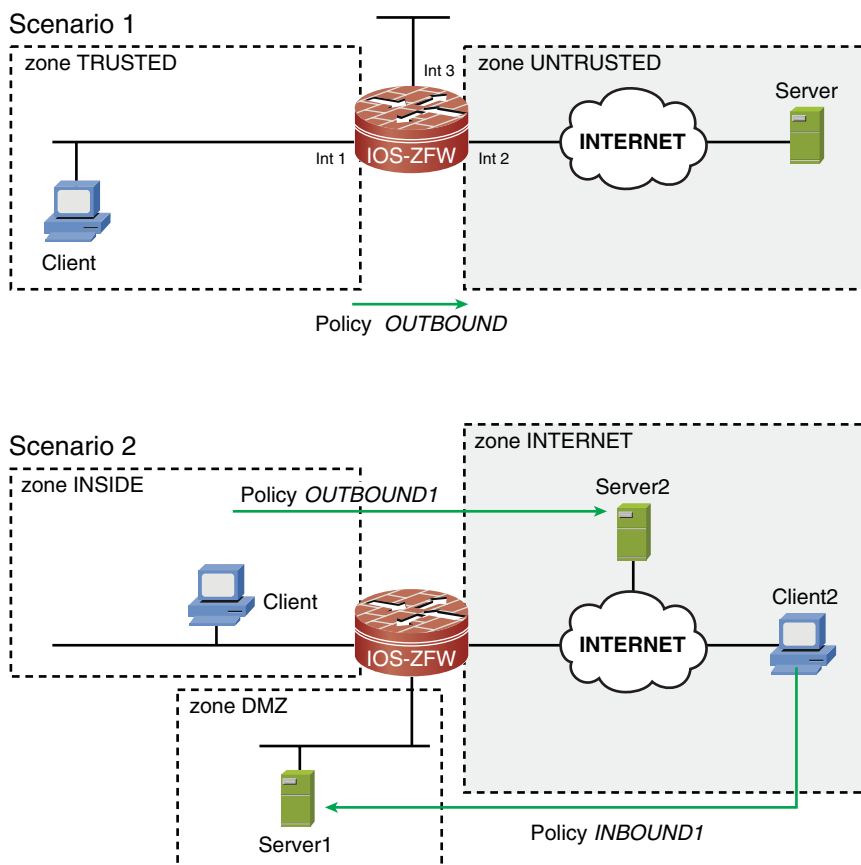


Figure 10-1 Sample Scenarios Using the Zone-Based Policy Firewall

- Clients on the INSIDE zone are not allowed to access the DMZ, for there is no inter-zone policy connecting INSIDE and DMZ.

Now that you know that zone-based policies are *unidirectional* and that classic interface ACLs are not the ideal companion for the ZFW, the next section presents the main elements used for hierarchical firewall policy construction.

Note A CBAC rule (`ip inspect`) must not be configured for an interface belonging to a security zone. The `ip inspect` commands might still be used for interfaces not part of security zones. Nonetheless, this configuration is not recommended.

Building Blocks for Zone-Based Firewall Policies

This section introduces **class-maps** and **policy-maps** (the major components of a ZFW policy) and their relevant attributes. Subsequent sections freely employ these concepts to build policies of practical application for *generic inspection* demands. If you are already familiar with QoS configuration, you can quickly detect the similarities between the Class-based Policy Language (CPL) and the Modular QoS CLI (MQC). ZFW uses **type inspect class-maps** and **policy-maps** (instead of the regular maps used within the QoS domain).

A **class-map type inspect** can identify sets of packets that share some property. Two examples are the group of FTP packets (classified by means of a **match protocol** statement) and packets that travel from Network A to Network B (classified with the aid of an ACL bound to a **match access-group** statement).

A **policy-map type inspect** is employed to assign *actions* for classes previously defined with **class-map type inspect** commands. The actions for packets that meet the selection criteria for a certain class are **inspect**, **pass**, **police**, and **drop**. A brief analysis of each of these actions follows:

- **inspect**: Detects connection requests and provides the appropriate openings for return traffic. (This is similar to the **ip inspect** within the context of CBAC.)
- **pass**: This unidirectional action is equivalent to a permit statement in an interface ACL and cannot handle return traffic.
- **drop**: Corresponding to the ACL **deny** statements, this action blocks undesired traffic trying to cross the firewall.
- **police**: Designed to provide **rate-limit** functionality for classified traffic subject to the **inspect** action. (Not available for the reserved class called *class-default*).

Note The **log** option might be used with the **pass** and **drop** actions to register the packets that have been, respectively, permitted or denied.

Figure 10-2 shows not only the main building blocks for the construction of zone-based firewall policies but also a typical sequence of configuration tasks used to activate them:

- Two zones named ZONE1 and ZONE2 are created with **zone security** commands. A security zone must be defined before an interface can be made a member of it.
- After a **class-map type inspect** called CLASS1 is created, it requires the choice of a **match** criterion. (This can be a **match protocol**, **match access-group**, or **match class-map** statement.)
- The packets selected with the **class-map** CLASS1 are subject to an action in the **policy-map type inspect** called GENERIC1. Likewise, other actions are assigned to

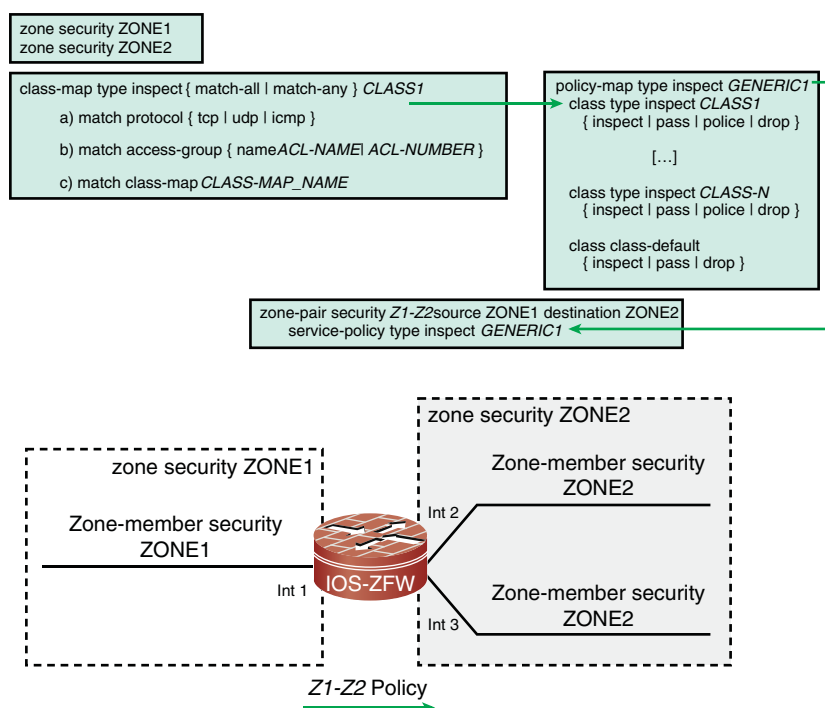


Figure 10-2 Main Components of a Zone-Based Firewall Policy

each class referred to by this **policy-map**. This sequential processing goes on until the reserved class called *class-default* is reached.

- An interzone policy called Z1-Z2 interconnects the source zone ZONE1 to the destination zone ZONE2. Z1-Z2 calls the **policy-map** GENERIC1 with a **service-policy type inspect** statement. All the actions associated with the classes belonging to GENERIC1 will be applied for traffic flowing from interfaces on ZONE1 to interfaces on ZONE2.
- Individual interfaces become part of security zones when a **zone-member security** command is issued under interface configuration mode.

After performing the tasks just described, a packet arriving on ZONE1 and destined to ZONE2 will be treated according to what is established by the interzone policy Z1-Z2.

Examples 10-1 through 10-3 assemble the configuration commands to build an interzone policy for traffic flowing from zone INSIDE to zone OUTSIDE in the topology shown in Figure 10-3. Example 10-4 simply teaches how to visualize zone-related information.

Example 10-1 focuses on defining *security zones* and later establishing zone membership for router interfaces (accomplished with **zone-member security** commands).

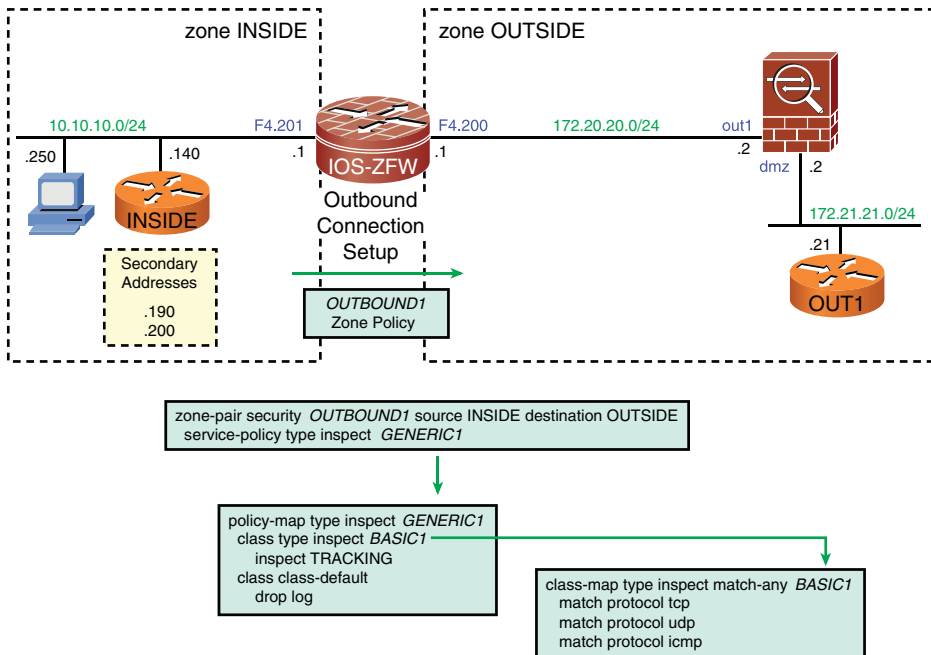


Figure 10-3 Main Topology for ICMP, UDP, and TCP Outbound Connection Examples

Example 10-2 introduces the concept of **parameter-map**, a mechanism used with the **inspect** action to modify some of its characteristics. This first example shows the settings of the default **parameter-map** and turns the **audit-trail** option on, with the goal of logging connection start and termination.

In Example 10-3, the **match-any** option for the **class-map** BASIC1 states that a **match** in any of the defined protocols is enough. The policy-map **GENERIC1** prescribes an **inspect** action, in accordance with the **parameter-map** TRACKING, for packets that fall under the class BASIC1. As seen on Example 10-2, this **parameter-map** simply sets the **audit-trail** option to **on**. The service-policy **GENERIC1** is then used to define the inter-zone policy named **OUTBOUND1**.

Example 10-1 Reference Configuration of Zones

```

! Defining two Zones
zone security INSIDE
zone security OUTSIDE
!

! Assigning Interfaces to Zones
interface FastEthernet4.200
encapsulation dot1Q 200
  
```

```

ip address 172.20.20.1 255.255.255.0
zone-member security OUTSIDE
!
interface FastEthernet4.201
encapsulation dot1Q 201
ip address 10.10.10.1 255.255.255.0
zone-member security INSIDE
!
! Instructing ZFW to log dropped packets (pre-15 IOS releases)
ip inspect log drop-pkt

```

Example 10-2 Basic Parameter-maps

```

! Displaying the default parameter-map settings
IOS-ZFW# show parameter-map type inspect default
audit-trail off
alert on
max-incomplete low unlimited
max-incomplete high unlimited
one-minute low unlimited
one-minute high unlimited
udp idle-time 30
icmp idle-time 10
dns-timeout 5
tcp idle-time 3600
tcp finwait-time 5
tcp synwait-time 30
tcp max-incomplete host unlimited block-time 0
sessions maximum 2147483647
!
! Defining parameters that pertain to the 'inspect' action
parameter-map type inspect TRACKING
audit-trail on
!
! Audit-trail setting changed to 'on' for the parameter-map called TRACKING
IOS-ZFW# show parameter-map type inspect TRACKING | include audit
audit-trail on

```

Example 10-3 Building Blocks for Zone Configuration

```

! Classifying packets with a type 'inspect' class-map
class-map type inspect match-any BASIC1
match protocol tcp
match protocol udp

```

```

match protocol icmp
!
! Defining a type inspect policy-map
policy-map type inspect GENERIC1
class type inspect BASIC1
    inspect TRACKING
class class-default
    drop log
!
! Defining a Zone Policy for traffic going from the INSIDE to the OUTSIDE zone
zone-pair security OUTBOUND1 source INSIDE destination OUTSIDE
    service-policy type inspect GENERIC1

```

Example 10-4 *Obtaining Information about Zone Configuration*

```

! Viewing existent Zone-pair Security Policies
IOS-ZFW# show zone-pair security
Zone-pair name OUTBOUND1
    Source-Zone INSIDE Destination-Zone OUTSIDE
    service-policy GENERIC1
!
! Displaying information about a specific Policy-map
IOS-ZFW# show policy-map type inspect GENERIC1
Policy Map type inspect GENERIC1
    Class BASIC1
        Inspect TRACKING
    Class class-default
        Drop log
!
! Viewing the settings for a given Class-map (used within a Policy-map)
IOS-ZFW# show class-map type inspect BASIC1
Class Map type inspect match-any BASIC1 (id 2)
    Match protocol tcp
    Match protocol udp
    Match protocol icmp

```

Note This chapter deals only with the so called **top-level class-maps**, which are enough for Layer 3 and Layer 4 traffic classification (but not for selection of packets at Layer 7). Chapter 12 covers this second task, accomplished by application-specific class-maps and policy-maps.

Note With the exception of the last section, “Intrazone Firewall Policies in IOS 15.X”, this chapter assumes the usage of a 12.4.22T IOS release (or higher, in the 12.4T train).

ICMP Connection Examples

This section uses the configurations documented in Examples 10-1 through 10-4 and the topology of Figure 10-3 to illustrate the ZFW in action for sample ICMP connections.

Example 10-5 shows an outbound ping from host 10.10.10.190 to 172.20.20.2. The **debug policy-firewall obj-creation** is useful to register session creation and the addresses involved. The example also shows that the **AUDIT_TRAIL_START** message reveals the zone-pair/class combination that triggered connection setup and suggests some useful **show** commands for displaying information about sessions.

Example 10-5 Sample Outbound Ping

```
! Ping from 10.10.10.190 to 172.20.20.2 creates session
FIREWALL* sis 84334860: Session Created
FIREWALL* sis 84334860: Pak 83EA5870
init_addr (10.10.10.190:0) resp_addr (172.20.20.2:0)
init_alt_addr (10.10.10.190:0) resp_alt_addr (172.20.20.2:0)
FIREWALL* sis 84334860: FO cls 0x84ED0160 clsgrp 0x10000000, target 0xA0000000, FO
0x845931C0, alert = 1, audit_trail = 1, L7 = Unknown-L7, PAMID = 5
FIREWALL* sis 84334860: Allocating L7 sis extension L4 = icmp, L7 = Unknown-L7,
PAMID=5
!
! Audit-trail information : session start
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:BASIC1):Start icmp session:
initiator (10.10.10.190:0) — responder (172.20.20.2:0)
!
! Information about sessions for a zone-pair
IOS-ZFW# show policy-map type inspect zone-pair OUTBOUND1 sessions
policy exists on zp OUTBOUND1
Zone-pair: OUTBOUND1
Service-policy inspect : GENERIC1
Class-map: BASIC1 (match-any)
  Match: protocol tcp
    0 packets, 0 bytes
    30 second rate 0 bps
  Match: protocol udp
    0 packets, 0 bytes
    30 second rate 0 bps
  Match: protocol icmp
    1 packets, 780 bytes
    30 second rate 0 bps
```

```

Inspect
  Number of Established Sessions = 1
Established Sessions
  Session 84334860 (10.10.10.190:8)=>(172.20.20.2:0) icmp SIS_OPEN
    Created 00:00:03, Last heard 00:00:03
    ECHO request
    Bytes sent (initiator:responder) [2316:2316]
Class-map: class-default (match-any)
  Match: any
Drop
    0 packets, 0 bytes
!
! Filtering the output to concentrate on session information
IOS-ZFW# show policy-map type inspect zone-pair sessions | include Session
  Number of Established Sessions = 1
Established Sessions
  Session 84334860 (10.10.10.190:8)=>(172.20.20.2:0) icmp SIS_OPEN

```

In Example 10-6, flow accounting is enabled in the ingress direction for interface F4.200 and in the egress direction for F4.201. The example makes it clear that while ZFW counts sessions, Netflow registers the number of packets belonging to a given flow. For instance, a single ping session is composed of 20 packets.

Example 10-6 *Enabling Flow Accounting*

```

! Ingress flow accounting on the outside interface
interface FastEthernet4.200
  ip address 172.20.20.1 255.255.255.0
ip flow ingress
  zone-member security OUTSIDE
!
! Egress flow accounting on the inside interface
interface FastEthernet4.201
  ip address 10.10.10.1 255.255.255.0
ip flow egress
  zone-member security INSIDE
!
! Sequence of 20 pings from the inside host 10.10.10.140 to 172.20.20.2
INSIDE# ping 172.20.20.2 source 10.10.10.140 repeat 20 size 700
Success rate is 100 percent (20/20), round-trip min/avg/max = 1/2/4 ms
!
! Audit-trail registers the session creation

```

```
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:BASIC1):Start icmp session:
initiator (10.10.10.140:0) — responder (172.20.20.2:0)
!
! The 20 packets are part of the same session
IOS-ZFW# show policy-map type inspect zone-pair sessions | include Session
      Number of Established Sessions = 1
      Established Sessions
        Session 84334C60 (10.10.10.140:8)=>(172.20.20.2:0) icmp SIS_OPEN
!
! Netflow counts the packets belonging to the same flow
IOS-ZFW# show ip cache flow | begin Src
SrcIf      SrcIPAddress  DstIf      DstIPAddress  Pr SrcP DstP  Pkts
Fa4.200    172.20.20.2      Fa4.201*   10.10.10.140  01 0000 0000   20
Fa4.200    172.20.20.2      Fa4.201    10.10.10.140  01 0000 0000   20
```

Example 10-7 documents a `tracert` from the Windows host 10.10.10.250 portrayed in Figure 10-3 to 172.21.21.21 (the OUT1 router). As seen on previous chapters, this type of trace test employs solely ICMP messages and, as a result, just requires the help of ICMP inspection.

Example 10-7 *Sample Windows Traceroute Session*

```
! Tracing the route from 10.10.10.250 to 172.21.21.21 (OUT1)
C:\Documents and Settings\Alexandre> tracert 172.21.21.21
Tracing route to 172.21.21.21 over a maximum of 30 hops
  1    <1 ms    <1 ms    <1 ms    10.10.10.1
  2     2 ms     *         1 ms    172.21.21.2
  3     2 ms     1 ms     1 ms    172.21.21.21
Trace complete.
!
! The Zone Policy Firewall (first hop) sends time-exceeded messages to the
Windows host
ICMP: time exceeded (time to live) sent to 10.10.10.250 (dest was 172.20.20.100)
!
! ICMP is inspected and the session is created
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:BASIC1):Start icmp session:
initiator (10.10.10.250:0) — responder (172.21.21.21:0)
!
! Sample ICMP time-exceeded message from the intermediate host (ASA2)
FIREWALL* sis 84A35240: ICMP Echo pkt 10.10.10.250 => 172.21.21.21
FIREWALL* sis 84A35240: ICMP Time Exceeded pkt 172.21.21.2 => 10.10.10.250
!
! Echo replies from the destination host are allowed back (result of inspection)
FIREWALL* sis 84A35240: ICMP Echo pkt 10.10.10.250 => 172.21.21.21
```

```

FIREWALL* sis 84A35240: ICMP Echo Reply pkt 172.21.21.21 => 10.10.10.250
!
! The trace activity generates a single ICMP session
IOS-ZFW# show policy-map type inspect zone-pair sessions | include Session
      Number of Established Sessions = 1
      Established Sessions
        Session 84A35240 (10.10.10.250:8)=>(172.21.21.21:0) icmp SIS_OPEN
!
! Flow accounting for the tracert session
IOS-ZFW# show ip cache flow | begin Src

```

SrcIf	SrcIPaddress	DstIf	DstIPaddress	Pr	SrcP	DstP	Pkts
Fa4.200	172.21.21.2	Fa4.201*	10.10.10.250	01	0000	0B00	2
Fa4.200	172.21.21.2	Fa4.201	10.10.10.250	01	0000	0B00	2
Fa4.200	172.21.21.21	Fa4.201*	10.10.10.250	01	0000	0000	3
Fa4.200	172.21.21.21	Fa4.201	10.10.10.250	01	0000	0000	3

Note Example 10-7 assumed that all the required configurations to allow trace through ASA (and to instruct it to display itself as a hop) were in place (if needed, refer to Chapter 7, “Through ASA Without NAT”).

UDP Connection Examples

This section builds on the configurations registered in Examples 10-1 through 10-4 and on the associated topology and policy scheme of Figure 10-3 to demonstrate the operation of ZFW for UDP. IOS traceroute is chosen for this purpose because it creates a need for the **pass** action for the auxiliary ICMP messages employed for tracing (*time-exceeded* and *port-unreachable*).

In Example 10-8, an ACL called TRACEROUTE1 is defined with the goal of permitting trace-related messages from any host on 172.16.0.0/12 to those addresses on the subnets described by the INSIDE1 **object-group**. A **pass** action is configured under **policy-map** MGMT1 for packets selected by the **class-map** TRACE1, whose match criterion is based on the ACL TRACEROUTE1. The MGMT1 **policy-map** is employed by the **zone-pair** policy INBOUND1 to enable the returning ICMP messages. For easier understanding, these CPL building blocks are shown in Figure 10-4.

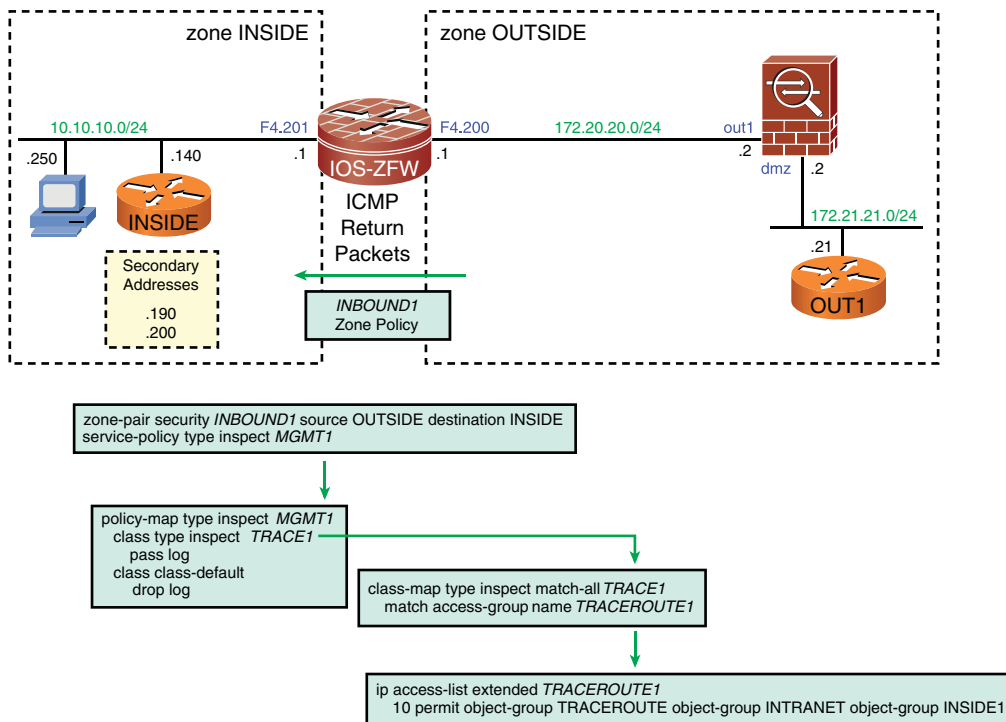


Figure 10-4 Reference Topology for the Analysis of IOS Traceroute

Example 10-8 Inbound Zone-Pair Policy to Enable IOS Traceroute

```

! Network object-groups
object-group network INTRANET
172.16.0.0 255.240.0.0
!
object-group network INSIDE1
10.10.10.0 255.255.255.0
172.20.20.0 255.255.255.0
!
! Service object-group (defining ICMP messages required for traceroute)
object-group service TRACEROUTE
icmp unreachable
icmp time-exceeded
!
! ACL allowing traceroute between some source/destination pairs
ip access-list extended TRACEROUTE1
permit object-group TRACEROUTE object-group INTRANET object-group INSIDE1
!
! Classifying packets based on a previously defined ACL
  
```

```

class-map type inspect match-all TRACE1
  match access-group name TRACEROUTE1
!
! Policy-map that uses the class-map called TRACE1 with 'pass' and 'log' actions
policy-map type inspect MGMT1
  class type inspect TRACE1
  pass log
class class-default
  drop log
!
! Defining a Zone Policy for traffic going from the OUTSIDE to the INSIDE zone
zone-pair security INBOUND1 source OUTSIDE destination INSIDE
service-policy type inspect MGMT1

```

Example 10-9 registers an IOS `traceroute` session from router 10.10.10.200 (INSIDE) to the destination address 172.21.21.21. This tracing test is made possible by the combination of two zone-based policies:

- The policy OUTBOUND1 (Example 10-3 and Figure 10-3) inspects UDP probes sent by the IOS router (INSIDE).
- The policy INBOUND1 (Example 10-8 and Figure 10-4) enables the pertinent ICMP error messages originated by L3 hops in the path to **pass** through the ZFW and reach the router called INSIDE (10.10.10.200).

Example 10-9 also shows the following:

- The first UDP probe (TTL = 1) is blocked by the IOS-ZFW router, without further inspection. Unlike ASA, the default behavior for an IOS router is to include itself as hop in a traceroute test. Notice that this happens because of the TTL value and not as a result of a Zone-based policy.
- The policy OUTBOUND1 inspects the second probe and allows it through. It is dropped by 172.21.21.2 (ASA2), which then sends a *time-exceeded* message back to 10.10.10.200. This message is allowed to pass by policy INBOUND1.
- Likewise, the *port-unreachable* message, sent by the last hop (in response to the third UDP probe), is allowed to pass through the firewall by policy INBOUND1.
- The UDP sessions are kept as *half-open*, and never reach the *established* state, because the UDP probes are dropped by the L3 hops in the path (when arriving on any of them with TTL = 1). UDP packets that are part of a traceroute task do not generate a UDP response from the end host to the origin. What the originating host sees is a set of ICMP error messages from the intermediate hops along the path.

It is convenient to emphasize that UDP probes inspected by the ZFW are totally decoupled from the ICMP messages necessary for **traceroute**. This is what renders UDP inspection insufficient for this task and justifies the use of the **pass** action.

Example 10-9 Sample IOS Traceroute Session

```

! IOS traceroute from 10.10.10.200 (INSIDE Router) to 172.21.21.21 (OUT1 Router)
INSIDE# traceroute 172.21.21.21 source 10.10.10.200 probe 1
Tracing the route to 172.21.21.21
  1 10.10.10.1 0 msec
  2 172.21.21.2 0 msec
  3 172.21.21.21 0 msec
ICMP: time exceeded rcvd from 10.10.10.1
ICMP: time exceeded rcvd from 172.21.21.2
ICMP: dst (10.10.10.200) port unreachable rcv from 172.21.21.21
!
! First UDP probe (DestPort 33434) is blocked by IOS because it has TTL =1
IOS-ZFW# ICMP: time exceeded (time to live) sent to 10.10.10.200 (dest was
172.21.21.21)
!
! Second UDP probe (DestPort 33435) creates a session
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:BASIC1):Start udp session:
initiator (10.10.10.200:49296) — responder (172.21.21.21:33435)
!
! ICMP time-exceeded message from 172.21.21.2 (ASA2) allowed to 'pass' by class
TRACE1
%FW-6-PASS_PKT: (target:class)-(INBOUND1:TRACE1) Passing icmp pkt 172.21.21.2:0 =>
10.10.10.200:0 with ip ident 0
%FW-6-LOG_SUMMARY: 1 packet were passed from 172.21.21.2:0 => 10.10.10.200:11
(target:class)-(INBOUND1:TRACE1)
!
! Third UDP probe (DestPort 33436) creates another outbound session
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:BASIC1):Start udp session:
initiator (10.10.10.200:49297) — responder (172.21.21.21:33436)
!
! ICMP port-unreachable from 172.21.21.21 (OUT1) allowed to 'pass' (class TRACE1)
%FW-6-LOG_SUMMARY: 1 packet were passed from 172.21.21.21:0 => 10.10.10.200:3
(target:class)-(INBOUND1:TRACE1)
!
! UDP sessions associated with probes (never get to the 'Established' state)
IOS-ZFW# show policy-map type inspect zone-pair sessions | include Session
Number of Half-open Sessions = 2
Half-open Sessions
  Session 84A3CC40 (10.10.10.200:49296)=>(172.21.21.21:33435) udp SIS_OPENING
  Session 84A3CE40 (10.10.10.200:49297)=>(172.21.21.21:33436) udp SIS_OPENING
!

```

! Flow accounting for the IOS traceroute session

IOS-ZFW# show ip cache flow | begin Src

SrcIf	SrcIPAddress	DstIf	DstIPAddress	Pr	SrcP	DstP	Pkts
Fa4.200	172.21.21.2	Fa4.201*	10.10.10.200	01	0000	0B00	1
Fa4.200	172.21.21.2	Fa4.201	10.10.10.200	01	0000	0B00	1
Fa4.200	172.21.21.21	Fa4.201*	10.10.10.200	01	0000	0303	1
Fa4.200	172.21.21.21	Fa4.201	10.10.10.200	01	0000	0303	1

Note The show ip cache flow command displays the ICMP type/code information in the destination port (DstP) column. For instance, 0B00 means type 11, code 0.

TCP Connection Examples

This section relies on the configurations summarized in Examples 10-1 through 10-4 and on the associated topology of Figure 10-3 to illustrate the operation of ZFW for TCP.

Example 10-10 registers a telnet session from 10.10.10.140 to 172.20.20.2 (ASA2). The example characterizes, with the help of Netflow, that many packets that belong to a flow generate a single session through the ZFW.

Example 10-11 complements Example 10-10 by revealing that the ZFW is aware of TCP Sequence and Acknowledgment numbers and the complete TCP state machine. The **debug policy-firewall protocol tcp** command is the adequate instrumentation to unveil what is going on behind the scenes.

Example 10-10 Sample Outbound Telnet Session

! Host 10.10.10.140 (INSIDE router) telnets to 172.20.20.2 (ASA2)

FIREWALL* sis 84334E60: Session Created

FIREWALL* sis 84334E60: Pak 83EA5870

init_addr (10.10.10.140:38403) resp_addr (172.20.20.2:23)

init_alt_addr (10.10.10.140:38403) resp_alt_addr (172.20.20.2:23)

FIREWALL* sis 84334E60: FO cls 0x84ED0160 clsgrp 0x10000000, target 0xA0000000, FO 0x845932C0, alert = 1, audit_trail = 1, L7 = Unknown-L7, PAMID = 5

FIREWALL* sis 84334E60: Allocating L7 sis extension L4 = tcp, L7 = Unknown-L7, PAMID= 5

%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:BASIC1):Start tcp session:

initiator (10.10.10.140:38403) — responder (172.20.20.2:23)

!

! Connection as seen by ASA (identity connection from ASA's standpoint)

%ASA-6-302013: Built inbound TCP connection 376 for dmz:10.10.10.140/38403 (10.10.10.140/38403) to identity:172.20.20.2/23 (172.20.20.2/23)

!

! Several telnet packets but just one firewall session...

```

IOS-ZFW# show ip cache flow | begin Src
SrcIf      SrcIPAddress  DstIf      DstIPAddress  Pr SrcP DstP  Pkts
Fa4.200    172.20.20.2     Fa4.201*   10.10.10.140  06 0017 9603   50
Fa4.200    172.20.20.2     Fa4.201    10.10.10.140  06 0017 9603   50
!
IOS-ZFW# show policy-map type inspect zone-pair sessions | include Session
Number of Established Sessions = 1
Established Sessions
  Session 84334E60 (10.10.10.140:38403)=>(172.20.20.2:23) tcp SIS_OPEN
!
! More information about the 'Inspect' action of the Zone Policy Firewall
IOS-ZFW# show policy-map type inspect zone-pair sessions | begin Inspect
Inspect
  Number of Established Sessions = 1
  Established Sessions
    Session 84334E60 (10.10.10.140:38403)=>(172.20.20.2:23) tcp SIS_OPEN
      Created 00:00:12, Last heard 00:00:08
      Bytes sent (initiator:responder) [53:162]
  Class-map: class-default (match-any)
    Match: any
    Drop
      0 packets, 0 bytes

```

Example 10-11 TCP Inspection and Sequence Numbers

```

! Host 10.10.10.140 (INSIDE router) telnets to 172.20.20.2 (ASA2)
INSIDE# telnet 172.20.20.2
Trying 172.20.20.2 ... Open
Password: *****
ASA2>
!
! TCP connection establishment as seen by source host
Reserved port 53250 in Transport Port Agent for TCP IP type 1
TCP: sending SYN, seq 551274375, ack 0
TCP2: Connection to 172.20.20.2:23, advertising MSS 536
TCP2: state was CLOSED -> SYNSENT [53250 -> 172.20.20.2(23)]
TCP2: state was SYNSENT -> ESTAB [53250 -> 172.20.20.2(23)]
TCP: tcb 84992A30 connection to 172.20.20.2:23, peer MSS 1380, MSS is 536
TCB84992A30 connected to 172.20.20.2.23
!
! The Policy Firewall is aware of Sequence and Acknowledgement numbers
FIREWALL* sis 84333E60: pak 83EA5870 SIS_CLOSED/LISTEN TCP SYN SEQ 551274375
LEN 0(10.10.10.140:53250) => (172.20.20.2:23)
FIREWALL* sis 84333E60: pak 83EA5870 SIS_OPENING/SYNSENT TCP SYN ACK 551274376

```

```
SEQ 3941962499 LEN 0(10.10.10.140:53250) <= (172.20.20.2:23)
FIREWALL* sis 84333E60: pak 83EA5870 SIS_OPENING/SYNRCVD TCP ACK 3941962500
SEQ 551274376 LEN 0(10.10.10.140:53250) => (172.20.20.2:23)
```

The previous two examples characterized ZFW's awareness of the TCP state machine. Some other ZFW capabilities for handling generic TCP inspection deserve specific mention:

- When the first packet of a TCP session contains flags other than the SYN, it is blocked by the ZFW because it does not comply with the three-way handshake rules.
- SYN packets containing data are dropped.
- TCP packets transporting an illegal combination of TCP flags are discarded.
- Invalid TCP segments and segments that contain invalid TCP Options are blocked.
- Segments with inconsistent values in the Sequence number or in the Acknowledgment number fields are dropped.
- *Out-of-Order* TCP segments are discarded.
- If one of the parties involved in a TCP session proposes an illegal window scaling, the connection is terminated.
- IP packets with a *length* value not adequate for carrying a standard TCP segment are blocked.

These protection mechanisms are similar to those provided by CBAC. One noticeable difference is that although for ZFW they are all enabled by default, some of them need to be explicitly configured in a CBAC environment.

ZFW and ACLs

The examples analyzed so far have employed either a **match protocol** or a **match access-group** statement to classify packets. A question, motivated by the knowledge of CBAC operations, is often raised at this point: *How can L3 filtering statements be combined with inspect rules to produce more restrictive policies?*

Example 10-12 refers to the topology in Figure 10-3 and shows that interface ACLs are not the right answer to this question. In this case, the restrictive ACL 100 was applied to interface F4.200 in the incoming direction, in classic CBAC style, with the goal of blocking inbound connection initiation. If such a configuration appeared in a real-life deployment, it would denote a misunderstanding of ZFW theory of operations because

- ZFW policies are unidirectional and do not require restrictive ACLs to block connection setup from the destination zone. This is the *implicit deny* behavior brought by ZFW.

- Input ACLs are processed before inspection happens. As a result, even though the echo request is inspected, the echo reply is blocked before ZFW can handle it.

Example 10-12 *An Important Concept About Interface ACLs and the Zone Firewall*

```

! Defining a restrictive ACL to be applied to the outside interface (F4.200)
IOS-ZFW# show access-list 100
Extended IP access list 100
    10 permit icmp any 10.10.10.0 0.0.0.255 unreachable log
    20 deny ip any any log
!
interface FastEthernet4.201
 ip address 10.10.10.1 255.255.255.0
 zone-member security INSIDE
!
interface FastEthernet4.200
 ip address 172.20.20.1 255.255.255.0
 ip access-group 100 in
 zone-member security OUTSIDE
!
! Outbound ping from 10.10.10.190 to 172.20.20.2
INSIDE# ping 172.20.20.2 source 10.10.10.190 repeat 1
Success rate is 0 percent (0/1)
!
! The echo request creates the session
FIREWALL* sis 84338A60: Session Created
FIREWALL* sis 84338A60: Pak 83EA5870
init_addr (10.10.10.190:0) resp_addr (172.20.20.2:0)
init_alt_addr (10.10.10.190:0) resp_alt_addr (172.20.20.2:0)
FIREWALL* sis 84338A60: FO cls 0x84ED0A20 clsgrp 0x10000000, target
0xA0000000, FO 0x84593BC0, alert = 1, audit_trail = 1, L7 = Unknown-
17, PAMID = 5
FIREWALL* sis 84338A60: Allocating L7 sis extension L4 = icmp, L7 = Unknown-17,
PAMID=5
!
! The interface ACL blocks the echo reply before inspection acts on return packets
%SEC-6-IPACCESSLOGDP: list 100 denied icmp 172.20.20.2 -> 10.10.10.190 (0/0), 1
packet

```

Suppose for a while that the ACL named OUT was configured, as in Example 10-13, with the goal of limiting not only the source/destination pairs allowed to cross the firewall but

also the application protocols subject to the inspection policy in Example 10-3 (OUTBOUND1). At first sight, this configuration seems to work fine:

- An echo request sent from 10.10.10.190 to 172.20.20.2 was permitted by ACL OUT and, in the sequence, inspected by the OUTBOUND1 policy. The opening for the echo reply was built, and this packet was correctly delivered to 10.10.10.190.
- Given that the ACL OUT is checked before inspection, any packet without the appropriate **permit** statement, arriving on F4.201, would have been discarded.

Example 10-14 depicts a situation in which the problems start to appear. A second zone policy called INBOUND1 was added to the scenario of Example 10-13, instructing the ZFW to promote generic inspection of TCP. An inbound telnet connection (from host 172.20.20.250 to 10.10.10.140) created the TCP session through the firewall, but its returning packets were blocked by the interface ACL before they could be handled by the inspection process. This shows that interface ACLs are definitely neither the right method to materialize address filtering nor to restrict the particular application protocols to be inspected by ZFW. Despite these issues, hope still exists. Examples 10-15 to 10-17 discuss the right way to add classic L3/L4 filtering to stateful inspection, without breaking ZFW functionality.

Example 10-13 *Interface ACL and Zone Firewall Interaction*

```
! New service object-groups
object-group service PING
  icmp echo
  icmp echo-reply

! Outbound ACL to be applied on interface F4.201 (filtering incoming traffic)
ip access-list extended OUT
  10 permit tcp object-group INSIDE1 object-group INTRANET eq telnet log
  20 permit object-group PING object-group INSIDE1 object-group INTRANET log
  1000 deny ip any any log
!
interface FastEthernet4.201
  ip address 10.10.10.1 255.255.255.0
  ip access-group OUT in
  zone-member security INSIDE
!
interface FastEthernet4.200
  ip address 172.20.20.1 255.255.255.0
  zone-member security OUTSIDE
!
! Zone Policy for traffic from the INSIDE zone to the OUTSIDE (refer to Example 10-3)
zone-pair security OUTBOUND1 source INSIDE destination OUTSIDE
```



```

service-policy type inspect GENERIC1
!
! Ping from 10.10.10.190 to 172.20.20.2 (echo-reply is received)
INSIDE# ping 172.20.20.2 source 10.10.10.190 repeat 1 size 1200
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
INSIDE# ICMP: echo reply rcvd, src 172.20.20.2, dst 10.10.10.190

! Interface ACL permits outbound traffic (before inspection)
%SEC-6-IPACCESSLOGDP: list OUT permitted icmp 10.10.10.190 -> 172.20.20.2 (8/0), 1
packet
!
! Session is created by the ZFW (traffic returns normally)
FIREWALL sis 8425BFE0: Session Created
FIREWALL sis 8425BFE0: Pak 840A4484
init_addr (10.10.10.190:0) resp_addr (172.20.20.2:0)
init_alt_addr (10.10.10.190:0) resp_alt_addr (172.20.20.2:0)
FIREWALL sis 8425BFE0: FO cls 0x85065DA0 clsgrp 0x10000000, target
0xA0000000, FO 0x844BAE20, alert = 1, audit_trail = 1, L7 = Unknown-
17, PAMID = 5
FIREWALL sis 8425BFE0: Allocating L7 sis extension L4 = icmp, L7 = Unknown-17,
PAMID= 5

```

Example 10-14 *Effect of Outbound Interface ACL on Inbound Policy*

```

! Inbound Zone-pair policy (add-on to Example 10-13)
class-map type inspect match-any BASIC3
  match protocol tcp
!
policy-map type inspect INBOUND3
  class type inspect BASIC3
    inspect TRACKING
  class class-default
    drop log
!
zone-pair security INBOUND1 source OUTSIDE destination INSIDE
  service-policy type inspect INBOUND3
!
! Host 172.20.20.250 telnets to 10.10.10.140 (inbound session is created by ZFW)
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(INBOUND1:BASIC3):Start telnet session:
initiator (172.20.20.250:1029) — responder (10.10.10.140:23)
!
! The interface ACL blocks the return packet before it is permitted by inspection
%SEC-6-IPACCESSLOGP: list OUT denied tcp 10.10.10.140(23) -> 172.20.20.250(1029),
1 packet

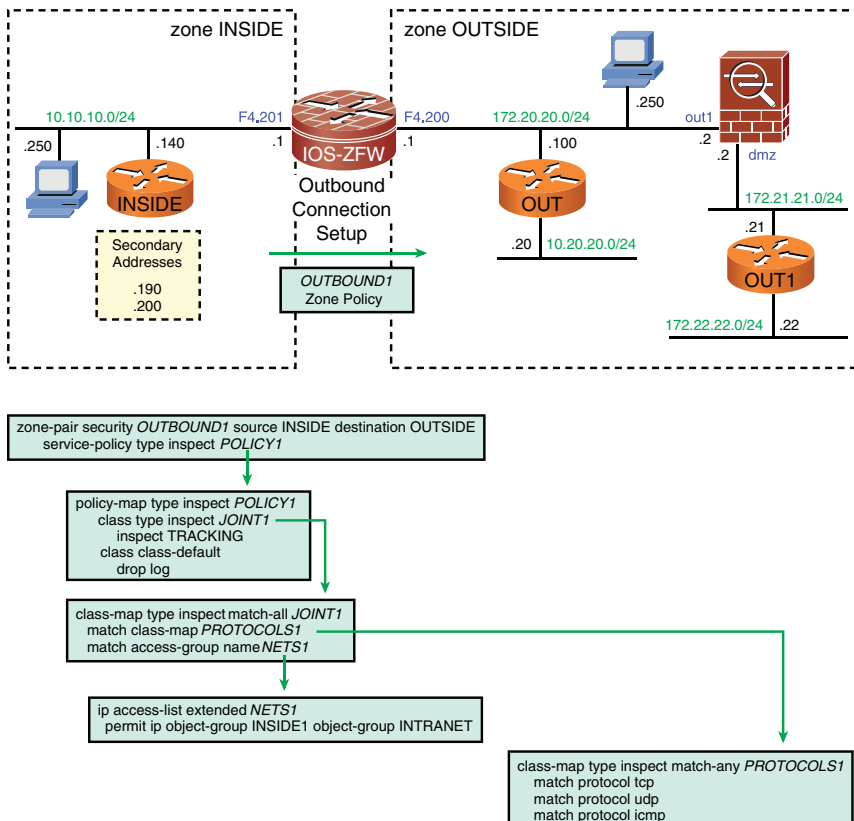
```

```

! After blocking the return traffic, an ICMP 3/13 message is sent back to
10.10.10.140
ICMP: dst (172.20.20.250) administratively prohibited unreachable sent to
10.10.10.140
!
! The ZFW cannot handle the session correctly (as a result of the ACL drops)
FIREWALL* sis 84229160: L4 result: SKIP packet 0x83D9E9F0
(172.20.20.250:1029) (10.10.10.140:23) bytes 28 ErrStr =
Retransmitted Segment
FIREWALL* sis 84229160: L4 result: DROP packet 0x83D9E9F0
(10.10.10.140:23) (172.20.20.250:1029) bytes 24 ErrStr = Out-Of-Order
Segment

```

Figure 10-5 shows the reference topology and the companion Zone Policy structure for a scenario in which ACLs are integrated in the right manner into the ZFW security design. The underlying goal is to provide generic outbound inspection (TCP, UDP, and ICMP) for packets travelling from the set of networks defined by **object-group** INSIDE1 to the IP address block 172.16.0.0/12 (specified by **object-group** INTRANET since Example 10-8). The configuration commands summarized in this figure are detailed in Example 10-15. The roles of the elements that compose this firewall policy follow:



- The **class-map** PROTOCOLS1 uses a **match-any** approach to select the protocol that is subject to generic inspection.
- The **access-list** NETS1 defines the source/destination pairs that are enabled to activate the ZFW. Any combination of addresses outside of this scope is denied.
- A second class called JOINT1 is in charge of selecting packets in accord with the PROTOCOLS1 class and that simultaneously obey the address rules established by ACL NETS1. This is the reason to use a **match-all** class-map. (This capability of referencing one **class-map** inside another adds flexibility to ZFW-based policies.)
- **Policy-map** POLICY1 states that packets complying to class JOINT1 are inspected. Otherwise, the *class-default* comes into the scene and drops the packet.
- **Service-policy** POLICY1 is referenced by the **zone-pair** OUTBOUND1 to connect the source zone INSIDE to the destination zone OUTSIDE.

Example 10-15 also illustrates the effect of the configurations in place:

- A ping to a valid destination (172.20.20.100) matches the two conditions in the class JOINT1, and therefore, is allowed to cross the firewall. A hit is added to **access-list** NETS1, but the session log is a duty of the **audit-trail** feature.
- A ping to destination 10.20.20.20, not contained in the ACL NETS1, is dropped by the *class-default*.
- A telnet to 10.20.20.20 (invalid destination) does not match the ACL clause within class JOINT1 and is also blocked by the *class-default*.

Example 10-15 *The Right Way to Integrate ACLs with the ZFW (1)*

```
! Defining source and destination hosts that will be used as a match criterion
ip access-list extended NETS1
 permit ip object-group INSIDE1 object-group INTRANET
!
! Class-map that matches any of the protocols (representing generic inspection)
class-map type inspect match-any PROTOCOLS1
 match protocol tcp
 match protocol udp
 match protocol icmp
```

```

!
! Class-map matching 02 conditions simultaneously
class-map type inspect match-all JOINT1
  match class-map PROTOCOLS1
  match access-group name NETS1
!
! Defining and applying an outbound Zone-pair security policy
policy-map type inspect POLICY1
  class type inspect JOINT1
    inspect TRACKING
  class class-default
    drop log
!
zone-pair security OUTBOUND1 source INSIDE destination OUTSIDE
  service-policy type inspect POLICY1
!
! Ping to a valid destination matches the 02 clauses of class JOINT1
INSIDE# ping 172.20.20.100 source 10.10.10.190 size 1000 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 8/8/8 ms

%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:JOINT1):Start
icmp session: initiator (10.10.10.190:0) — responder
(172.20.20.100:0)
!
IOS-ZFW# show access-list NETS1 ! include match
  10 permit ip object-group INSIDE1 object-group INTRANET (1 match)
!
! Ping to a destination not matching ACL 'NETS1' (10.20.20.20) is blocked
INSIDE# ping 10.20.20.20 source 10.10.10.190 size 1200 repeat 1
Success rate is 0 percent (0/1)
!
%FW-6-DROP_PKT: Dropping icmp session 10.10.10.190:0 10.20.20.20:0 on
zone-pair OUTBOUND1 class class-default due to DROP action found in
policy-map with ip ident 0
%FW-6-LOG_SUMMARY: 1 packet were dropped from 10.10.10.190:8 => 10.20.20.20:0
(target:class)-(OUTBOUND1:class-default)
!
! Telnet session to an invalid destination (according to ACL NETS1) is blocked
INSIDE# telnet 10.20.20.20
Trying 10.20.20.20 ...
% Connection timed out; remote host not responding

%FW-6-DROP_PKT: Dropping Other session 10.10.10.140:54274
10.20.20.20:23 on zone-pair OUTBOUND1 class class-default due to

```

```
DROP action found in policy-map with ip ident 0
%FW-6-LOG_SUMMARY: 4 packets were dropped from 10.10.10.140:54274 =>
10.20.20.20:23 (target:class)-(OUTBOUND1:class-default)
!
IOS-ZFW# show policy-map type inspect zone-pair OUTBOUND1 | begin Drop
Drop
4 packets, 96 bytes
```

Example 10-16 brings a second situation in which a **match-all class-map (JOINT2)** limits the source/destination pairings enabled to initiate inbound connections through the ZFW. Figure 10-6 shows the network topology to which these configurations apply. The basic design goals that guided this policy construction follow:

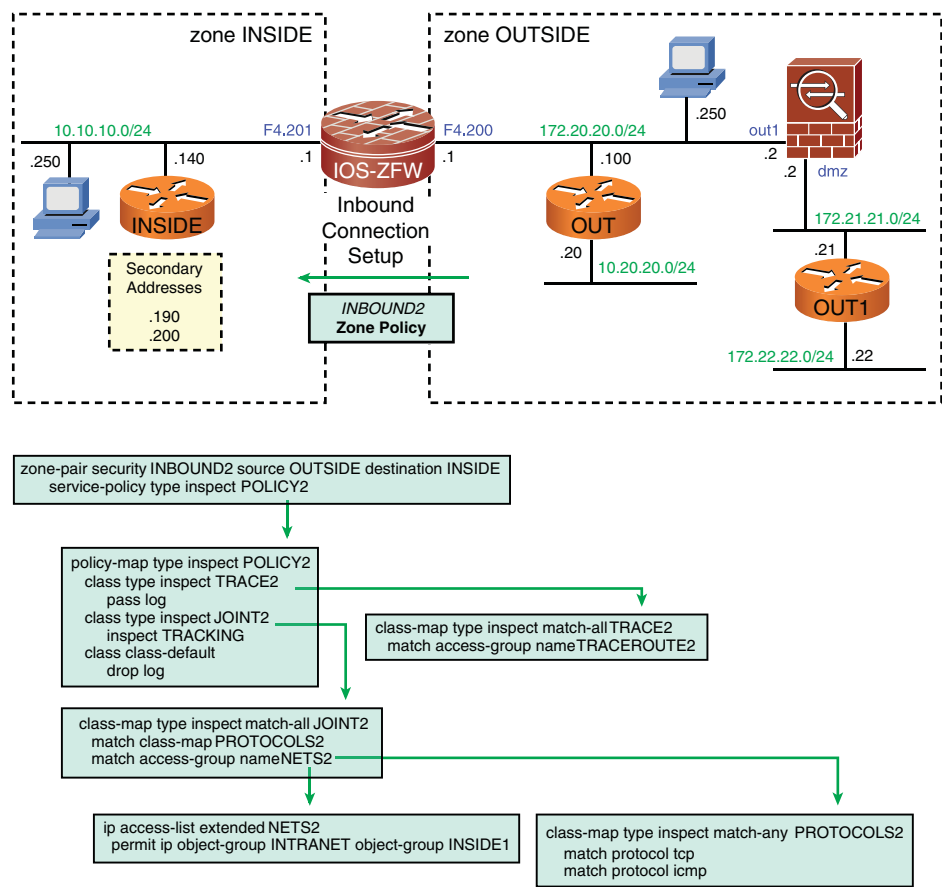


Figure 10-6 Reference Topology for Example 10-16

- Allow the ICMP messages used for traceroute back to the originating hosts (without inspection). This is accomplished by prescribing a **pass** action within **policy-map** POLICY2 for the class TRACE2.
- Allow TCP and ICMP connection initiation (followed by proper stateful inspection) for inbound requests coming from sources in the 172.16.0.0/12 block (INTRANET) and destined to the subnets specified by the INSIDE1 **object-group**. This is achieved by assigning an **inspect** action to the class JOINT2 under **policy-map** POLICY2 (and later attaching this **service-policy** to the **zone-pair** INBOUND2).
- The previous requirements must not interfere with the outbound policy OUT-BOUND1, earlier developed in Example 10-15. These two policies are meant to coexist harmonically.

Example 10-16 also documents the behavior of this additional configuration:

- An outbound IOS **traceroute** issued by 10.10.10.190 and heading toward the valid destination 172.22.22.22 (from ACL NETS1 standpoint) is permitted by policy OUT-BOUND1. The auxiliary ICMP messages are allowed back by class TRACE2.
- An outbound IOS **traceroute** directed to 10.20.20.20 (outside of the scope determined by ACL NETS1) is blocked by the *class-default* under **policy-map** OUTBOUND1.
- An inbound ping from the invalid source 10.20.20.20 is blocked because it does not comply with the rules stated by ACL NETS2.
- An inbound NTP packet sourced from 172.20.20.100 is dropped because all UDP-based applications are prohibited by class PROTOCOLS2 (irrespective of the IP addresses involved).

Example 10-16 *The Right Way to Integrate ACLs with the ZFW (2)*

```
! ACL and class-map to allow traceroute between selected source/destination pairs
ip access-list extended TRACEROUTE2
  permit object-group TRACEROUTE object-group INTRANET object-group INSIDE1
!
class-map type inspect match-all TRACE2
  match access-group name TRACEROUTE2
!
! Defining source/destination combinations that will be subject to inbound
inspection
ip access-list extended NETS2
  permit ip object-group INTRANET object-group INSIDE1
!
! Protocols subject to inbound inspection
class-map type inspect match-any PROTOCOLS2
  match protocol tcp
```

```

match protocol icmp
!
! Class-map matching 02 conditions simultaneously
class-map type inspect match-all JOINT2
  match class-map PROTOCOLS2
  match access-group name NETS2
!
policy-map type inspect POLICY2
  class type inspect TRACE2
    pass log
  class type inspect JOINT2
    inspect TRACKING
  class class-default
    drop log
!
! Defining and applying an inbound Zone-pair security Policy
zone-pair security INBOUND2 source OUTSIDE destination INSIDE
  service-policy type inspect POLICY2
!
! The route is traced correctly between an allowed source/destination pair
INSIDE# traceroute 172.22.22.22 source 10.10.10.190 probe 1
Tracing the route to 172.22.22.22
  1 10.10.10.1 4 msec
  2 172.21.21.2 0 msec
  3 172.21.21.21 0 msec
ICMP: time exceeded rcvd from 10.10.10.1
ICMP: time exceeded rcvd from 172.21.21.2
ICMP: dst (10.10.10.190) port unreachable rcv from 172.21.21.21
!
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:JOINT1):Start udp session:
  initiator (10.10.10.190:49163) — responder (172.22.22.22:33435)
%FW-6-PASS_PKT: (target:class)-(INBOUND2:TRACE2) Passing icmp pkt 172.21.21.2:0 =>
10.10.10.190:0 with ip ident 0
!
IOS-ZFW# show policy-map type inspect zone-pair sessions | include Session
      Number of Half-open Sessions = 2
      Half-open Sessions
        Session 8488F780 (10.10.10.190:49163)=>(172.22.22.22:33435) udp SIS_OPENING
        Session 8488F980 (10.10.10.190:49164)=>(172.22.22.22:33436) udp SIS_OPENING
!
! IOS traceroute blocked for a source/destination pair not allowed by ACL 'NETS1'
INSIDE# traceroute 10.20.20.20 source 10.10.10.190 probe 1
Type escape sequence to abort.
Tracing the route to 10.20.20.20

```

```

1 10.10.10.1 4 msec
2 *
3 *
!
! First UDP probe (TTL = 1) dropped by the IOS-ZFW router
ICMP: time exceeded (time to live) sent to 10.10.10.190 (dest was 10.20.20.20)
!
! Subsequent UDP Probes (destined to 10.20.20.20) not allowed through the firewall
%FW-6-DROP_PKT: Dropping udp session 10.10.10.190:49167
10.20.20.20:33435 on zone-pair OUTBOUND1 class class-default due to
DROP action found in policy-map with ip ident 0
%FW-6-LOG_SUMMARY: 1 packet were dropped from 10.10.10.190:49167 =>
10.20.20.20:33435 (target:class)-(OUTBOUND1:class-default)
%FW-6-LOG_SUMMARY: 1 packet were dropped from 10.10.10.190:49168 =>
10.20.20.20:33436 (target:class)-(OUTBOUND1:class-default)
!
! Inbound ping from invalid source 10.20.20.20 is blocked (does not match ACL
'NETS2')
OUT1# ping 10.10.10.140 source 10.20.20.20 repeat 1
Success rate is 0 percent (0/1)
!
%FW-6-DROP_PKT: Dropping icmp session 10.20.20.20:0 10.10.10.140:0 on
zone-pair INBOUND2 class class-default due to DROP action found in
policy-map with ip ident 0
%FW-6-LOG_SUMMARY: 1 packet were dropped from 10.20.20.20:8 =>
10.10.10.140:0
(target:class)-(INBOUND2:class-default)
!
! Inbound NTP packet is blocked (UDP is not a valid protocol)
OUT# NTP message sent to 10.10.10.140, from interface 'FastEthernet4.200'
(172.20.20.100).

%FW-6-DROP_PKT: Dropping Other session 172.20.20.100:123
10.10.10.140:123 on zone-pair INBOUND2 class class-default due to
DROP action found in policy-map with ip ident 0
%FW-6-LOG_SUMMARY: 1 packet were dropped from 172.20.20.100:123 =>
10.10.10.140:123 (target:class)-(INBOUND2:class-default)

```

The two previous examples covered scenarios in which conditions regarding source and destination IP addresses were coupled with the original inspection rules. Example 10-17 complements this theory by replacing the original ACL NETS1 by another, called ACL1, which is aimed at imposing restrictions on the application protocols that are allowed to start sessions through the ZFW. The authorized protocols are described by the services object-group BASIC-MGMT.

The **match-all class-map** JOINT1 is born from the union of ACL1 with the class PROTOCOLS1. The **inspect** action applied to class JOINT1 lies at the core of the **policy-map** MGMT1 used to materialize the zone policy OUTBOUND1. (In spite of this change in the **policy-map**, the reference topology is still that in Figure 10-5.)

Two outbound TCP session requests (telnet and HTTP), as handled by this **zone-pair** rule, are also shown in Example 10-17. These two connection attempts are blocked due to the intervention of **access-list** ACL1, which clearly states that for TCP only SSH and HTTPS services are permitted.

Example 10-17 A More Complex ACL Combined with a Zone Policy

```
! Defining the services object-group and using it on ACL named 'ACL1'
object-group service BASIC-MGMT
  tcp eq 443
  tcp eq 22
  icmp echo
  udp range 33434 33534
!
ip access-list extended ACL1
  10 permit object-group BASIC-MGMT object-group INSIDE1 object-group INTRANET
  997 deny tcp any any
  998 deny udp any any
  999 deny icmp any any
  1000 deny ip any any
!
! Replacing the ACL for the 'JOINT1' class-map (reference topology in Figure 10-5)
class-map type inspect match-all JOINT1
  match class-map PROTOCOLS1
  match access-group name ACL1
!
! Defining and applying an outbound Zone-pair security policy
policy-map type inspect MGMT1
  class type inspect JOINT1
    inspect TRACKING
  class class-default
    drop log
!
zone-pair security OUTBOUND1 source INSIDE destination OUTSIDE
  service-policy type inspect MGMT1
!
! Outbound telnet from 10.10.10.140 to 172.20.20.2 is blocked
%FW-6-DROP_PKT: Dropping Other session 10.10.10.140:40450
172.20.20.2:23 on zone-pair OUTBOUND1 class class-default due to
DROP action found in policy-map with ip ident 0
%FW-6-LOG_SUMMARY: 4 packets were dropped from 10.10.10.140:40450 =>
```

```

172.20.20.2:23 (target:class)-(OUTBOUND1:class-default)
!
IOS-ZFW# show access-1 ist ACL1 | include match
      997 deny tcp any any (4 matches)
!
! Outbound HTTP from 10.10.10.250 to 172.20.20.100 is blocked
%FW-6-DROP_PKT: Dropping http session 10.10.10.250:1039
172.20.20.100:80 on zone-pair OUTBOUND1 class class-default due to
DROP action found in policy-map with ip ident 0
%FW-6-LOG_SUMMARY: 3 packets were dropped from 10.10.10.250:1039 =>
172.20.20.100:80 (target:class)-(OUTBOUND1:class-default)
!
IOS-ZFW# show access-list ACL1 | include match
      997 deny tcp any any (3 matches)

```

Note Be attentive to the absence of any interface ACL in the last 03 scenarios. The purpose was to show that, when the right configuration path is taken, complex policies are certainly possible with the ZFW methodology. You just need an open mind to adapt to this new way of building firewall policies.

ZFW and NAT

At this stage, the theory of operations of NAT and its applicability to practical designs are considered well-known topics. IOS NAT techniques were already discussed in Chapter 9, and this section shows some examples aimed at characterizing that NAT does not pose any problem to ZFW.

Figure 10-7 depicts the reference topology for the analysis of inside static NAT. Example 10-18 refers to the network represented in this figure. In this scenario, the NAT reference inside/outside for router interfaces coincides with the names assigned to the security zones. The internal host 10.10.10.190 (inside local) is statically translated to 172.20.20.190 (inside global). The example details the behavior of such a configuration for an outbound ping from 10.10.10.190 toward 172.20.20.2:

- Translation of the source address happens before inspection.
- Generic ICMP inspect session takes place, and the echo request is delivered to the target host (172.20.20.2). It is noteworthy that the ZFW process is totally aware of the address translation events (compare *init_addr* and *init_alt_addr* parameters).
- 172.20.20.2 sends the echo reply back to 172.20.20.190. (The address it sees.)
- The destination address in the echo reply packet is *untranslated* to 10.10.10.190.
- The example also reveals how flows are counted with Netflow, either on the ingress or egress direction.

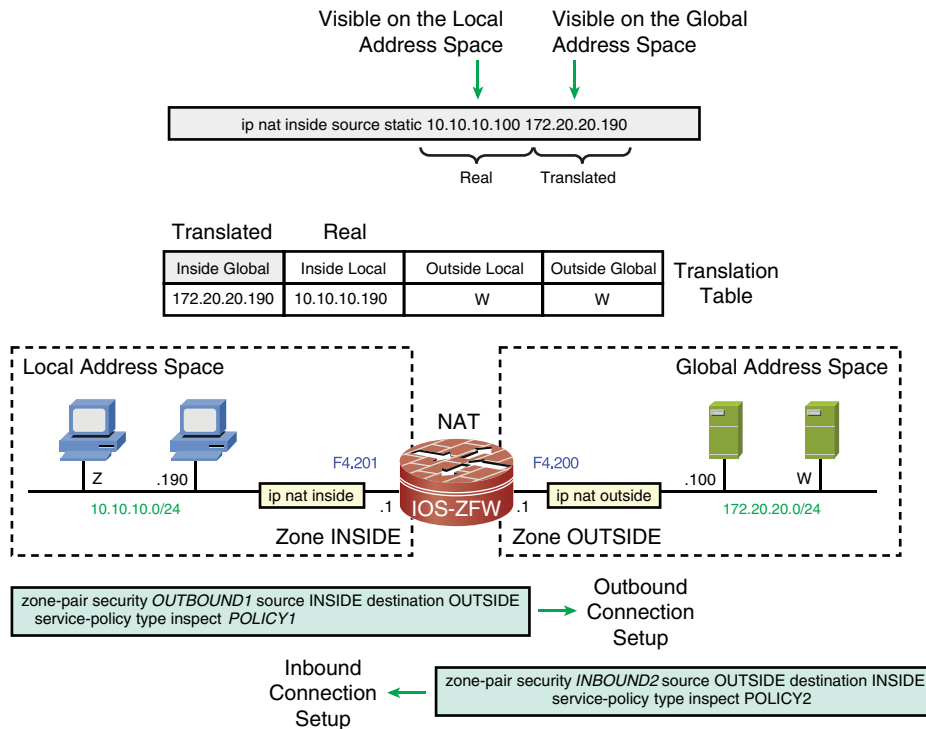


Figure 10-7 Reference Topology for Inside NAT Combined with ZFW

Example 10-18 Sample Inside NAT and the ZFW

```
! Instruct IOS to log translation activation
ip nat log translations syslog
!
! Establishing inside/outside reference for NAT
interface FastEthernet4.201
 ip address 10.10.10.1 255.255.255.0
 ip flow egress
 ip nat inside
 zone-member security INSIDE
!
interface FastEthernet4.200
 ip address 172.20.20.1 255.255.255.0
 ip flow ingress
 ip nat outside
 zone-member security OUTSIDE
!
! Zone-pair security policies previously defined (Example 10-16)
zone-pair security OUTBOUND1 source INSIDE destination OUTSIDE
```

```

service-policy type inspect POLICY1
!
zone-pair security INBOUND2 source OUTSIDE destination INSIDE
service-policy type inspect POLICY2
!
! Defining an inside static mapping
ip nat inside source static 10.10.10.190 172.20.20.190
!
! Outbound ping from 10.10.10.190 to 172.20.20.2
INSIDE# ping 172.20.20.2 source 10.10.10.190 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
!
! Translation of the source address happens (before inspection)
%IPNAT-6-CREATED: icmp 10.10.10.190:5 172.20.20.190:5 172.20.20.2:5 172.20.20.2:5
NAT*: s=10.10.10.190->172.20.20.190, d=172.20.20.2 [6]
!
! Generic ICMP inspect session is created
init_addr (10.10.10.190:0) resp_addr (172.20.20.2:0)
init_alt_addr (172.20.20.190:0) resp_alt_addr (172.20.20.2:0)
FIREWALL* sis 84890380: F0 cls 0x850C01A0 clsgrp 0x10000000, target
0xA0000000, F0 0x848C3980, alert = 1, audit_trail = 1, L7 = Unknown-
17, PAMID = 5
FIREWALL* sis 84890380: Allocating L7 sis extension L4 = icmp, L7 = Unknown-17,
PAMID=5
!
! Echo reply packet is untranslated
NAT*: s=172.20.20.2, d=172.20.20.190->10.10.10.190 [10176]
!
! Ingress and egress flows are created
IOS-ZFW# show ip cache flow | begin Src

```

SrcIf	SrcIPAddress	DstIf	DstIPAddress	Pr	SrcP	DstP	Pkts
Fa4.200	172.20.20.2	Fa4.201	172.20.20.190	01	0000	0000	1
Fa4.200	172.20.20.2	Fa4.201*	10.10.10.190	01	0000	0000	1

Figure 10-8 is the natural companion for the outside NAT sample scenario proposed in Example 10-19. In this reference topology, which builds upon the configuration presented in Example 10-18, the real host 172.20.20.100 (outside global) was statically mapped to the internal address 10.10.10.100 (outside local).

Example 10-19 portrays the sequence of operations for an outbound ping, from host 10.10.10.140 to 10.10.10.100, when outside NAT and ZFW inspection are enabled at the same time. Some remarkable facts related to this environment follow:

- Untranslation of the destination address (10.10.10.100 > 172.20.20.100) takes place before generic ICMP inspection comes into play.

- The echo request is inspected, and a session is created between the real addresses 10.10.10.190 (inside local) and 172.20.20.100 (outside global).
- 172.20.20.100 sends the echo reply to 10.10.10.140 and the source address of this packet is translated to 10.10.10.100 (outside local).
- The example also reveals how Netflow comes into the picture to count both ingress and egress flows.

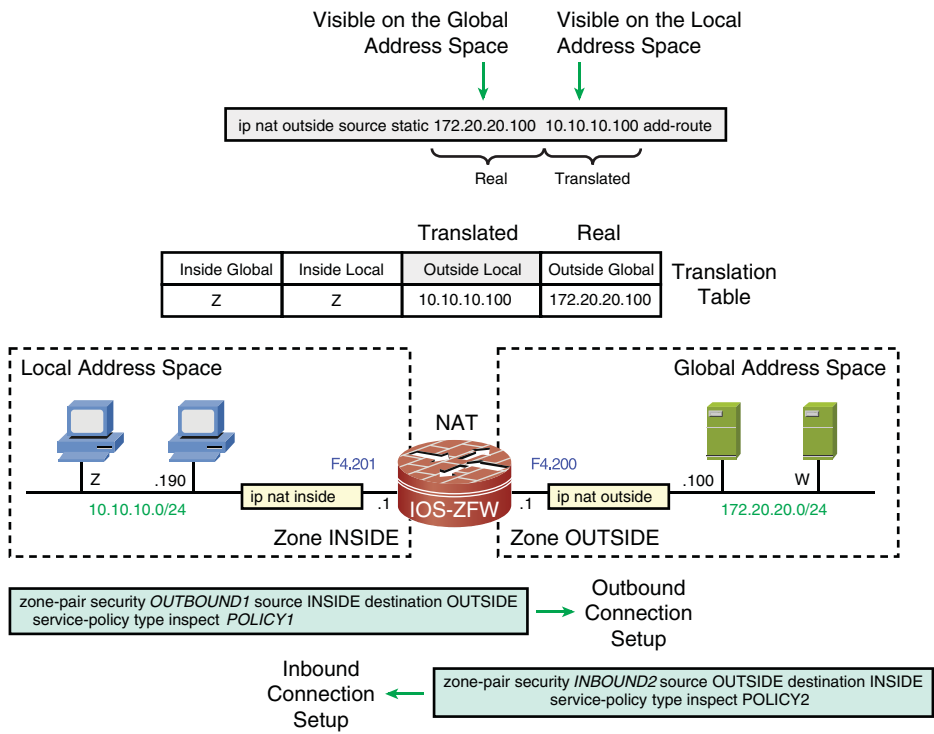


Figure 10-8 Reference Topology for Outside NAT Combined with ZFW

Example 10-19 Sample Outside NAT and the ZFW

```
! Adding an outside NAT statement
ip nat outside source static 172.20.20.100 10.10.10.100 add-route
!
! Outbound ping from 10.10.10.140 to 10.10.10.100 (outside local)
INSIDE# ping 10.10.10.100 source 10.10.10.140 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
!
! Untranslation of the destination address happens (before inspection)
%IPNAT-6-CREATED: icmp 10.10.10.140:11 10.10.10.140:11 10.10.10.100:11
172.20.20.100:11
```

```

NAT: s=10.10.10.140, d=10.10.10.100->172.20.20.100 [12]
!
! ICMP inspect session is created
init_addr (10.10.10.140:0) resp_addr (172.20.20.100:0)
init_alt_addr (10.10.10.140:0) resp_alt_addr (10.10.10.100:0)
FIREWALL sis 84890B80: FO cls 0x850C01A0 clsgrp 0x10000000, target
0xA0000000, FO 0x848C3980, alert = 1, audit_trail = 1, L7 = Unknown-
17, PAMID =5
FIREWALL sis 84890B80: Allocating L7 sis extension L4 = icmp, L7 = Unknown-17,
PAMID=5
!
IOS-ZFW# show policy-map type inspect zone-pair sessions | include Session
      Number of Established Sessions = 1
      Established Sessions
        Session 84890B80 (10.10.10.140:8)=>(172.20.20.100:0) icmp SIS_OPEN
!
! Source address in the echo reply is translated to 10.10.10.100
NAT*: s=172.20.20.100->10.10.10.100, d=10.10.10.140 [12]
!
! Ingress and egress flows are created
IOS-ZFW# show ip cache flow | begin Src

```

SrcIf	SrcIPAddress	DstIf	DstIPAddress	Pr	SrcP	DstP	Pkts
Fa4.200	10.10.10.100	Fa4.201*	10.10.10.140	01	0000	0000	1
Fa4.200	172.20.20.100	Fa4.201	10.10.10.140	01	0000	0000	1

Example 10-20 is basically a composition of its two antecedents. This scenario mixes the inside and outside static NAT definitions shown in Figures 10-7 and 10-8, giving rise to the Dual NAT effect. The order of operations, when the internal host 10.10.10.190 (inside local) issues a ping to the address 10.10.10.100 (outside local), is the following:

- Translation of the source address (10.10.10.190 > 172.20.20.190).
- Untranslation of the destination address (10.10.10.100 > 172.20.20.100).
- Generic ICMP inspection triggered by the echo request packet. ZFW has complete visibility of the NAT processes going on, which is characterized by the mappings seen in the **debug policy-firewall obj-creation** output (*init_addr - init_alt_addr* and *resp_addr - resp_alt_addr*). The ZFW session is always created between the real addresses.
- Translation of the source address in the echo-reply (172.20.20.100 > 10.10.10.100).
- Untranslation of the destination address in the echo-reply (172.20.20.190 > 10.10.10.190).
- Netflow is particularly useful for Dual NAT because it unveils the addresses visible on each address space. As already studied in Chapter 9, when enabled for inbound

traffic (outside to inside), ingress Netflow precedes NAT and sees global addresses. Conversely, egress Netflow happens after NAT and is aware of local addresses.

Example 10-20 *Dual NAT and the ZFW*

```
! Outbound ping from 10.10.10.190 to 10.10.10.100 (outside local)
INSIDE# ping 10.10.10.100 source 10.10.10.190 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
!
! Translation of the source address (inside local > inside global)
NAT: s=10.10.10.190->172.20.20.190, d=10.10.10.100 [11]
!
! Untranslation of the destination address (outside local > outside global)
NAT: s=172.20.20.190, d=10.10.10.100->172.20.20.100 [11]
!
! Inspect session is created
init_addr (10.10.10.190:0) resp_addr (172.20.20.100:0)
init_alt_addr (172.20.20.190:0) resp_alt_addr (10.10.10.100:0)
FIREWALL sis 84890980: FO cls 0x850C01A0 clsgrp 0x10000000, target
0xA0000000, FO 0x848C2F80, alert = 1, audit_trail = 1, L7 = Unknown-
17, PAMID =5
FIREWALL sis 84890980: Allocating L7 sis extension L4 = icmp, L7 = Unknown-17,
PAMID =5
!
! Translation of the echo reply source address (outside global > outside local)
NAT*: s=172.20.20.100->10.10.10.100, d=172.20.20.190 [11]
!
! Untranslation of the echo reply destination address (inside global > inside
local)
NAT*: s=10.10.10.100, d=172.20.20.190->10.10.10.190 [11]
!
! The connection is always created between the real addresses
IOS-ZFW# show policy-map type inspect zone-pair sessions | include Session
      Number of Established Sessions = 1
      Established Sessions
          Session 84890980 (10.10.10.190:8)=>(172.20.20.100:0) icmp SIS_OPEN
!
! Ingress and egress flows are created
IOS-ZFW# show ip cache flow | begin Session
SrcIf      SrcIPaddress  DstIf      DstIPaddress  Pr SrcP DstP  Pkts
Fa4.200    172.20.20.100  Fa4.201    172.20.20.190  01 0000 0000  1
Fa4.200    10.10.10.100   Fa4.201*   10.10.10.190  01 0000 0000  1
```

This section ends by discussing the behavior of a ZFW policy that involves ACL-based filtering with NAT.

Example 10-21 refers to the topology depicted in Figure 10-9. In this sample network, a ZFW policy controlling traffic from the INSIDE to the OUTSIDE zone was defined.

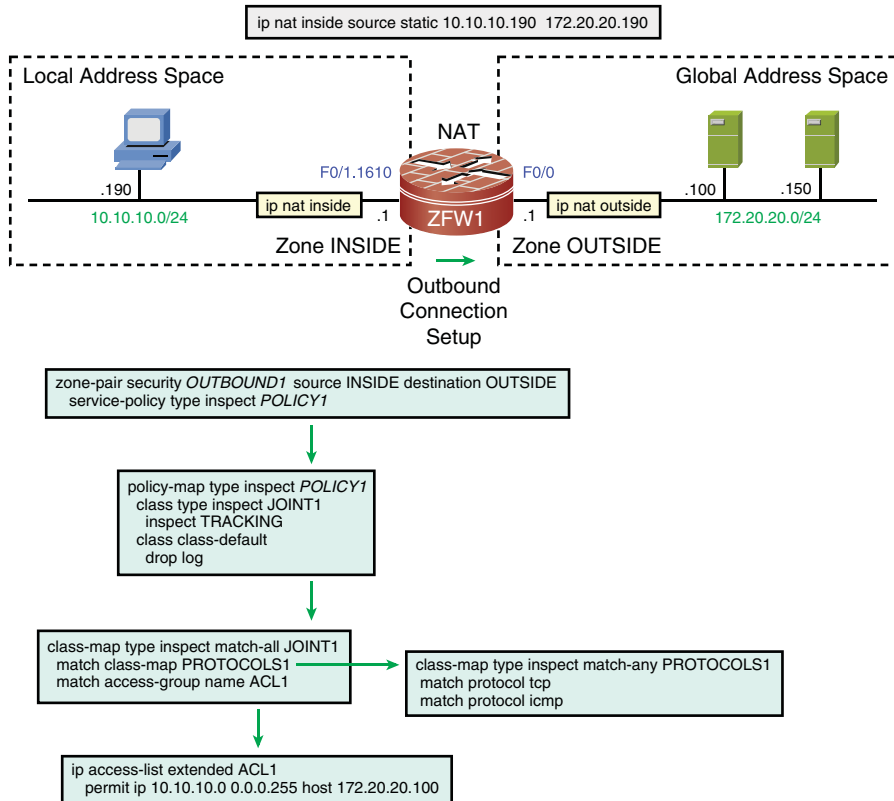


Figure 10-9 NAT, ACL and ZFW (1)

Example 10-21 highlights some important facts:

- Translation of the source address takes place before ZFW inspection.
- The source address in ACL1 must specify the real addresses of the inside hosts (represented, in this case, by subnet 10.10.10.0/24).
- A ping destined to 172.20.20.150 (an address not permitted by ACL1), is dropped. This happens because the **match** statement associated with ACL1 under **class-map JOINT1** was not met.

Note In the event NAT is defined for an outside host, the ACL should refer to the real address of the destination. For example, if the command **ip nat outside source static 172.20.20.100 10.10.10.100 add-route** had been configured, ACL1 would need to include a **permit** statement for the destination host 172.20.20.100 (rather than to 10.10.10.100).

Example 10-21 *Outbound Connection, NAT, ACL, and ZFW*

```

! Ping from 10.10.10.190 to 172.20.20.100 (permitted by ACL1)
%IPNAT-6-CREATED: icmp 10.10.10.190:10 172.20.20.190:10 172.20.20.100:10
172.20.20.100:10
FIREWALL* sis 48589740: Session Created
FIREWALL* sis 48589740: Pak 47CE3220 init_addr (10.10.10.190:0) resp_addr
(172.20.20.100:0)
init_alt_addr (172.20.20.190:0)
resp_alt_addr (172.20.20.100:0)
FIREWALL* sis 48589740: FO cls 0x4769BA80 clsgrp 0x10000000, target
0xA0000000, FO 0x485ED780, alert = 1, audit_trail = 1, L7 = Unknown-
17, PAMID = 5
FIREWALL* sis 48589740: Allocating L7 sis extension L4 = icmp, L7=Unknown-17,
PAMID = 5

%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:JOINT1):Start
icmp session: initiator (10.10.10.190:0) — responder
(172.20.20.100:0)
!
ZFW1# show policy-map type inspect zone-pair OUTBOUND1 sessions | include Session
      Number of Established Sessions = 1
      Established Sessions
        Session 48589740 (10.10.10.190:8)=>(172.20.20.100:0) icmp SIS_OPEN
!
! The ACL used in the Zone policy must specify the real addresses of the inside
hosts
ZFW1# show access-list ACL1
Extended IP access list ACL1
    10 permit ip 10.10.10.0 0.0.0.255 host 172.20.20.100 (1 match)
!
! Ping from 10.10.10.190 to 172.20.20.150 (not permitted by ACL1)

%IPNAT-6-CREATED: icmp 10.10.10.190:11 172.20.20.190:11 172.20.20.150:11
172.20.20.150:11
%FW-6-DROP_PKT: Dropping icmp session 10.10.10.190:0 172.20.20.150:0
on zone-pair OUTBOUND1 class class-default due to DROP action found
in policy-map with ip ident 0

```

Example 10-22 relates to the scenario portrayed in Figure 10-10. This new arrangement focuses on an inbound ZFW policy to complement the analysis presented in Example 10-21. NAT still happens before ZFW inspection, and the destination address associated with the ACL (under **class-map JOINT2**) is the real address of the inside host (10.10.10.190 in this case).

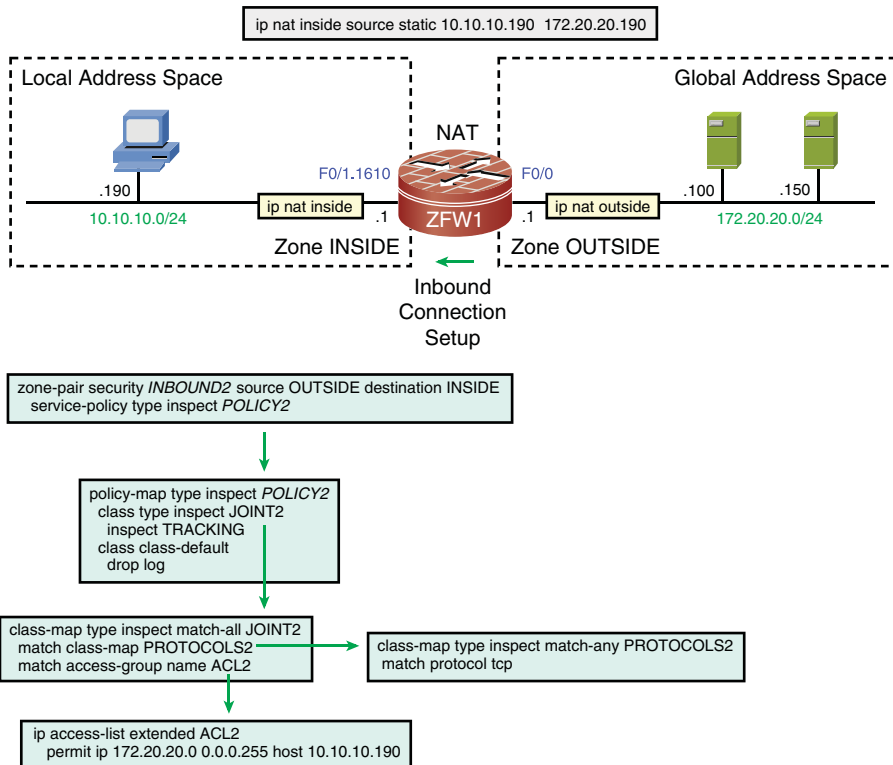


Figure 10-10 NAT, ACL and ZFW (2)

Example 10-22 *Inbound Connection, NAT, ACL, and ZFW*

```

! 172.20.20.100 telnets to 172.20.20.190 (translated address)
%IPNAT-6-CREATED: tcp 10.10.10.190:23 172.20.20.190:23 172.20.20.100:33946
172.20.20.100:33946
FIREWALL* sis 48589740: Session Created
FIREWALL* sis 48589740: Pak 47A9827C init_addr (172.20.20.100:33946)
resp_addr (10.10.10.190:23) init_alt_addr (172.20.20.100:33946)
resp_alt_addr (172.20.20.190:23)
FIREWALL* sis 48589740: FO cls 0x4769BA80 clsgrp 0x10000000, target
0xA0000007, FO 0x485ED680, alert = 1, audit_trail = 1, L7 = Other,
PAMID = 2
FIREWALL* sis 48589740: Allocating L7 sis extension L4 = tcp, L7 = Other, PAMID = 2
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(INBOUND2:JOINT2):Start
telnet session: initiator (172.20.20.100:33946) — responder
  
```

```

(10.10.10.190:23)
FIREWALL* sis 48589740: create host entry 482136C0 addr 10.10.10.190 bucket 180
(vrf 0:0) fwfo 0x49FF5B40
!
ZFW1# show policy-map type inspect zone-pair INBOUND2 sessions | include Session
      Number of Established Sessions = 1
      Established Sessions
        Session 48589740 (172.20.20.100:33946)=>(10.10.10.190:23) telnet:tcp
SIS_OPEN
!
! The ACL used in the Zone policy needs to specify the real addresses
ZFW1# show access-list ACL2
Extended IP access list ACL2
  10 permit ip 172.20.20.0 0.0.0.255 host 10.10.10.190 (1 match)

```

Note It might be instructive to compare this last ZFW example with the behavior of CBAC in a similar situation. ZFW uses real addresses in the ACL definitions, whereas CBAC (which relies on interface ACLs) references the translated addresses of the destination hosts.

ZFW in Transparent Mode

This section briefly examines the use of the ZFW approach in a topology that employs the Transparent mode. The policy building blocks and structure are completely analogous to those used so far, with the particularity that the ZFW-enabled interfaces reside on the same L3 subnet.

Example 10-23 assembles the commands for Transparent mode operation in the network represented in Figure 10-11. In this arrangement, Routers R1 and R2 are connected to subnet 10.10.10.0/24 but located on different security zones. A ZFW policy called OUTBOUND1 controls connection setup from zone INSIDE to zone OUTSIDE.

Example 10-23 also shows some tests pertaining to this scenario:

- An inbound ping from 10.10.10.6 to 10.10.10.10 was blocked because there is no policy enabling connection initiation from the OUTSIDE zone to the INSIDE.
- An outbound ping from 10.10.10.10 to 10.10.10.6 is permitted.
- An outbound NTP session from client 10.10.10.10 to server 10.10.10.6 is enabled through the ZFW.

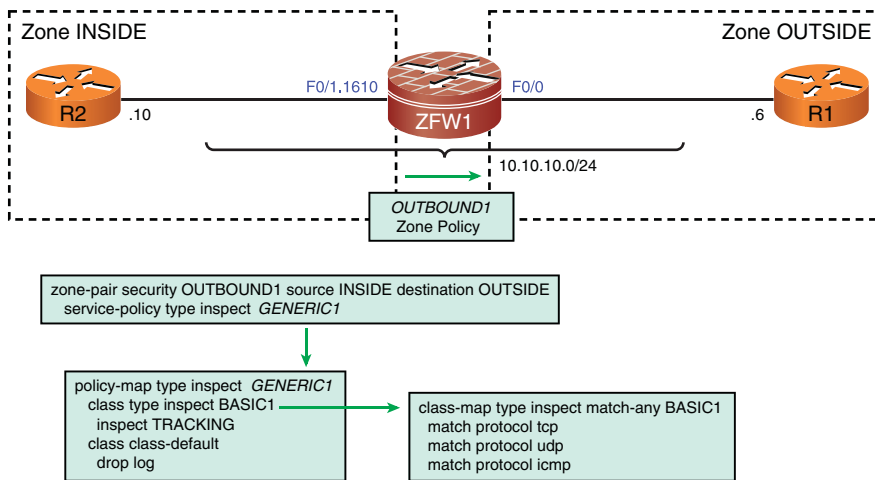


Figure 10-11 Reference Topology for the Analysis of ZFW in Transparent Mode

Example 10-23 Bridging Between ZFW Security Zones

```
! Basic configuration for bridging between zones
bridge irb
bridge 1 protocol ieee
bridge 1 route ip
!
interface FastEthernet0/1.1610
 encapsulation dot1Q 1610
 no ip address
 zone-member security INSIDE
bridge-group 1
!
interface FastEthernet0/0
 no ip address
 zone-member security OUTSIDE
bridge-group 1
!
interface BVI1
 ip address 10.10.10.2 255.255.255.0
!
ZFW1# show zone-pair security
Zone-pair name OUTBOUND1
    Source-Zone INSIDE Destination-Zone OUTSIDE
    service-policy GENERIC1
!
! Inbound ping from 10.10.10.6 to 10.10.10.10 (not allowed)

%FW-6-DROP_PKT: Dropping icmp session 10.10.10.6:0 10.10.10.10:0 due
to policy match failure with ip ident 0
```

```

!
! Outbound ping from 10.10.10.10 to 10.10.10.6

%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:BASIC1):Start
icmp session: initiator (10.10.10.10:0) — responder (10.10.10.6:0)
!
ZFW1# show policy-map type inspect zone-pair OUTBOUND1 sessions ! include Session
      Number of Established Sessions = 1
      Established Sessions
        Session 48863720 (10.10.10.10:8)=>(10.10.10.6:0) icmp SIS_OPEN
!
! Outbound NTP session (server 10.10.10.6 is on the OUTSIDE zone)

FIREWALL* sis 48863120: Session Created
FIREWALL* sis 48863120: Pak 47CE3220 init_addr (10.10.10.10:123)
resp_addr (10.10.10.6:123)
init_alt_addr (10.10.10.10:123) resp_alt_addr (10.10.10.6:123)
FIREWALL* sis 48863120: FO cls 0x4769BB60 clsgroup 0x10000000, target
0xA0000000, FO 0x485ED680, alert = 1, audit_trail = 1, L7 = Unknown-
17, PAMID = 5
FIREWALL* sis 48863120: Allocating L7 sis extension L4 = udp, L7= Unknown-17,
PAMID = 5
!
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:BASIC1):Start
udp session: initiator (10.10.10.10:123) — responder
(10.10.10.6:123)

ZFW1# show policy-map type inspect zone-pair OUTBOUND1 sessions ! include Session
      Number of Established Sessions = 1
      Established Sessions
        Session 48863120 (10.10.10.10:123)=>(10.10.10.6:123) udp SIS_OPEN

```

Figure 10-12 proposes an application scenario for ZFW in Transparent mode. The figure shows the topology and the corresponding policy scheme for a network in which there is a demand for *conditional bridging* between a wired and a wireless network. Some possible motivations for this type of arrangement follow:

- Simpler deployment without compromising security. Transparent mode enables quick insertion of new hosts with no need of any network renumbering. Further, the server resources (typically on the wired zone) can be protected from hosts requesting access from the wireless side.
- If, for example, this is a remote branch for a certain company, it will be easier to manage DHCP scopes in a centralized way. No matter if connected to the wired or wireless network, a host located in a given branch can receive an IP address belonging to a unique pool.

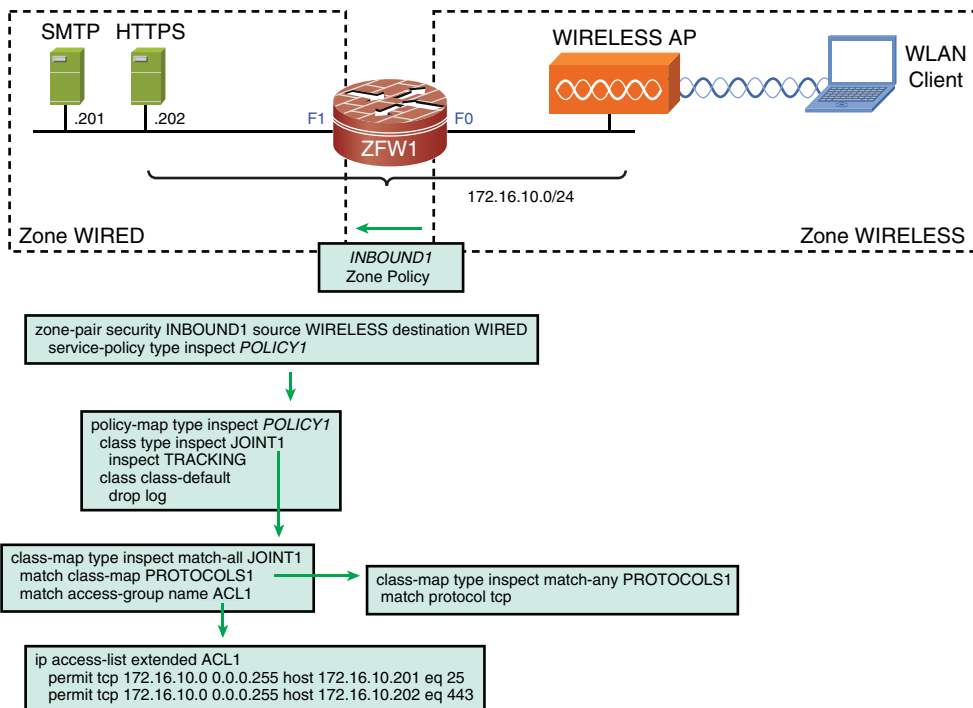


Figure 10-12 Important Application Scenario for ZFW in Transparent Mode

Note If you need more details about Transparent mode connectivity for Cisco IOS routers, refer to Chapter 5, “Firewalls in the Network Topology.”

Defining Connection Limits

This section characterizes how the ZFW lends itself to the challenging task of mitigating DoS attacks. The **parameter-map**, a ZFW entity that has been used so far solely for turning **audit-trail** on, accomplishes this.

Example 10-24 relates to the topology in Figure 10-13 and establishes the acceptable rate of a new half-open connection setup (within a 1-minute interval), that can cause the ZFW system to start deleting half-open sessions. More precisely, for this particular example

- **one-minute high 50** means that the ZFW accepts a rate of 50 half-open sessions per minute before it starts deleting them.
- **one-minute low 40** determines that the ZFW stops deleting embryonic sessions if the measured rate falls below 40 connections/minute.
- The measurements are performed every minute, and whenever a threshold is crossed (either the high or the low one), an ALERT message is issued.

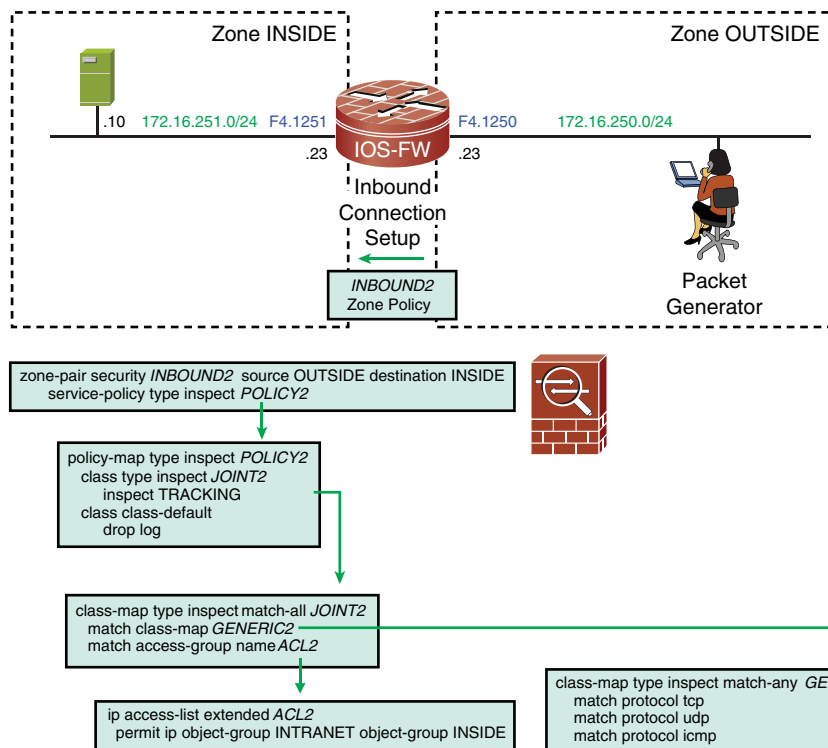


Figure 10-13 Reference Topology for the Study of Connection Limits

Example 10-24 Setting Acceptable Connection Rates

```

! Assigning interfaces to zones
interface FastEthernet4.1250
ip address 172.16.250.23 255.255.255.0
ip flow ingress
zone-member security OUTSIDE
!
interface FastEthernet4.1251
ip address 172.16.251.23 255.255.255.0
ip flow egress
zone-member security INSIDE
!
! Defining acceptable connection rates (per minute) inside the parameter-map
parameter-map type inspect TRACKING
audit-trail on
one-minute low 40
one-minute high 50
!
  
```

```

! Network object-groups used in the ACL called 'ACL2'
object-group network INSIDE
172.16.251.0 255.255.255.0
 172.16.252.0 255.255.255.0
!
object-group network INTRANET
 172.16.0.0 255.240.0.0
!
! ACL and class-maps to allow inspection between selected source/destination pairs
ip access-list extended ACL2
 permit ip object-group INTRANET object-group INSIDE
!
class-map type inspect match-any GENERIC2
 match protocol tcp
 match protocol udp
 match protocol icmp
!
class-map type inspect match-all JOINT2
 match class-map GENERIC2
 match access-group name ACL2
!
! Defining and applying an inbound Zone-pair security policy
policy-map type inspect POLICY2
 class type inspect JOINT2
   inspect TRACKING
 class class-default
   drop log
!
zone-pair security INBOUND2 source OUTSIDE destination INSIDE
 service-policy type inspect POLICY2
!
! Sequence of telnet sessions generated to host 172.16.251.10
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(INBOUND2:JOINT2):
Start telnet session: initiator (172.16.250.145:33000) — responder
(172.16.251.10:23)
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(INBOUND2:JOINT2):
Start telnet session: initiator (172.16.250.146:33001) — responder
(172.16.251.10:23)
[ output suppressed ]
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(INBOUND2:JOINT2):
Start telnet session: initiator (172.16.250.150:33049) — responder
(172.16.251.10:23)
%FW-6-SESS_AUDIT_TRAIL: (target:class)-(INBOUND2:JOINT2):Stop telnet
session: initiator (172.16.250.151:33050) sent 0 bytes -responder
(172.16.251.10:23) sent 0 bytes

```



```

!
! One minute rate is measured and connection admission is policed
%FW-4-ALERT_ON: (target:class)-(INBOUND2:JOINT2):getting aggressive,
count (51/2147483647) current 1-min rate: 51
!
! Decrease in connection rate is detected by the firewall
%FW-4-ALERT_OFF: (target:class)-(INBOUND2:JOINT2):calming down,
count (0/2147483647) current 1-min rate: 19

```

One significant distinction between CBAC and ZFW anti-DoS capabilities resides in that while the former relies on global settings, the latter enables the specification of limits on a per-class basis. For instance, in Example 10-24, the rate of sessions admission is applied to the class JOINT2 (under the **policy-map** POLICY2).

In Example 10-25, absolute limits for the number of embryonic sessions to a given host are defined in the **parameter-map** TRACKING (which is later tied to the **inspect** action for class JOINT2, from Example 10-24). The examples show the ALERT messages issued when a *rising* or *falling* threshold is crossed.

Example 10-25 *Setting a Limit for the Embryonic Connections*

```

! Defining the acceptable number of Incomplete Connections
parameter-map type inspect TRACKING
  max-incomplete low 800
  max-incomplete high 1000
!
! Number of accumulated half-open sessions crosses the threshold of 1000
%FW-4-ALERT_ON: (target:class)-(INBOUND2:JOINT2):getting aggressive,
count (1001/1000) current 1-min rate: 1134
!
! Number of half-open sessions falls below the lower limit of 800
%FW-4-ALERT_OFF: (target:class)-(INBOUND2:JOINT2):calming down,
count (799/800) current 1-min rate: 1800

```

Note The settings for the *default* **parameter-map** were documented in Example 10-2. Among other possibilities, **parameter-maps** belonging to the **type inspect** category, also enable the specification of connection timers for each protocol and an absolute limit for the number of established sessions (refer to the **sessions maximum** option).

Inspection of Router Traffic

Up to now, this chapter examined connections crossing the ZFW. This last section shifts gears a little bit and focuses on traffic directed to the ZFW router.

The default operation of the ZFW is to permit any traffic to and from the router interfaces, regardless of zone membership and **zone-pair security** settings. If there is a need to change this initial behavior, a system-defined zone, whose reserved name is *self*, must be invoked.

Although interfaces are not pointed as members of the *self* zone, this special zone handles packets traveling from any router interface to a *nonself* zone (and conversely). You should keep in mind that policies including the *self* zone are still unidirectional.

Figure 10-14 shows a reference topology in which the **zone-pair security** OUT-SELF was created to deal with traffic sourced from the OUTSIDE zone and destined to the *self* zone. The detailed configurations commands are listed in Example 10-26. For this particular example, there is no **zone-pair** policy designed to control connections initiating in any of the router addresses and destined to the OUTSIDE zone.

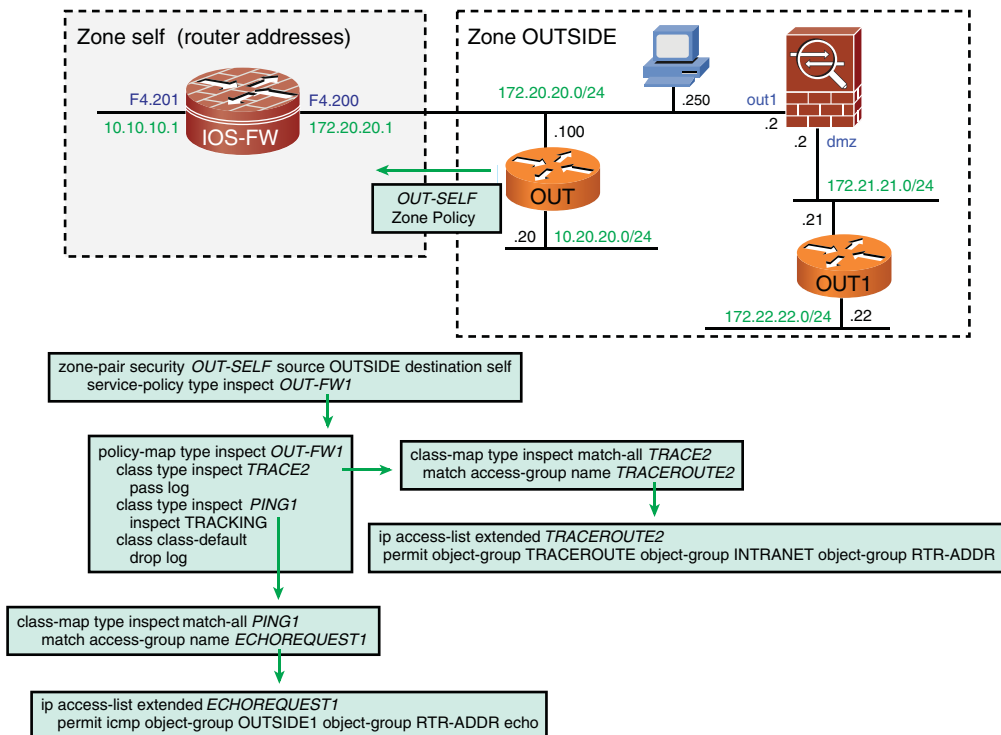


Figure 10-14 Sample Inspection of Traffic Directed to Addresses of the ZFW Router

The design goals for the scenario of Example 10-26 follow:

- Permit **traceroute** from the router interfaces to the subnets contained in the **IN-TRANET object-group**. This is accomplished by assigning the **pass** action to class **TRACE2** under the **policy-map OUT-FW1**. This class handles the relevant ICMP messages issued by L3 hops when tracing activities are being executed.
- Inspect ICMP messages sourced from addresses described by the **OUTSIDE1 object-group** and directed to router interfaces. Class **PING1** (under **policy-map OUT-FW1**) takes care of this demand.

Some tests are performed in the context of Example 10-26 with the aim to reveal how this configuration behaves:

- A ping session from address 172.20.20.100 toward the router address 172.20.20.1 is inspected by class **PING1**.
- The **traceroute** from 10.10.10.1 to 172.22.22.22 succeeds because of the **pass** action implemented by class **TRACE2**.
- A ping from the **OUT1** router (172.21.21.21) is dropped because it does not comply with the rules specified by **access-list ECHOREQUEST1** (concerning source addresses).
- A telnet from the **OUT** router (172.20.20.100) to 172.20.20.1 is blocked because this protocol is deemed prohibited (from the perspective of class **PING1**).

Example 10-26 *Sample Policy to Inspect Router Traffic*

```
! Network object-groups used in the configuration
object-group network INTRANET
 172.16.0.0 255.240.0.0
!
object-group network RTR-ADDR
 host 10.10.10.1
 host 172.20.20.1
!
object-group network OUTSIDE1
 172.20.20.0 255.255.255.0
!
! Allowing traceroute tasks started from the interface addresses of the IOS-ZFW
router
ip access-list extended TRACEROUTE2
 permit object-group TRACEROUTE object-group INTRANET object-group RTR-ADDR
!
class-map type inspect match-all TRACE2
 match access-group name TRACEROUTE2
!
```

```

! Allowing pings from the OUTSIDE1 subnets to the interface addresses of router
IOS-ZFW

ip access-list extended ECHOREQUEST1
  permit icmp object-group OUTSIDE1 object-group RTR-ADDR echo
!
class-map type inspect match-all PING1
  match access-group name ECHOREQUEST1
!
! Defining and applying a Zone-pair security policy (from the OUTSIDE to the
router)
policy-map type inspect OUT-FW1
  class type inspect TRACE2
    pass log
  class type inspect PING1
    inspect TRACKING
class class-default
  drop log
!
zone-pair security OUT-SELF source OUTSIDE destination self
  service-policy type inspect OUT-FW1
!
! Ping from 172.20.20.100 (OUT router) to the interface address 172.20.20.1
OUT# ping 172.20.20.1 source 172.20.20.100 repeat 2
Success rate is 100 percent (2/2), round-trip min/avg/max = 4/4/4 ms
!
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUT-SELF:PING1):Start icmp session:
  initiator (172.20.20.100:0) — responder (172.20.20.1:0)
!
IOS-ZFW# show ip cache flow | begin Src
SrcIf      SrcIPaddress  DstIf      DstIPaddress  Pr SrcP DstP  Pkts
Fa4.200    172.20.20.100    Local      172.20.20.1   01 0000 0800    2
!
! Traceroute from the router address 10.10.10.1 to host 172.22.22.22
IOS-ZFW# traceroute 172.22.22.22 source 10.10.10.1 probe 1
Tracing the route to 172.22.22.22
  1 172.21.21.2 1252 msec
  2 172.21.21.21 0 msec
%FW-6-PASS_PKT: (target:class)-(OUT-SELF:TRACE2) Passing icmp pkt 172.21.21.2:0 =>
10.10.10.1:0 with ip ident 0
ICMP: time exceeded rcvd from 172.21.21.2
ICMP: dst (10.10.10.1) port unreachable rcv from 172.21.21.21
!
! Ping from the OUT1 router (172.21.21.21) to the interface address 172.20.20.1
OUT1# ping 172.20.20.1 source 172.21.21.21 repeat 1

```

```

Success rate is 0 percent (0/1)
!
%FW-6-DROP_PKT: Dropping icmp session 172.21.21.21:0 172.20.20.1:0 on zone-pair
OUT-SELF class class-default due to DROP action found in policy-map with ip
ident 0
%FW-6-LOG_SUMMARY: 1 packet were dropped from 172.21.21.21:8 => 172.20.20.1:0
(target:class)-(OUT-SELF:class-default)
!
! Telnet from the OUT router (172.20.20.100) to the interface address 172.20.20.1
OUT# telnet 172.20.20.1
Trying 172.20.20.1 ...
% Connection timed out; remote host not responding
!
%FW-6-DROP_PKT: Dropping Other session 172.20.20.100:64515
172.20.20.1:23 on zone-pair OUT-SELF class class-default due to DROP
action found in policy-map with ip ident 0
%FW-6-LOG_SUMMARY: 4 packets were dropped from 172.20.20.100:64515 =>
172.20.20.1:23 (target:class)-(OUT-SELF:class-default)

```

This was a just a quick look at the inspection capabilities for router traffic, within the realm of the ZFW. At this point, you are strongly encouraged to investigate additional policies that involve the special zone called *self*.

Intrazone Firewall Policies in IOS 15.X

All the scenarios studied so far considered the use of Cisco IOS releases belonging to the 12.4T train, which are characterized by an *implicit permit* for traffic flowing between two interfaces part of the same security zone.

Starting on release 15.0(1)M, IOS behaves in a way diametrically opposite to its predecessors: Any traffic between interfaces in the same zone is blocked by default. And if there is any interest in modifying this operation, you need to define an intrazone ZFW policy.

The topology represented in Figure 10-15 is used as a reference for the study of intrazone policies. In this figure, interfaces F0/1.1610 and F0/0 are members of the INSIDE zone and, initially, no traffic is enabled between them in any direction. Example 10-27 refers to this scenario and further details some important intrazone features:

- The *implicit deny* behavior of IOS 15.X is illustrated for one ICMP and one UDP connection attempts between hosts 10.10.6.6 and 10.10.10.10.
- After the ZFW policy called INTRAZONE1 is defined, connection initiation from any of the interfaces in the INSIDE zone is enabled. This is a way of demonstrating that intrazone firewall policies are *bidirectional* in nature. To gain more control for flow setup, ACLs need to be used (as part of the **class-map** definition in the ZFW structure).

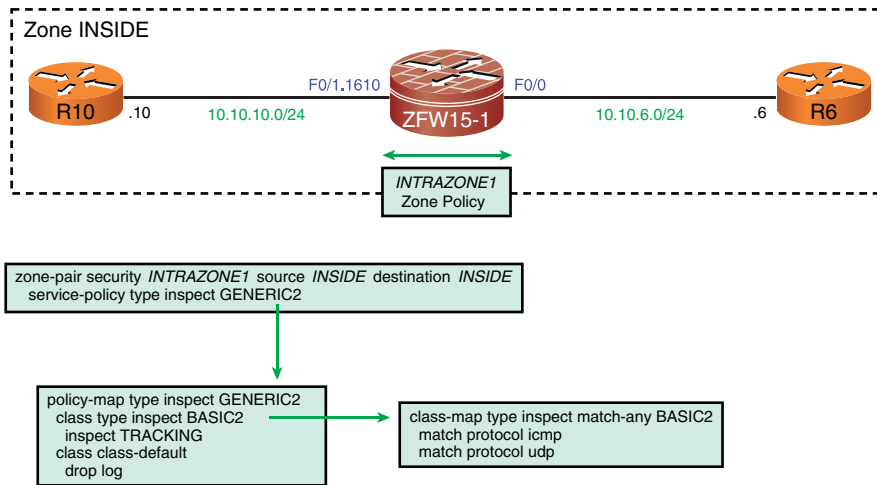


Figure 10-15 Reference Topology for the Analysis of ZFW Intrazone Policies

Example 10-28 also refers to Figure 10-15 and documents the usage of the **show policy-firewall** commands to reveal information about ZFW active sessions, configuration, and statistics in IOS 15.X. This new set of commands is simpler than the **show policy-map type inspect** options available on pre-15.0 IOS releases. If you already have an IOS 15.X installed in your ZFW router, repeating the examples contained in this chapter with the newer **show** commands might be an instructive exercise.

Example 10-27 ZFW and Intrazone Policies

```
! Basic information about security zones
ZFW15-1# show zone security
zone self
  Description: System defined zone
zone INSIDE
  Member Interfaces:
    FastEthernet0/1.1610
    FastEthernet0/0
!
! Default behavior is to block traffic between interfaces in the same zone

%FW-6-DROP_PKT: Dropping icmp session 10.10.6.6:0 10.10.10.10:0 due to policy
match failure with ip ident 0
%FW-6-DROP_PKT: Dropping udp session 10.10.6.6:123 10.10.10.10:123 due to policy
match failure with ip ident 0
!
! Defining a zone-pair to control intrazone traffic

ZFW15-1# show zone-pair security
```

```

Zone-pair name INTRAZONE1
    Source-Zone INSIDE Destination-Zone INSIDE
    service-policy GENERIC2
!
! Sample intrazone ping from 10.10.6.6 to 10.10.10.10

FIREWALL* sis 49CFAEC0: Session Created
FIREWALL* sis 49CFAEC0: Pak 4936A924 init_addr (10.10.6.6:8)
resp_addr (10.10.10.10:0) init_alt_addr (10.10.6.6:8) resp_alt_addr
(10.10.10.10:0)
FIREWALL* sis 49CFAEC0: FO cls 0x4B1EAA00 clsgrp 0x10000000, target
0xA0000000, FO 0x49EDCC00, alert = 1, audit_trail = 1, L7 = Unknown-
17, PAMID = 0

%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(INTRAZONE1:BASIC2):Start icmp ses-
sion: initiator (10.10.6.6:8) —
responder (10.10.10.10:0)

! Sample intrazone ping from 10.10.10.10 to 10.10.6.6

%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-
(INTRAZONE1:BASIC2):Start icmp session: initiator (10.10.10.10:8) —
responder (10.10.6.6:0)

```

Example 10-28 Obtaining Information About the Firewall Policy

```

! Sample UDP session from 10.10.6.6 (NTP Client) to 10.10.10.10
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-
(INTRAZONE1:BASIC2):Start udp session: initiator (10.10.6.6:123) —
responder (10.10.10.10:123)
!
! Obtaining information about active firewall sessions in IOS 15.X

ZFW15-1# show policy-firewall session
    Established Sessions = 1
        Session 49CFB240 (10.10.6.6:123)=>(10.10.10.10:123) udp SIS_OPEN
        Created 00:00:26, Last heard 00:00:26
        Bytes sent (initiator:responder) [48:48]
!
! Displaying information about ZFW configuration in IOS 15.X

ZFW15-1# show policy-firewall config zone-pair
Zone-pair                : INTRAZONE1
Source Zone               : INSIDE
Destination Zone          : INSIDE

```

```

Service-policy inspect : GENERIC2
  Class-map : BASIC2(match-any)
    Match protocol icmp
    Match protocol udp
  Action : inspect
    Parameter-map : TRACKING
  Class-map : class-default(match-any)
    Match any
  Action : drop log
    Parameter-map : Default
!
ZFW15-1# show policy-firewall config policy-map
Policy Map type inspect GENERIC2
  Class BASIC2
    Inspect TRACKING
  Class class-default
    Drop log
!
ZFW15-1# show policy-firewall config class-map
Class Map type inspect match-any BASIC2 (id 1)
  Match protocol icmp
  Match protocol udp
!
! Information about ZFW statistics in IOS 15.X

ZFW15-1# show policy-firewall stats
Global Stats:
  Packet inspection statistics [process switch:fast switch]
  udp packets: [0:24]
  icmp packets: [0:10]
  Session creations since subsystem startup or last reset 13
  Current session counts (estab/half-open/terminating) [0:0:0]
  Maxever session counts (estab/half-open/terminating) [1:1:0]
  Last session created 00:00:53
  Last statistic reset never
  Last session creation rate 1
  Maxever session creation rate 1
  Last half-open session total 0

```


Summary

This chapter presented the main concepts that pertain to the *Zone-based Policy Firewall* (ZFW). It showed how this structured approach for firewall policy creation on IOS can significantly facilitate design and operations.

Having studied this chapter, you can now do the following:

- Describe how the ZFW philosophy differs from CBAC's policy.
- Confidently employ the *Cisco Policy Language (CPL)* to build structured Zone-based Firewall Policies. The CPL equally applies to topologies that use either routed or transparent mode.
- Apply ZFW to build L4 Firewall policies (generic inspection of TCP, UDP, and ICMP). Higher layer policies are studied in Chapter 12.
- Understand how to integrate ACL filtering capabilities within a ZFW design, without disrupting operations.
- Describe how the ZFW interacts with Network Address Translation.
- Define connection limits within ZFW inspection policies to contribute for DoS attack mitigation.
- Understand how router traffic is handled by the ZFW.

Additional Protection Mechanisms

This chapter covers the following topics:

- Antispoofing
- TCP Flags Filtering on IOS
- Filtering on the TTL value on IOS
- Handling IP Options
- Dealing with IP Fragmentation
- Flexible Packet Matching
- Time-based ACLs
- Connection Limits on ASA
- TCP Normalization on ASA
- Threat Detection on ASA

“There are more things in heaven and earth, Horatio, Than are dreamt of in your philosophy.”—William Shakespeare, Hamlet

Having thoroughly studied stateful inspection, it is now time to devote some attention to a set of additional protection mechanisms that operate on Layers 3 and 4 and might prove useful to many network environments. Although most of them are stateless features, they have the potential to greatly contribute to network security, mainly with the underlying goal of building better designs.

This chapter starts with the analysis of antispoofing techniques and then focuses on special filtering resources designed to identify, among other parameters, the particular value carried in the TCP flags and IP TTL fields or the type of IP options (in case they are present) in the IP Datagram Header.

Following this discussion, the operation of IP fragmentation is investigated and some means of mitigating the risks associated with this typical attack vector are proposed. The filtering efforts culminate on the analysis of the Flexible Packet Matching (FPM) IOS functionality and time-based Access Control Lists (ACL).

The chapter ends with the study of TCP normalization, connection limiting, and threat detection, which correspond to resources available in the Adaptive Security Appliances (ASA).

Antispoofing

IP spoofing is an action through which a potential intruder copies or falsifies a trusted source IP address. This is typically employed as an auxiliary technique for countless types of network-based attacks. Some of the classic motivations behind IP spoofing practice follow:

- Impersonate some trusted host or user and take profit of the privileges that arise from this trust relationship.
- Divert attention away from the real originator of the attack in an attempt to remain undetected.
- Cast suspicion on legitimate hosts.

Among the methods that materialize IP spoofing, some deserve special mention:

- Simply creating packets with falsified source addresses using packet generation tools.
- Injecting malicious commands or data into some existent stream of data.
- Influencing the modification of routing tables so that packets get directed to the spoofed address. This can be done using host-oriented protocols such as DHCP and ARP or even routing protocols in cases in which the peers do not employ authentication. (For an analysis of routing protocol authentication, refer to Chapter 5, “Firewalls in the Network Topology.”)

Having characterized the evil that can derive from IP spoofing, it becomes notable that investments must be made to combat it. This section covers two important antispoofing mechanisms that have been available for a while on firewall devices.

Classic Antispoofing Using ACLs

This method represents what is described in RFC 2827 (BCP 38) as best current practices to defeat IP source address spoofing. This is a widespread resource, particularly for the Internet edge because on such an environment, the boundary between private and public addresses (in the sense of RFC 1918) is clearly demarcated.

Figure 11-1 portrays a simplified Internet access topology for a customer whose block of assigned IP addresses is symbolized by X.Y.Z.0/24, meaning that this particular group of

valid addresses should exist exclusively in this customer's premises. In this scenario, the Cust-GW and ISP routers play complementary roles:

- ACL 140 is designed to ensure that incoming packets arriving on interface *serial n* of the Cust-GW router do not use a source address belonging to the X.Y.Z.0/24 range. Addresses in this block can be seen only as the destination of the packet.
- ACL 150 is conceived to guarantee that only those source addresses that are part of the block X.Y.Z.0/24 leave the Cust-GW router (through *serial n*) toward the Internet.

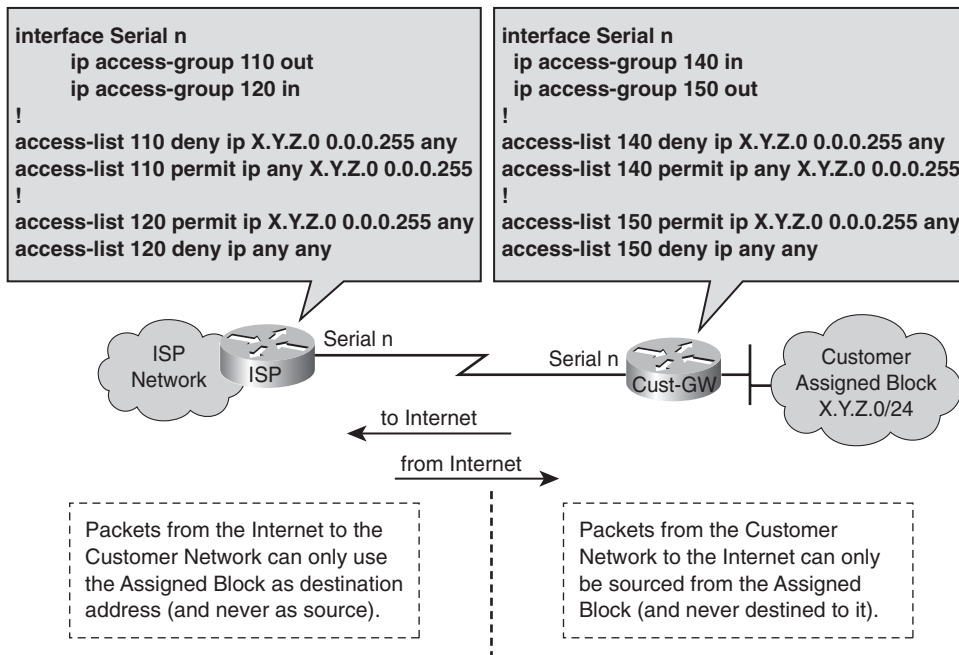


Figure 11-1 Antispoofing Using ACLs

ACL 120 mirrors the settings from ACL 150 on the ISP router, and likewise, ACL 110 corresponds to the customer's ACL 140. The idea is to enforce the best practices security measures and avoid blindly relying on the customer's initiative for the implementation of these mitigation techniques. If all the companies connected to the Internet and the corresponding ISPs that provide them with access were committed to doing their respective homework, they would significantly contribute to eliminating IP spoofing from the Internet.

Antispoofing with uRPF on IOS

The unicast Reverse Path Forwarding (uRPF) verification is a feature that leverages the contents of the IP Forwarding table on Cisco routers (CEF table) to mitigate source address spoofing. Basically, when this ingenious mechanism is enabled, packets arriving

on an interface with source addresses that are not reachable using the existent forwarding table are discarded. IOS provides two ways to implement uRPF:

- **Strict uRPF:** In this mode, IOS verifies if the IP source addresses of the received packets are reachable via the specific router interface on which they arrived. Packets that do not pass this test are discarded.
- **Loose uRPF:** In this mode, the only demand on the source IP address is to be reachable through the CEF table using any of the available interfaces. Packets that do not exist on the CEF table are dropped.

Figure 11-2 depicts a reference network that was conceived to illustrate the behavior of the IOS uRPF functionality. This figure also brings the IP routing tables on the R1 router and the ASA appliance and is directly related in Examples 11-1 and 11-2.

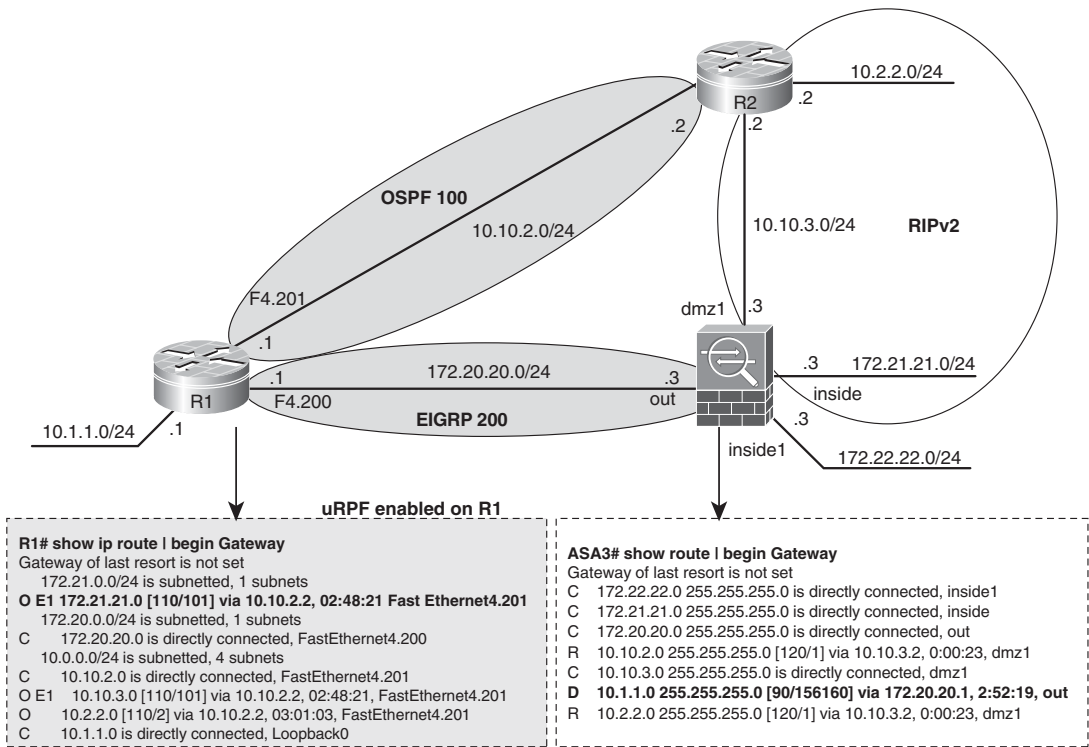


Figure 11-2 Reference Topology for uRPF Analysis on IOS

Example 11-1 refers to the scenario of Figure 11-2 and documents the IOS strict uRPF feature. In this example, the command `ip verify unicast source reachable-via rx` is enabled on interface f4.200, meaning that the source addresses contained on packets

received on this particular interface must be reachable through it. It is important to observe that

- From the perspective of R1's routing table, network 172.21.21.0/24 (ASA *inside*) is accessible using R2's address 10.10.2.2 as its next hop (via interface f4.201).
- From ASA's standpoint, the remote network 10.1.1.0/24 is reachable via interface *out* using 172.20.20.1 (R1) as the next hop.

When the host 172.21.21.5, which uses ASA's *inside* address as default gateway, tries to reach 10.1.1.1, the strict uRPF check on R1 drops the packet. This happens because the packet arrives through f4.200 and, as per R1 knowledge, should have arrived via f4.201.

Example 11-1 *Strict uRPF on IOS*

```
! Enabling strict uRPF verification on interface f4.200
interface FastEthernet4.200
 encapsulation dot1Q 200
 ip address 172.20.20.1 255.255.255.0
 ip verify unicast source reachable-via rx
!
R1#show cef interface f4.200 | include RPF
  IP unicast RPF check is enabled
  Input features: uRPF
!
! Verifying uRPF drops
R1# CEF-Drop: Packet from 172.21.21.5 (Fa4.200) to 10.1.1.1, uRPF feature
!
R1# show ip interface f4.200 | include verif
  IP verify source reachable-via RX
  36 verification drops
  0 suppressed verification drops
  0 verification drop-rate
!
R1# show ip traffic | include RPF
  0 no route, 36 unicast RPF, 0 forced drop
```

Example 11-2 also builds upon the topology of Figure 11-2. In this case, the command **ip verify unicast source reachable-via any** was employed to materialize the loose uRPF check, which requires only the source address contained in the received packet to be accessible through any of the router's interfaces. There is no need anymore for the source address to be reachable via exactly the same interface where it arrived. This lighter demand translates into the possibility of using asymmetric routing, which was not feasible using strict uRPF.

In Example 11-2, the host 172.22.22.5 uses ASA's *inside1* interface address as its default gateway and sends a packet to 10.1.1.1, a destination that resides on R1's LAN. The important detail here is that 172.22.22.0/24 is not part of any routing process on ASA and, as such, is not announced to neighbor routers. Given that this source network does not appear on R1's routing table (neither dynamically nor statically), when a packet belonging to it arrives on f4.200, the loose uRPF test fails and the packet is dropped.

Example 11-2 Loose uRPF on IOS

```
interface FastEthernet4.200
  encapsulation dot1Q 200
  ip address 172.20.20.1 255.255.255.0
  ip verify unicast source reachable-via any
!
R1# show ip interface f4.200 | include verif
IP verify source reachable-via ANY
49 verification drops
4 suppressed verification drops
0 verification drop-rate
!
! Host 172.22.22.5 pings 10.1.1.1, using 172.22.22.3 (ASA3) as gateway

R1# CEF-Drop: Packet from 172.22.22.5 (Fa4.200) to 10.1.1.1, uRPF feature
!
R1# show ip traffic | include RPF
0 no route, 4 unicast RPF, 0 forced drop
```

Note The uRPF verification feature works even when there is a configured default route. For instance, if R1 had a default route pointing to 172.20.20.3 (ASA), a packet from 172.22.22.5 to 10.1.1.1 would still be discarded by the loose uRPF check. If there is the interest (or need) in considering the default route a valid option for the uRPF check, the `ip verify unicast source reachable-via any allow-default` command should be used.

Antispoofing with uRPF on ASA

Before studying how ASA employs uRPF, it is convenient to characterize how it deals with asymmetric traffic.

Example 11-3 relates to the topology depicted in Figure 11-2. In this scenario, a packet from host 172.21.21.5 (connected to ASA *inside* interface) is sent to destination 10.1.1.1 via *out* interface. The echo-reply is sent back by host 10.1.1.1 via interface f4.201 on R1 (refer to R1's routing table in Figure 11-2). The packet is dropped because ASA expects the reply to arrive on interface *out* and there is no specific permission for echo-replies on *dmz1*, the interface through which the packet is actually delivered to ASA.

Example 11-3 shows that defining an Access Control Entry (ACE) that enables echo-replies on interface *dmz1* solves the problem.

Example 11-3 ASA and Asymmetric routing (1)

```
! Echo request from 172.21.21.5 (inside) to 10.1.1.1 (out) is allowed
%ASA-6-302020: Built outbound ICMP connection for faddr 10.1.1.1/0 gaddr
172.21.21.5/768 laddr 172.21.21.5/768
ICMP echo request from inside:172.21.21.5 to out:10.1.1.1 ID=768 seq=26112 len=32
!
! Echo reply arrives on interface dmz1
%ASA-3-106014: Deny inbound icmp src dmz1:10.1.1.1 dst inside:172.21.21.5
(type 0, code 0)
!
ASA3# show asp drop
Frame drop:
  No route to host (no-route)                                4
  Flow is denied by configured rule (acl-drop)                1
!
! Permission for echo-replies is created on interface dmz1

access-list DMZ1 extended permit icmp 10.1.1.0 255.255.255.0 172.21.21.0
255.255.255.0 echo-reply log interval 120
!
! Echo request from 172.21.21.5 (inside) to 10.1.1.1 (out) is allowed
ICMP echo request from inside:172.21.21.5 to out:10.1.1.1 ID=768 seq=28160 len=32
!
! Echo reply is now allowed by the appropriate ACL entry
%ASA-6-106100: access-list DMZ1 permitted icmp dmz1/10.1.1.1(0) ->
inside/172.21.21.5(0) hit-cnt 1 first hit [0xfe149ab1, 0x0]
%ASA-6-302020: Built inbound ICMP connection for faddr 10.1.1.1/0 gaddr
172.21.21.5/768 laddr 172.21.21.5/768
ICMP echo reply from dmz1:10.1.1.1 to inside:172.21.21.5 ID=768 seq=28160 len=32
```

Example 11-4 illustrates a situation in which ICMP inspection is globally enabled on ASA and the permission for echo-replies is already configured on interface *dmz1*. Now ICMP is deemed a stateful protocol, and ASA interprets the arrival of the reply packet on a distinct interface as a security violation.

Example 11-5 registers a scenario in which ICMP inspection is avoided for traffic that is known to produce asymmetric routing. To achieve this, a **class-map** that matches on source/destination IP address combinations is employed inside the *global_policy* policy-map.

Example 11-4 *ASA and Asymmetric routing (2)*

```

! Echo request from 172.21.21.5 (inside) to 10.1.1.1 (out) is allowed
ICMP echo request from inside:172.21.21.5 to out:10.1.1.1 ID=768 seq=28416 len=32
!
! The ACL 'DMZ1' permits the echo reply arriving on interface dmz1
%ASA-6-106100: access-list DMZ1 permitted icmp dmz1/10.1.1.1(0) ->
inside/172.21.21.5(0) hit-cnt 1 first hit [0xfe149ab1, 0x0]
%ASA-6-302020: Built inbound ICMP connection for faddr 10.1.1.1/0 gaddr
172.21.21.5/768 laddr 172.21.21.5/768
!
! But ICMP inspection denies it (there is no matching session between inside and
dmz1)
%ASA-4-313004: Denied ICMP type=0, from laddr 10.1.1.1 on interface dmz1 to
172.21.21.5: no matching session
!
ASA3# show asp drop
Frame drop:
    ICMP Inspect seq num not matched (inspect-icmp-seq-num-not-matched)          1

```

Example 11-5 *ASA and Asymmetric routing (3)*

```

! Avoiding ICMP inspection for src/dest pairs that result in asymmetric routing
access-list ICMP1 extended deny icmp 172.21.21.0 255.255.255.0 10.1.1.0
255.255.255.0
access-list ICMP1 extended permit icmp 172.21.21.0 255.255.255.0 any
!
class-map CLASS1
  match access-list ICMP1
!
policy-map global_policy
class CLASS1
  inspect icmp
!
service-policy global_policy global
!
! Verifying the service-policy exceptions in action
ASA3# show service-policy flow icmp host 172.21.21.5 host 10.10.10.1 echo
Global policy:
  Service-policy: global_policy
    Class-map: CLASS1
      Match: access-list ICMP1
        Access rule: permit icmp 172.21.21.0 255.255.255.0 any

```

```

Action:
  Input flow: inspect icmp
Class-map: class-default
Match: any
Action:

```

After briefly reviewing ASA's handling of asymmetric traffic, it is time to turn your attention to the uRPF feature. Figure 11-3 documents the reference topology for this analysis.

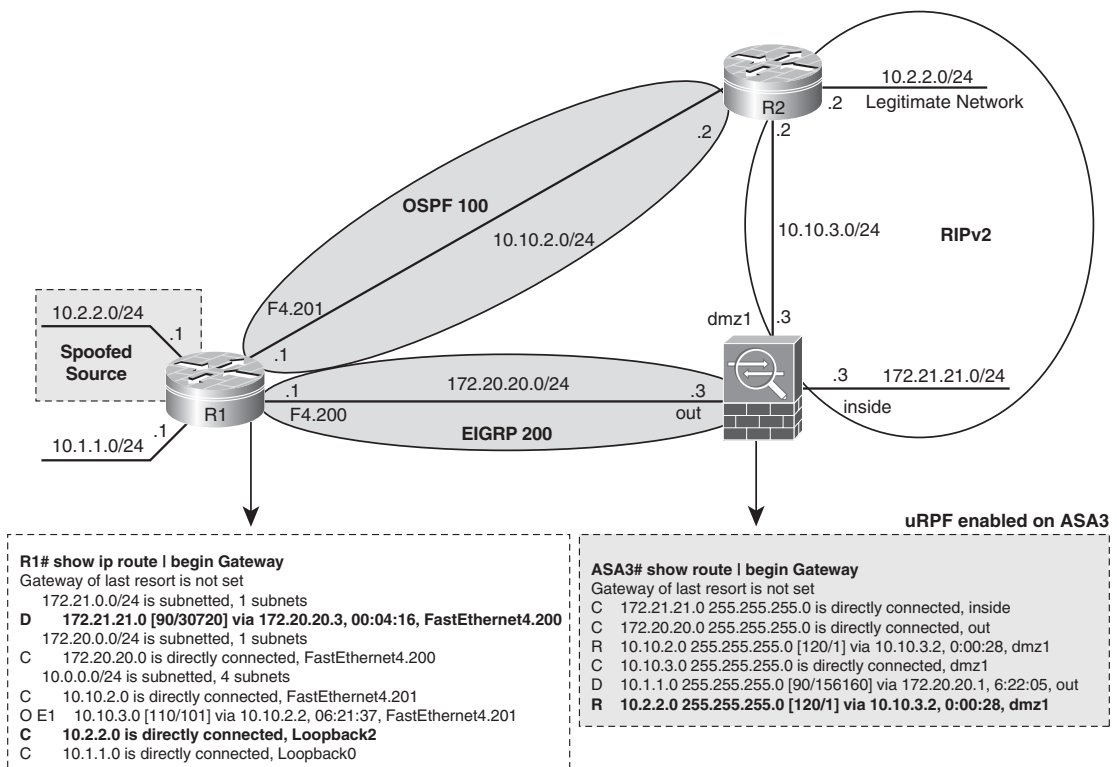


Figure 11-3 Reference Topology for uRPF Analysis on ASA

Example 11-6 is directly related to Figure 11-3. A host behind R1 (10.2.2.1) emulates a spoofed source. As per ASA3's routing table, the network 10.2.2.0/24 is reachable via interface *dmz1*, using R2 as gateway. When ASA sees an echo request packet from 10.2.2.1 arriving on interface *out*, the uRPF check (which is enabled for *out*) comes to the scene and denies access. The example also shows how the Packet Tracer tool can help characterize a uRPF violation.

Example 11-6 *uRPF on ASA*

```

! Enabling anti-spoofing (uRPF check) on interface 'out'
ip verify reverse-path interface out
!
! Emulating a spoofed source (10.2.2.1) on R1
R1# ping 172.21.21.5 source 10.2.2.1 repeat 1
Success rate is 0 percent (0/1)
!
! ASA detects the uRPF violation
%ASA-3-313001: Denied ICMP type=8, code=0 from 10.2.2.1 on interface out
%ASA-1-106021: Deny ICMP reverse path check from 10.2.2.1 to 172.21.21.5 on
interface out
ASA3# show asp drop | include rpf
Reverse-path verify failed (rpf-violated)
!
! Characterizing a uRPF violation with the Packet Tracer tool
ASA3# packet-tracer input out icmp 10.2.2.1 8 0 1 172.21.21.5
Phase: 1
Type: FLOW-LOOKUP
Subtype:
Result: ALLOW
[ output suppressed ]
Phase: 3
Type: ROUTE-LOOKUP
Subtype: input
Result: ALLOW
Config:
Additional Information:
in 10.2.2.0 255.255.255.0 dmz1

Result:
input-interface: out
input-status: up
input-line-status: up
output-interface: dmz1
output-status: up
output-line-status: up
Action: drop
Drop-reason: (rpf-violated) Reverse-path verify failed
!
ASA3# show ip verify statistics
interface mgmt: 0 unicast rpf drops
interface out: 1 unicast rpf drops
interface dmz1: 0 unicast rpf drops
interface inside: 0 unicast rpf drops

```

TCP Flags Filtering

The previous chapters showed that TCP is the truly stateful L4 protocol and that the so-called TCP flags play an important role to the TCP state machine. The stateful firewall solutions keep track not only of source/destination IP and L4 ports combinations but also verify the consistency of the TCP flags field with the sequence and acknowledgment numbers.

This section presents a special type of IOS ACL that makes it possible to filter based upon specific values carried on the TCP flags field. This resource may serve, for instance, the following purposes:

- Characterizing the distribution of flag combinations among the packets crossing the IOS router. This task is accomplished using only **permit** statements and was already studied in Chapter 4, “Learn the Tools. Know the Firewall.”
- Statelessly blocking certain flag combinations that do not occur on normal TCP operation but are used for reconnaissance attacks. This solution would be more suitable on routers that do not perform stateful inspection.

Because special filtering is used throughout this chapter, it is interesting to have quick look to the header information. Figure 11-4 documents the IP header and presents the meaning of its main fields, and Figure 11-5 does a similar job for the UDP and TCP headers, devoting differentiated attention to the meaning of the TCP flags.

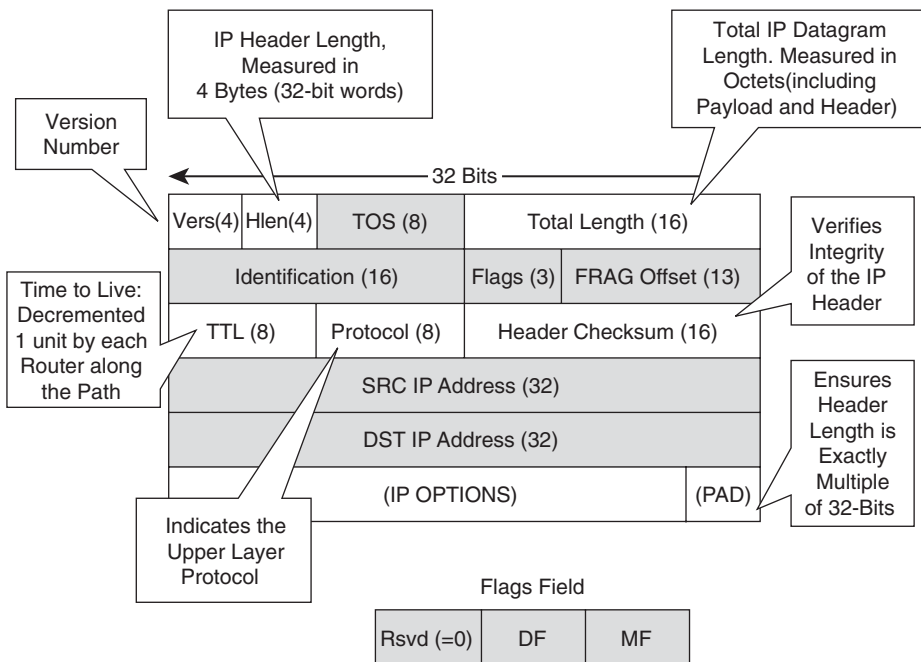


Figure 11-4 Revisiting the IP Header

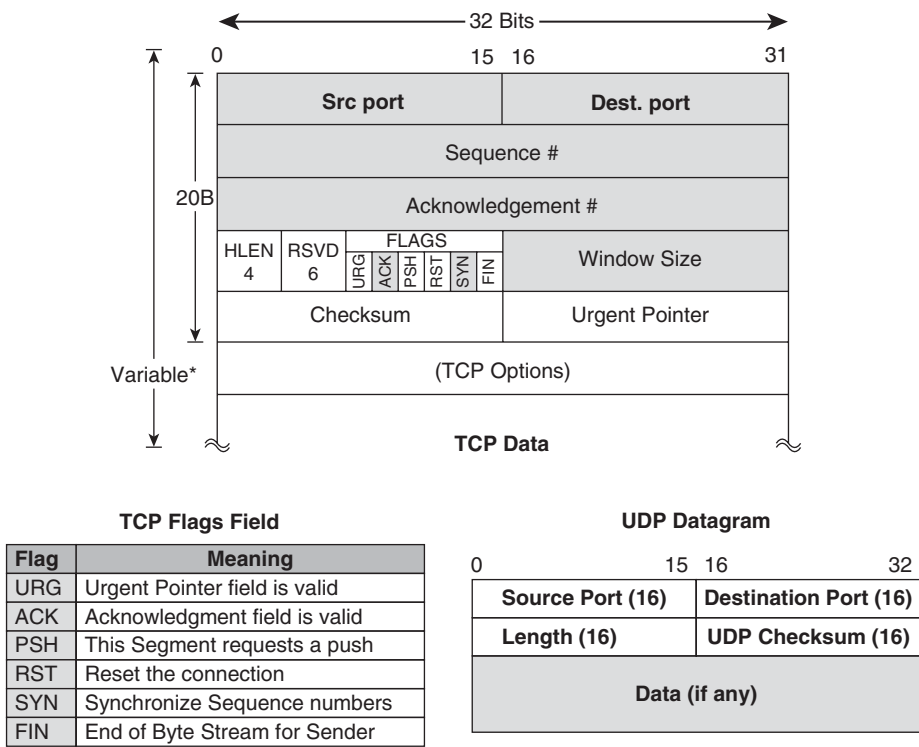


Figure 11-5 *Revisiting the TCP and UDP Headers*

Example 11-7 shows a Netflow v9 flow record designed to show information about several header fields that are important in the context of this chapter but are not present in the original Netflow record. This Netflow configuration is consistently used with the Special ACLs analyzed in the topology of Figure 11-6.

Example 11-7 *Reference Netflow Configuration for Use Within This Chapter*

```
flow record FLEXRECORD1
 match ipv4 precedence
 match ipv4 protocol
 match ipv4 source address
 match ipv4 destination address
 match transport source-port
 match transport destination-port
 match interface input
 collect ipv4 total-length
 collect ipv4 id
 collect ipv4 fragmentation flags
 collect ipv4 fragmentation offset
```

```

collect ipv4 ttl
collect ipv4 option map
collect transport tcp flags
collect interface output
collect counter bytes
collect counter packets
!
flow monitor FLEX1
record FLEXRECORD1

```

Figure 11-6 shows the reference topology used for the special ACLs covered in this chapter. The ACLs are always applied to interface FastEthernet4 (F4) as an input feature, and their names are chosen according to the section under analysis.

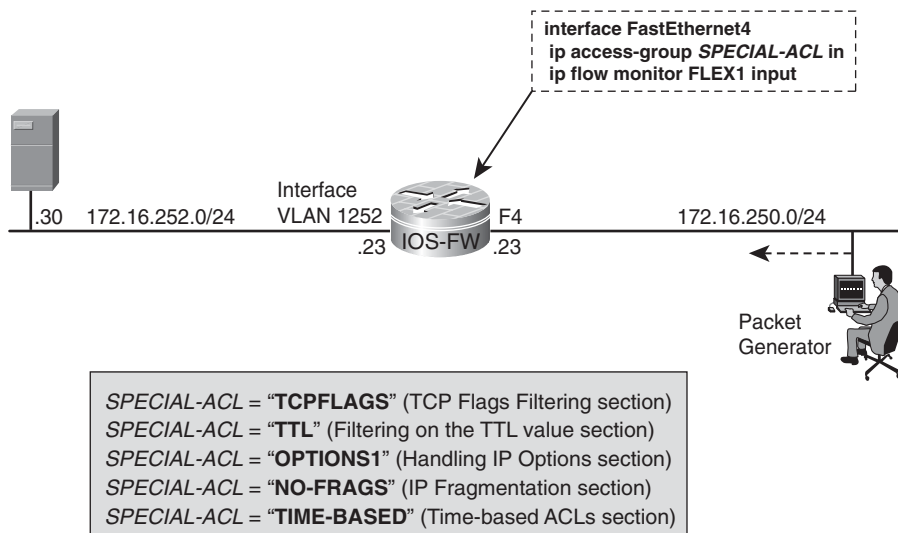


Figure 11-6 Reference Topology for the Study of Special ACLs

Example 11-8 presents a sample ip extended ACL used for TCP flag filtering. Some possible meanings for the entries in this ACL follow:

- Line 10 is aimed to block the NMAP Xmas Tree scan.
- Line 20 is intended to deny the NMAP Null scan.
- Line 30 is meant to block the last part of the NMAP Connect scan.
- Line 40 represents the beginning of the three-way handshake, where only the SYN flag is present.

- Line 50 represents the ACK flag marked, corresponding to the last part of the three-way handshake.
- Line 60 contains the ACK and PUSH flags marked while the SYN and URG are cleared, which is typical of the data transfer phase. It is convenient to notice that there is no reference to the values of the FIN and RST bits, thus allowing them to be used for connection termination.
- Line 70 covers another possibility of connection termination.

This example does not represent any attempt to explore the vast territory of TCP state transitions. It is just aimed at illustrating the behavior of a special type of ACL.

Example 11-8 shows not only the TCPFLAGS ACL in action but also the important information unveiled by Netflow.

Example 11-8 *Sample Filtering Based on the TCP Flags Field*

```
! Stateless filtering based on specific combinations of TCP flags
IOS-FW# show access-list TCPFLAGS
Extended IP access list TCPFLAGS
 10 deny tcp any any match-all +fin +psh +urg
 20 deny tcp any any match-all -ack -fin -psh -rst -syn -urg
 30 deny tcp any any match-all +ack +rst
 40 permit tcp any any match-all -ack -fin -psh -rst +syn -urg
 50 permit tcp any any match-all +ack -fin -psh -rst -syn -urg
 60 permit tcp any any match-all +ack +psh -syn -urg
 70 permit tcp any any match-all -ack -psh +rst -syn -urg

!
interface FastEthernet4
ip address 172.16.250.23 255.255.255.0
ip access-group TCPFLAGS in
ip flow monitor FLEX1 input
!
! Generating TCP packets with the values 41,43,45,47 in the flags field

IOS-FW# show flow monitor FLEX1 cache aggregate transport tcp flags transport
destination-port ipv4 destination address
Processed 16 flows
Aggregated to 4 flows


| IPV4 DST ADDR | TRNS DST PORT | TCP FLAGS | flows | bytes | pkts |
|---------------|---------------|-----------|-------|-------|------|
| 172.16.252.30 | 80            | 0x29      | 4     | 640   | 4    |
| 172.16.252.30 | 80            | 0x2B      | 4     | 640   | 4    |
| 172.16.252.30 | 80            | 0x2D      | 4     | 640   | 4    |
| 172.16.252.30 | 80            | 0x2F      | 4     | 640   | 4    |


```

```

!
IOS-FW# show access-list TCPFLAGS | include \
    10 deny tcp any any match-all +fin +psh +urg (89 matches)
!
! Generating TCP packets with values between 20 and 23 in the flags field

IOS-FW# show flow monitor FLEX1 cache aggregate transport tcp flags transport
destination-port ipv4 destination address
Processed 15 flows
Aggregated to 4 flows

```

IPV4 DST ADDR	TRNS DST PORT	TCP FLAGS	flows	bytes	pkts
172.16.252.30	80	0x14	4	640	4
172.16.252.30	80	0x15	4	640	4
172.16.252.30	80	0x16	4	640	4
172.16.252.30	80	0x17	3	480	3

```

!
IOS-FW# show access-list TCPFLAGS | include \
    30 deny tcp any any match-all +ack +rst (61 matches)

```

Note The **match-all** and **match-any** options are available only with extended named ACLs.

Filtering on the TTL Value

IP extended ACLs (named and numbered) can be employed to filter based upon the TTL value of packets. Although any Time-To-Live (TTL) value in the range 0 to 255 might be filtered, special handling is necessary when the TTL field assumes a value of 0 or 1.

Packets whose TTL value is 0 or 1 are sent to the process level because, according to basic IP definitions, they will never leave the device. The process level must check if a given packet is destined for the device itself and whether an ICMP TTL Expire message needs to be sent back.

Example 11-9 shows a sample ACL that was created to filter packets carrying TTL values lower than 30. The example also shows typical ACL log messages and the insight provided by the Netflow definitions of Example 11-7.

Example 11-9 Sample Filtering Based on the IP TTL Field

```

! Stateless filtering based on a range of TTL values
ip access-list extended TTL
deny tcp any any ttl lt 30 log
deny udp any any ttl lt 30 log
deny icmp any any ttl lt 30 log

```



```

permit tcp any host 172.16.252.30 eq www
permit tcp any host 172.16.252.30 eq 443
!
interface FastEthernet4
 ip address 172.16.250.23 255.255.255.0
 ip access-group TTL in
 ip flow monitor FLEX1 input
!
IOS-FW# show flow monitor FLEX1 cache aggregate ipv4 source address ipv4 protocol
ipv4 ttl
Processed 7 flows
Aggregated to 7 flows

```

IPV4 SRC ADDR	IP PROT	IP TTL	flows	bytes	pkts
=====	=====	=====	=====	=====	=====
172.16.250.201	6	37	1	260	1
172.16.250.202	6	12	1	260	1
172.16.250.203	6	28	1	260	1
172.16.250.205	6	5	1	260	1
172.16.250.206	6	9	1	260	1
172.16.250.207	6	17	1	260	1
172.16.250.208	6	50	1	260	1

```

!
! Sample logs of packets denied by ACL named 'TTL'
%SEC-6-IPACCESSLOGP: list TTL denied tcp 172.16.250.202(13002) ->
172.16.252.30(80), 1 packet
%SEC-6-IPACCESSLOGP: list TTL denied tcp 172.16.250.203(13003) ->
172.16.252.30(80), 1 packet
!
IOS-FW# show access-list TTL
Extended IP access list TTL
  10 deny tcp any any ttl lt 30 log (5 matches)
  20 deny udp any any ttl lt 30 log
  30 deny icmp any any ttl lt 30 log
  40 permit tcp any host 172.16.252.30 eq www (2 matches)
  50 permit tcp any host 172.16.252.30 eq 443

```

Note Dropping packets with TTL values 0 or 1 happens at the process level.

Handling IP Options

The IP Options field is optionally transmitted in the IP Datagram header but must be recognized and somehow dealt with by all IP-based systems (either hosts or gateways). The IP Options typically implement control functions that may be useful in particular situations but that are dispensable most of the time for regular communications.

The IP Options field is variable in length and must be a multiple of 4 bytes (32 bits) so that it can be included in the IP header length computation (which uses 4 bytes as its unit). The maximum acceptable options length is 40 bytes, leading to an IP header length of 60 bytes (0xF, corresponding to 15 pieces of 4 bytes). Two IP Options formats are allowed:

- **Single octet format:** Contains only the option type information.
- **Multiple octets format:** Contains option type (1 byte) and option length (1 byte) fields and the actual option-data bytes. The length byte refers to the total length of the field, and the data portion may contain other control fields, depending on the option carried. Examples 11-7 and 11-8 illustrate two sample options of this type.

Figure 11-7 presents two sample packets used for testing the Loose Source Route (LSR) and Strict Source Route (SSR) options. An SSR predetermines the address of every intermediate router along the path from source to destination, whereas an LSR defines only some of the possible hops. When using an LSR, routers are allowed to select any path between two of the listed hops. Both the LSR and SSR options include the Record Route option. (The actual IP used by each hop is registered, and the recorded route should be reversed by the destination host and attached to the possible reply packets.)

Figure 11-7 also brings the detailed structure of an LSR option suitable to specify up to four addresses. The type byte is subdivided in three parts:

- The copy bit is set to 1, meaning that this option must be copied into all possible fragments of the original packet. When this bit assumes the value 0, the option should appear only on the first fragment. The class field equals 0, stating that the LSR is deemed a *control* option. The option number is 3, and the resultant value for the type field is 131 (or 0x83).
- The length byte is set to 20 because it includes the four addresses (4 bytes each), three 1-byte fields (type, length, and pointer) and one extra *padding* byte (to adapt the length to the 4-bytes unit).
- The pointer (or offset) byte establishes the packet's current location within the source route. The pointer is set to 4 in the sample packet to indicate that the address present in position 4 must be considered. The router that processes this option increments the pointer by 4.

The structure for the SSR option is similar to that of the LSR. The main difference is in the option type. The SSR has the value 137 (0x89), and the LSR uses 131 (0x83).

Sample Echo Request packet carrying the Loose Source Route option

Ethernet Packet: 162 bytes

Dest Addr: 0015.6200.9871, Source Addr: 000B.468F.3991

Protocol: 0x0800

IP Version: 0x4, HdrLen: 0xA, TOS: 0x80 (Prec=Flash Override)

Length: 148, ID: 0x67A1, Flags-Offset: 0x0000

TTL: 60, Protocol: 1 (ICMP), Checksum: 0x000F (OK)

Source: 172.16.250.151, Dest: 172.16.252.30

Options: Length = 20

Loose Source and Record Route Option: 83

Length = 20 Pointer = 4

Position 4 = 172.16.250.23

Position 8 = 172.16.251.1

Position 12 = 0.0.0.0

Position 16 = 0.0.0.0

ICMP Type: 8, Code: 0 (Echo Request)

Checksum: 0xF7FF (OK)

Identifier: 0000, Sequence: 0000

Echo Data:

0 : 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

Sample Echo Request packet carrying the Strict Source Route option

Ethernet Packet: 162 bytes

Dest Addr: 0015.6200.9871, Source Addr: 000B.468F.3991

Protocol: 0x0800

IP Version: 0x4, HdrLen: 0xA, TOS: 0x80 (Prec=Flash Override)

Length: 148, ID: 0x6EC7, Flags-Offset: 0x0000

TTL: 60, Protocol: 1 (ICMP), Checksum: 0xD5F6 (OK)

Source: 172.16.250.145, Dest: 172.16.252.30

Options: Length = 20

Strict Source and Record Route Option: 89

Length = 20 Pointer = 4

Position 4 = 172.16.250.23

Position 8 = 172.16.252.30

Position 12 = 0.0.0.0

Position 16 = 0.0.0.0

ICMP Type: 8, Code: 0 (Echo Request)

Checksum: 0xF7FF (OK)

Identifier: 0000, Sequence: 0000

Echo Data:

0 : 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

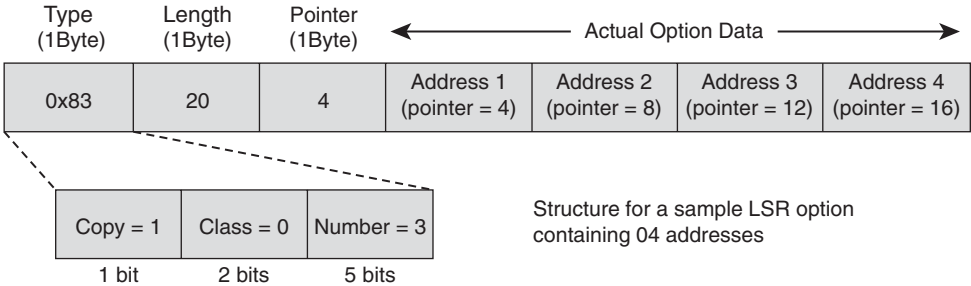


Figure 11-7 Sample Packets Containing the Source Route Options

Figure 11-8 shows two new sample packets containing the Record Route and Timestamp options. The figure documents the structure for a Timestamp option aimed to carry only timestamps (rather than IP address/timestamp pairs). Other details about the format of this particular option follow:

- The copy bit is set to 0, which means that, if IP fragmentation occurs, the option should appear only on the first fragment. The class field equals 2 (*debugging and measurement*) and the option number is 4, yielding a value of 68 (or 0x44) for the type byte.
- The length value in this case is chosen to be 36, giving 32 bytes for timestamps (each being 04 bytes long) and 4 bytes for option description.
- The pointer byte points to the first free timestamp (in this case, position 5).
- The overflow field is a 4-bit unsigned integer that represents the number of IP systems that could not register timestamps because of lack of space in the data portion.

- The 4 bits in the Flag field determine how timestamps should be registered. A value of 0 means only timestamps (4 bytes each) are to be recorded. A value of 1 establishes that each timestamp is preceded by the IP address of the system that recorded it. Finally, a value of 2 predefines the IP addresses, and recording occurs only when a system finds its own address in the IP list.

Because of their potential of being employed as an attack artifact, IP options are frequently considered “evil” and, as soon as they are detected, they are altogether dropped. The current section presents some ways to deal with this optional attribute of the IP header.

IOS offers the possibility to permit specific types of IP options and may also be instructed to drop every packet containing options.

ASA, as will be documented in the pertinent section, discards and logs all packets conveying IP options.

Sample Echo Request packet carrying the Record Route option

Ethernet Packet: 182 bytes
 Dest Addr: 0015.6200.9871, Source Addr: 000B.468F.3991
 Protocol: 0x0800
 IP Version: 0x4, **HdrLen: 0xF**, TOS: 0x80 (Prec=Flash Override)
Length: 168, ID: 0x6CE7, Flags-Offset: 0x0000
 TTL: 60, Protocol: 1 (ICMP), Checksum: 0xACEE (OK)
 Source: 172.16.250.152, Dest: 172.16.252.30
Options: Length = 40
Record Route Option: 07 Length = 40 Pointer = 4
 Position 4 = 0.0.0.0
 Position 8 = 0.0.0.0
 Position 12 = 0.0.0.0
 Position 16 = 0.0.0.0
 Position 20 = 0.0.0.0
 Position 24 = 0.0.0.0
 Position 28 = 0.0.0.0
 Position 32 = 0.0.0.0
 Position 36 = 0.0.0.0
 ICMP Type: 8, Code: 0 (Echo Request)
 Checksum: 0xF7FF (OK)
 Identifier: 0000, Sequence: 0000
 Echo Data:
 0 : 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

Sample Echo Request packet carrying the Internet Timestamp option

Ethernet Packet: 178 bytes
 Dest Addr: 0015.6200.9871, Source Addr: 000B.468F.3991
 Protocol: 0x0800
 IP Version: 0x4, **HdrLen: 0xE**, TOS: 0x80 (Prec=Flash Override)
Length: 164, ID: 0x6216, Flags-Offset: 0x0000
 TTL: 60, Protocol: 1 (ICMP), Checksum: 0x7A8C (OK)
 Source: 172.16.250.146, Dest: 172.16.252.30
Options: Length = 36
Internet Timestamp Option: 44
 Length: 36 Pointer: 5 **Overflow: 4** Flags: 0
 Position 5 : Timestamp: 00000000
 Position 9 : Timestamp: 00000000
 Position 13: Timestamp: 00000000
 Position 17: Timestamp: 00000000
 Position 21: Timestamp: 00000000
 Position 25: Timestamp: 00000000
 Position 29: Timestamp: 00000000
 Position 33: Timestamp: 00000000
 ICMP Type: 8, Code: 0 (Echo Request)
 Checksum: 0xF7FF (OK)
 Identifier: 0000, Sequence: 0000
 Echo Data:
 0 : 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

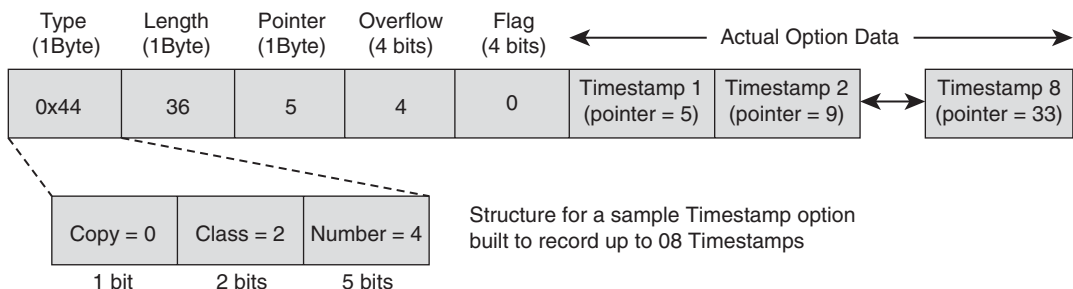


Figure 11-8 Sample Packets Containing the Record Route and Timestamp Options

Having quickly revisited the theory behind IP options, it is now time to shift gears to some practical work.

Stateless Filtering of IP Options on IOS

Example 11-10 documents a sample ACL that illustrates how IOS can selectively permit IP options. The first four lines permit the options documented within Figures 11-7 and 11-8 for ICMP packets, and the fifth line denies packets bearing any other types of options and options carried in protocols other than ICMP.

Example 11-11 shows the ACL *OPTIONS1* in action for packets containing either the SSR or LSR options. Observe the flow information provided by the special Netflow recorded crafted in Example 11-7.

Example 11-10 Baseline ACL for IP Options Filtering on IOS

```
ip access-list extended OPTIONS1
    permit icmp any any option lsr log
    permit icmp any any option ssr log
    permit icmp any any option record-route log
    permit icmp any any option timestamp log
    deny ip any any option any-options
    permit ip any any
!
interface FastEthernet4
ip address 172.16.250.23 255.255.255.0
ip access-group OPTIONS1 in
ip flow monitor FLEX1 input
```

Example 11-11 Filtering Packets with the Source Route Options

```
! ICMP packets containing the SSR Option are permitted
IOS-FW# show flow monitor FLEX1 cache aggregate ipv4 source address ipv4 protocol
ipv4 option map
Processed 3 flows
Aggregated to 3 flows


| IPV4 SRC ADDR  | <b>IPV4 OPTION MAP</b> | IP PROT | flows | bytes | pkts |
|----------------|------------------------|---------|-------|-------|------|
| 172.16.250.140 | 0x00000100             | 1       | 1     | 168   | 1    |
| 172.16.250.141 | 0x00000100             | 1       | 1     | 168   | 1    |
| 172.16.250.142 | 0x00000100             | 1       | 1     | 168   | 1    |


!
%SEC-6-IPACCESSLOGDP: list OPTIONS1 permitted icmp 172.16.250.140 -> 172.16.252.30
(0/0), 1 packet
!
IOS-FW# show access-list OPTIONS1 | include \(\
```

```

    20 permit icmp any any option ssr log (10 matches)
!
! ICMP packets carrying the LSR Option are allowed

IOS-FW# show flow monitor FLEX1 cache aggregate ipv4 source address ipv4 protocol
ipv4 option map
Processed 3 flows
Aggregated to 3 flows

```

IPV4 SRC ADDR	IPV4 OPTION MAP	IP PROT	flows	bytes	pkts
172.16.250.140	0x00000004	1	1	168	1
172.16.250.141	0x00000004	1	1	168	1
172.16.250.142	0x00000004	1	1	168	1

```

!
IOS-FW# show access-list OPTIONS1 | include \(\
    10 permit icmp any any option lsr log (12 matches)

```

Example 11-12 registers the activity of the ACL *OPTIONS1* for packets containing either the Record Route or Timestamp options.

Example 11-13 shows the same ACL dropping packets that carry other option types. This last example also reveals, by means of the **show ip traffic** command, statistical information about IP options handling by this specific router. Notice, in the example, that the router sent 72 ICMP *Parameter Problem* messages in response to the 72 packets containing *bad options*. (Before each test all the appropriate counters were cleared.)

Example 11-12 Filtering Packets with the Record Route and Timestamp Options

```

! ICMP packets bearing the Record Route option are forwarded
IOS-FW# show flow monitor FLEX1 cache aggregate ipv4 source address ipv4 protocol
ipv4 option map
Processed 2 flows
Aggregated to 2 flows

```

IPV4 SRC ADDR	IPV4 OPTION MAP	IP PROT	flows	bytes	pkts
172.16.250.140	0x00000040	1	1	188	1
172.16.250.141	0x00000040	1	1	188	1

```

!
IOS-FW# show access-list OPTIONS1 | include \(\
    30 permit icmp any any option record-route log (8 matches)
!
! ICMP packets containing the Timestamp option are permitted

IOS-FW# show flow monitor FLEX1 cache aggregate ipv4 source address ipv4 protocol
ipv4 option map
Processed 4 flows

```

```
Aggregated to 4 flows
IPV4 SRC ADDR    IPV4 OPTION MAP  IP PROT    flows    bytes    pkts
=====
172.16.250.140   0x00000008             1          1       184        1
172.16.250.141   0x00000008             1          1       184        1
172.16.250.142   0x00000008             1          1       184        1
172.16.250.143   0x00000008             1          1       184
!
IOS-FW# show access-list OPTIONS1 | include \(
    40 permit icmp any any option timestamp log (9 matches)
```

Example 11-13 *Dropping Packets That Carry IP Options That Are Not Allowed*

```
! Packets containing an option that was not explicitly allowed, are dropped
IOS-FW# show flow monitor FLEX1 cache aggregate ipv4 source address ipv4 protocol
ipv4 option map
Processed 3 flows
Aggregated to 3 flows
IPV4 SRC ADDR    IPV4 OPTION MAP  IP PROT    flows    bytes    pkts
=====
172.16.250.140   0x00000000             1          1       160        1
172.16.250.141   0x00000000             1          1       160        1
172.16.250.142   0x00000000             1          1       160        1
!
ICMP: dst (172.16.252.30) administratively prohibited unreachable sent to
172.16.250.141
%SEC-6-IPACCESSLOGDP: list OPTIONS1 denied icmp 172.16.250.141 -> 172.16.252.30
(0/0), 1 packet
!
IOS-FW# show access-list OPTIONS1 | include \(
    50 deny ip any any option any-options log (4 matches)
!
! Information concerning packets containing IP options
IOS-FW# show ip traffic
IP statistics:
  Rcvd: 269 total, 0 local destination
        0 format errors, 0 checksum errors, 0 bad hop count
        0 unknown protocol, 0 not a gateway
        7 security failures, 72 bad options, 269 with options
  Opts: 151 end, 0 nop, 7 basic security, 54 loose source route
        53 timestamp, 0 extended security, 8 record route
        0 stream ID, 57 strict source route, 0 alert, 0 cipso, 0 ump
        0 other
  Frags: 0 reassembled, 0 timeouts, 0 couldn't reassemble
        0 fragmented, 0 fragments, 0 couldn't fragment
```

```

Bcast: 0 received, 0 sent
Mcast: 0 received, 0 sent
Sent: 0 generated, 172 forwarded
Drop: 97 encapsulation failed, 0 unresolved, 0 no adjacency
        0 no route, 0 unicast RPF, 0 forced drop
        1204 options denied
Drop: 0 packets with source IP address zero
Drop: 0 packets with internal loop back IP address
        0 physical broadcast
ICMP statistics:
Rcvd: 0 format errors, 0 checksum errors, 0 redirects, 0 unreachable
        0 echo, 0 echo reply, 0 mask requests, 0 mask replies, 0 quench
        0 parameter, 0 timestamp, 0 timestamp replies, 0 info request, 0 other
        0 irdp solicitations, 0 irdp advertisements
        0 time exceeded, 0 info replies
Sent: 0 redirects, 25 unreachable, 0 echo, 0 echo reply
        0 mask requests, 0 mask replies, 0 quench, 0 timestamp, 0 timestamp replies
        0 info reply, 0 time exceeded, 72 parameter problem
        0 irdp solicitations, 0 irdp advertisements

```

IP Options Drop on IOS

Instead of selectively enabling specific options, IOS offers the possibility to drop all packets that carry any kind of IP Options. Example 11-14 illustrates such a situation:

- A warning message is issued when the global configuration command **ip options drop** is entered. The message reminds you that some protocols such as RSVP might not work properly if all packets conveying IP options are discarded.
- The **show ip traffic** command (issued after all the pertinent counters were cleared) reveals that there are only *options denied*. (Compare this output with that of Example 11-13.)

Example 11-14 *Deliberately Dropping Packets That Carry IP Options on IOS*

```

! Instructing IOS to drop packets containing IP Options
IOS-FW(config)# ip options drop
% Warning: RSVP and other protocols that use IP Options packets may not
function as expected.
!
IOS-FW# show ip traffic | include option
        0 security failures, 0 bad options, 0 with options
        76 options denied

```


Note The `no ip source-route` global configuration command instructs Cisco IOS routers to discard datagrams containing a source route option.

IP Options Drop on ASA

The behavior of ASA for IP Options handling basically reproduces that of the `ip options drop` command earlier explained for IOS. Example 11-15 portrays a situation in which an *extended ping* (containing the Record Route option) is sent to a host behind ASA. As you can see in the example, the firewall not only drops the packet but also issues a log message that registers the specific option present in the packet.

This example also documents the usage of the `show asp drop` command, a helpful resource to point out the cause of a packet drop by ASA. This command is extensively employed throughout this chapter.

Example 11-15 ASA Behavior for IP Options

```
! Generating a packet containing the Record Route option
R1# ping ip
Target IP address: 172.16.222.13
Repeat count [5]: 1
Datagram size [100]: 200
Timeout in seconds [2]: <Enter>
Extended commands [n]: y
Source address or interface: 172.16.220.10
Type of service [0]: <Enter>
Set DF bit in IP header? [no]: <Enter>
Validate reply data? [no]: <Enter>
Data pattern [0xABCD]: <Enter>
Loose, Strict, Record, Timestamp, Verbose[none]: R
Number of hops [ 9 ]: <Enter>
Loose, Strict, Record, Timestamp, Verbose[RV]: <Enter>
Sweep range of sizes [n]: <Enter>
Type escape sequence to abort.
Sending 1, 200-byte ICMP Echos to 172.16.222.13, timeout is 2 seconds:
Packet sent with a source address of 172.16.220.10
Packet has IP options: Total option bytes= 39, padded length=40
Record route: <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
```

```

(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
Request 0 timed out
Success rate is 0 percent (0/1)
!
%ASA-6-106012: Deny IP from 172.16.220.10 to 172.16.222.13, IP options: "Record
Route"
!
ASA1# show asp drop | include option
IP option drop (invalid-ip-option)

```

3

Dealing with IP Fragmentation

One of the design goals for the Internet protocol, working at Layer 3 of the OSI model is to be independent of the underlying L2 media traversed along the path. This introduces an abstraction for the hosts being interconnected, enabling packets to be handled in the same way, irrespectively of Layer 2. Such an approach becomes even more important because newer L2 technologies are frequently offered, and it would be impractical to modify the whole internetwork every time some L2 medium was changed.

Every type of Layer 2 network that lends itself to the transport of IP datagrams includes among its basic definitions a frame format and a corresponding Maximum Frame Size. The IP Maximum Transfer Unit (IP MTU) for a given L2 medium is defined as the largest IP datagram that can be transmitted over that medium without the need to be divided into smaller packets (called *fragments*).

Ideally, each IP datagram sent by an originating host is carried over a single L2 frame. However, it is hard to know in advance what are all the possible MTUs throughout the route from source to destination, mainly if the network under consideration is the global Internet. Furthermore, being IP a connectionless protocol, there is no rule that obligates packets traveling between a pair of hosts to always follow the same path.

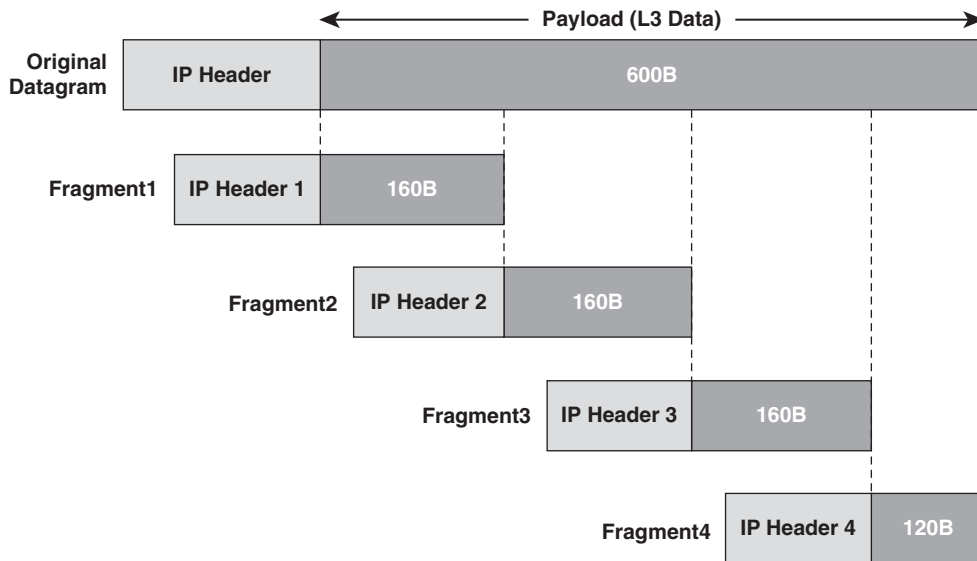
Whenever IP fragmentation takes place (either at the originating host or on a router across the path), the initial datagram is not reassembled until reaching the IP layer of its final destination. Other important characteristics of IP fragmentation follow:

- Each fragment becomes an individual packet (with a header length, a total length and an appropriate data area) that should be routed independently of any other datagrams.
- The IP header fields employed in the fragmentation process are the Identification (ID), the Flags and the Fragment Offset (FO). Combining the information of these fields allows the final destination to correctly perform the reassembly. (Please recall Figure 11-4).
- Each IP packet that a sender transmits contains a value for the Identification field that should be copied into all eventual fragments. Using the ID information, the destination host can determine which incoming fragments belong to a given datagram and buffer all of them until the last fragment received.
- All the fragments of a packet except the last one have the *more fragments* flag set to 1. For the last fragment, this flag assumes the value 0.
- The Fragment Offset corresponds to the sum of L3 data bytes of previous fragments (belonging to the same packet). For instance, consider a scenario in which the original IP data area has a length of 1000 bytes and each fragment can carry 200 bytes of IP data (MTU = 220). A value of 600 in the fragment offset field indicates that 600 data bytes should have arrived in the 03 previous fragments.
- Given that IP represents the fragment offset field as a multiple of 8 bytes, the fragment size (length of the data area within each fragment) should be divisible by 8. For example, if the MTU is 200 and the IP header length is 20 bytes, there is room for 180 bytes of data. But 180 is not divisible by 8, and the closest multiple of 8 is 176. This yields a fragment size of 176 bytes.

Figure 11-9 shows an example of IP fragmentation in which the IP MTU is 180, yielding a total length of 180 and an L3 data area of 160. Considering that 160 is divisible by 8, the resultant fragment size is 160 octets. The total length and data length of the original datagram were respectively 620 bytes and 600 bytes. Given this value of fragment size, it is possible to write for the L3 Data: (600 = 160 + 160 + 160 + 120). This figure includes a summarizing table that highlights the relevant header fields on each packet.

Figure 11-10 registers an original ICMP packet that is subject to fragmentation under the circumstances described in Figure 11-9. The resultant fragments arrive at the IOS-FW router, in the topology depicted in Figure 11-6.

Note The diagrams and table presented in Figure 11-9 were built with the intent to better explain the fragmentation process represented by the packets of Figure 11-10.



	Total Length	L3 Data Length	ID	Rsvd Bit	DF Bit	MF Bit	Fragment Offset (multiple of 08 bytes)	Resultant Flags-Offset
Original	620	600	0x6E81	0	0	0	0	0x0000
Frag 1	180	160	0x6E81	0	0	1	0	0x2000
Frag 2	180	160	0x6E81	0	0	1	160 = 8 x 20 (20 = 0x14)	0x2014
Frag 3	180	160	0x6E81	0	0	1	320 = 8 x 40 (40 = 0x28)	0x2028
Frag 4	140	120	0x6E81	0	0	0	480 = 8 x 60 (60 = 0x3C)	0x003C

Figure 11-9 *Revisiting IP Fragmentation*

So, if fragmentation is an integral part of the IP specification, *why worry about it?* Is it not possible just to forget this and let IP take care of fragmentation when necessity arises? Some good reasons to avoid it can be listed. If you are concerned about security (which is true for anyone using this book), this is even more important:

- Fragmentation is process-intensive for the routers that need to break the original packets into smaller ones.
- Remembering that every fragment is routed independently and that the main unit of performance for routers is pps (*packets per second*), a large amount of fragments increases the number of packets that need to be routed and, therefore, may impact other routers along the path.

Original Packet: Ethernet Packet: 634 bytes Dest Addr: 0015.6200.9871, Source Addr: 000B.468F.3991 Protocol: 0x0800 IP Version: 0x4, HdrLen: 0x5, TOS: 0x80 (Prec=Flash Override) Length: 620, ID: 0x6E81, Flags-Offset: 0x0000 TTL: 60, Protocol: 1 (ICMP), Checksum: 0xBEB9 (OK) Source: 172.16.250.150, Dest: 172.16.252.30 ICMP Type: 8, Code: 0 (Echo Request) Checksum: 0xF7FF (OK) Identifier: 0000, Sequence: 0000 Echo Data: 0 : 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 Fragment 1: Ethernet Packet: 194 bytes Dest Addr: 0015.6200.9871, Source Addr: 000B.468F.3991 Protocol: 0x0800 IP Version: 0x4, HdrLen: 0x5, TOS: 0x80 (Prec=Flash Override) Length: 180, ID: 0x6E81, Flags-Offset: 0x2000 (more fragments) TTL: 60, Protocol: 1 (ICMP), Checksum: 0xA071 (OK) Source: 172.16.250.150, Dest: 172.16.252.30 ICMP Type: 8, Code: 0 (Echo Request) Checksum: 0xF7FF (OK) Identifier: 0000, Sequence: 0000 Echo Data: 0 : 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000	Fragment 2: Ethernet Packet: 194 bytes Dest Addr: 0015.6200.9871, Source Addr: 000B.468F.3991 Protocol: 0x0800 IP Version: 0x4, HdrLen: 0x5, TOS: 0x80 (Prec=Flash Override) Length: 180, ID: 0x6E81, Flags-Offset: 0x2014 (more fragments) TTL: 60, Protocol: 1 (ICMP), Checksum: 0xA05D (OK) Source: 172.16.250.150, Dest: 172.16.252.30 Data: 0 : 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 Fragment 3: Ethernet Packet: 194 bytes Dest Addr: 0015.6200.9871, Source Addr: 000B.468F.3991 Protocol: 0x0800 IP Version: 0x4, HdrLen: 0x5, TOS: 0x80 (Prec=Flash Override) Length: 180, ID: 0x6E81, Flags-Offset: 0x2028 (more fragments) TTL: 60, Protocol: 1 (ICMP), Checksum: 0xA049 (OK) Source: 172.16.250.150, Dest: 172.16.252.30 Data: 0 : 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 Fragment 4: Ethernet Packet: 154 bytes Dest Addr: 0015.6200.9871, Source Addr: 000B.468F.3991 Protocol: 0x0800 IP Version: 0x4, HdrLen: 0x5, TOS: 0x80 (Prec=Flash Override) Length: 140, ID: 0x6E81, Flags-Offset: 0x003C TTL: 60, Protocol: 1 (ICMP), Checksum: 0xC05D (OK) Source: 172.16.250.150, Dest: 172.16.252.30 Data: 0 : 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

Figure 11-10 *Sample IP Datagram and Associated Fragments*

- End hosts need to reassemble the received fragments, another process-intensive task.
- If a single fragment is lost, the entire packet must be retransmitted, but because IP does not handle retransmission, this task is delegated to upper-layer protocols. TCP embeds a mechanism to deal with retransmission, but UDP (a connectionless L4 protocol) must rely on application protocols to take care of it.

Knowledgeable hackers understand the theory behind fragmentation and reassembly and try to take advantage of them, mainly to cause denial-of-service (DoS). Some fragmentation based attacks, that have hit networks and hosts through time, may be enumerated:

- **Ping of Death:** Sending a large ICMP echo request packet, in the form of a chain of fragments, which after reassembly, exceeds the maximum valid length of 65535 bytes for an IP datagram. This is an old attack and operating systems should have a defense for it. Anyway, if firewall devices impose restrictions on the maximum number of fragments in a chain, they might contribute to protection.
- **Reassembly buffer overflow:** Happens when there is an excessive number of incomplete datagrams waiting for reassembly in the receiving host. Firewalls performing

Virtual Reassembly can drop packets after their fragment tables are full or after the reassembly timeout for a given datagram has been reached.

- **Tiny fragment attack:** Employs small TCP packets, crafted in such a way that a part of the L4 header (for instance including the flags field) travels in the second fragment. With such an approach, the attacker hopes that only the first fragment is examined and the remaining ones will be allowed through. Firewalls enforcing a minimum acceptable fragment size protect against this kind of attack.
- **Attacks based on fragment overlapping:** This occurs when the offset of a certain fragment overlaps with the offset of another. This induces memory allocation on the IP stack of receiving hosts to hold packets that, after all, are invalid. This attack class can be used either with the intent of causing DoS (such as with the UDP *Teardrop* exploit) or in an attempt to overwrite the data portion of previous fragments in the chain and circumvent defense systems. For instance, in this last case, some random data can be included with a part of the attack in one fragment, whereas future fragments can be crafted to overwrite the original data with the remainder of the attack. Firewalls that support Virtual Fragment Reassembly can mitigate this class of attacks.

This list is a quick review of some well-known attack techniques that involve IP fragments. Information about many more can be found with simple Internet searches.

One natural recommendation after this initial analysis is the classic *know your network*. For example, within the boundaries of your Intranet, if you know about the L2 media in use, a suitable MTU might be chosen so that there is no need for fragmentation to take place. For this to be successful, the application team should be aware of this ideal packet size and adopt it. Such an approach contributes not only to avoid wasting resources but also to virtually eliminate fragmentation-oriented attacks.

For the Internet edge, the story is a bit different because you cannot determine all the possible access technologies traversed to reach your network.

Stateless Filtering of IP Fragments in IOS

After briefly discussing the IP fragmentation theory, you should have observed that the first fragment carries the L4 header and typically some L4 data, whereas the *noninitial fragments* contain only the remaining L4 data. This poses a challenge to packet filtering devices. For example, a TFTP packet (using UDP port 69) that undergoes fragmentation has the UDP port listed solely on the initial fragment. The subsequent fragments tell that this is an UDP packet (a value of 17 in the IP *protocol type* field) but does not include the L4 port information.

Example 11-16 shows an ACL that denies any noninitial fragments and permits specific ICMP and UDP traffic in the network, as shown in Figure 11-6. The packet generator

sends fragments analogous to those registered in Figure 11-10. Some **noteworthy** facts in direct relation with Example 11-16 follow:

- For each permitted ICMP packet, there are three denied packets containing fragments. This is a consequence of the original packet being divided into four fragments, three of which are naturally noninitial. (This is shown by both the ACL and Netflow.)
- The fragment offset in the Netflow output is not presented as a multiple of 8 bytes but rather as a decimal number referring to the amount of data in the previous fragment. For instance, the data portion in the packet with ID 28925 is divided into three fragments of 160 bytes each, and a final fragment of 120 bytes.
- An ICMP Fragment Reassembly Time Exceeded message (Type 11, code 1) is issued by the destination host because some fragments in each packet are always missing (as a result of being dropped by the IOS ACL).

Example 11-16 *Sample ACL for Filtering IP Fragments on IOS (1)*

```
ip access-list extended NO-FRAGS
deny udp any any fragments
deny tcp any any fragments
deny icmp any any fragments
deny ip any any fragments
permit udp any 172.16.252.0 0.0.0.255 eq 700
permit icmp any 172.16.252.0 0.0.0.255 echo
permit icmp any 172.16.252.0 0.0.0.255 echo-reply
!
interface FastEthernet4
ip address 172.16.250.23 255.255.255.0
ip access-group NO-FRAGS in
ip flow monitor FLEX1 input
!
IOS-FW# show access-list NO-FRAGS | include \(
    30 deny icmp any any fragments (18 matches)
    60 permit icmp any 172.16.252.0 0.0.0.255 echo (6 matches)
!
IOS-FW# show flow monitor FLEX1 cache aggregate ipv4 id ipv4 destination address
ipv4 fragmentation offset
Processed 6 flows
Aggregated to 6 flows


| <b>IPV4 ID</b> | <b>IPV4 DST ADDR</b> | <b>IP FRAG OFFSET</b> | <b>flows</b> | <b>bytes</b> | <b>pkts</b> |
|----------------|----------------------|-----------------------|--------------|--------------|-------------|
| 28637          | 172.16.252.30        | 0                     | 1            | 180          | 1           |
| 28637          | 172.16.252.30        | 160                   | 1            | 500          | 3           |
| <b>28925</b>   | 172.16.252.30        | 0                     | 1            | 180          | 1           |
| <b>28925</b>   | 172.16.252.30        | 160                   | 1            | 500          | 3           |


```

```

27684 172.16.252.30          0          1          180          1
27684 172.16.252.30        160          1          500          3
!
! ICMP time exceeded messages are sent back to the source hosts

ICMP: time exceeded (reassemble) sent to 172.16.250.144 (dest was 172.16.252.30)

```

In Example 11-17, an original UDP packet destined to port 700 on host 172.16.252.30, with 540 bytes of IP total length (and 520 bytes of L3 data) is divided into three fragments after crossing some medium with IP MTU = 200 bytes. Recall that, even though this MTU can convey 180 bytes of L3 data, it produces a fragment size of 176 octets (because of the need of being a multiple of 8 bytes). The original 520 bytes of data are then decomposed as: $520 = 176 + 176 + 168$.

Example 11-17 also characterizes the ratio 2:1 between noninitial fragments and the first one for a given L3 ID. It also highlights that the L4 port (700 in this case) is present only in the initial fragment.

Example 11-17 Sample ACL for Filtering IP Fragments on IOS (2)

```

IOS-FW# show flow monitor FLEX1 cache aggregate ipv4 protocol ipv4 destination
address transport destination-port
Processed 8 flows
Aggregated to 2 flows

```

IPV4 DST ADDR	TRNS DST PORT	IP PROT	flows	bytes	pkts
172.16.252.30	700	17	4	784	4
172.16.252.30	0	17	4	1536	8

```

!
IOS-FW# show access-list NO-FRAGS | include \
10 deny udp any any fragments (340 matches)
50 permit udp any 172.16.252.0 0.0.0.255 eq 700 (170 matches)

```

Virtual Fragment Reassembly on IOS

In the previous section, a stateless method of filtering noninitial fragments was presented. (This can be a valid solution to simply eliminate fragments in more controlled network environments in which MTUs are known in advance.) The current section proposes a new approach called the Virtual Fragment Reassembly, which is aimed at assembling fragments before applying inspection policies.

Example 11-18 initially establishes that the maximum acceptable number of fragments per datagram is 3. If this number is exceeded, a Syslog message is generated.

In a second scenario, Example 11-18 shows how to define an upper bound for the number of simultaneous reassembly sessions. When this threshold is crossed, a Syslog message is

issued. In both cases, the **show ip virtual-reassembly** command reveals the configured reassembly parameters and the current values being measured for each of them.

Example 11-18 IOS Virtual Fragment Reassembly in Action

```
interface FastEthernet4
 ip address 172.16.250.23 255.255.255.0
 ip flow monitor FLEX1 input
 ip virtual-reassembly max-fragments 3
!
%IP_VFR-4-TOO_MANY_FRAGMENTS: FastEthernet4: Too many fragments per datagram (more
than 3) - sent by 172.16.250.140, destined to 172.16.252.30
!
! Defining a limit for the number of simultaneous fragmented datagrams

interface FastEthernet4
 ip address 172.16.250.23 255.255.255.0
 ip flow monitor FLEX1 input
 ip virtual-reassembly max-fragments 5 max-reassemblies 10 timeout 8
!
%IP_VFR-4-FRAG_TABLE_OVERFLOW: FastEthernet4: the fragment table has reached its
maximum threshold 10
!
IOS-FW# show ip virtual-reassembly f4
FastEthernet4:
  Virtual Fragment Reassembly (VFR) is ENABLED...
  Concurrent reassemblies (max-reassemblies): 10
  Fragments per reassembly (max-fragments): 5
  Reassembly timeout (timeout): 8 seconds
  Drop fragments: OFF
  Current reassembly count:10
  Current fragment count:30
  Total reassembly count:0
  Total reassembly timeout count:53
```

Virtual Fragment Reassembly on ASA

The behavior of the Virtual Fragment Reassembly feature of ASA is similar to that of IOS. Example 11-19 starts by registering the default values for the reassembly parameters, which have the meanings explained in the following:

- **chain limit:** The maximum number of packets into which an original IP packet can be fragmented.
- **size limit:** The maximum number of packets that can be in the IP reassembly database waiting for reassembly.

- **timeout limit:** This timer defines the maximum number of seconds to wait for an entire fragmented packet to arrive and starts after the arrival of its first fragment. If any fragment in a chain is still missing after the timer expires, all the fragments already received for that packet are discarded.

Example 11-19 also shows how to modify some of the parameters on a per-interface basis:

- The fragment chain is changed to 3, and a fragment set containing more than 3 elements is entirely discarded.
- Following an increase of the reassembly timeout value to 30 seconds, a high rate of packets with missing fragments is directed through ASA. The example shows that the database size limit of 200 is exceeded and that there is a significant rise in the *Fail* counter.
- The last part of the example shows ASA detecting and discarding overlapping fragments. It is interesting to observe that ASA generically refers to this type of event as *deny IP teardrop fragment* (even when UDP was not the L4 protocol used).

Example 11-19 Virtual Fragment Reassembly in Action on ASA

```
! Default Fragmentation settings for an interface on ASA
ASA1# show fragment out1
Interface: out1
    Size: 200, Chain: 24, Timeout: 5, Reassembly: virtual
    Queue: 0, Assembled: 0, Fail: 0, Overflow: 0
!
! Changing the value of the fragment chain to 3 on interface 'out1'

ASA1(config)# fragment chain 3 out1
!
! A packet that was fragmented in more than 03 pieces is dropped
%ASA-4-209005: Discard IP fragment set with more than 3 elements: src =
172.16.220.120, dest = 172.16.222.13, proto = ICMP, id = 12369
!
! The 'Fail' counter increases
ASA1# show fragment out1
Interface: out1
    Size: 200, Chain: 3, Timeout: 5, Reassembly: virtual
    Queue: 0, Assembled: 0, Fail: 7, Overflow: 0
!
! Limit of simultaneous fragments being handled is reached

ASA1# show fragment out1
Interface: out1
    Size: 200, Chain: 24, Timeout: 30, Reassembly: virtual
    Queue: 200, Assembled: 0, Fail: 486, Overflow: 12958
```

```

!
%ASA-4-209003: Fragment database limit of 200 exceeded:  src = 172.16.220.213, dest
= 172.16.222.13, proto = UDP, id = 21000
!
! Overlapping IP fragments are discarded

%ASA-2-106020: Deny IP teardrop fragment (size = 160, offset = 0) from
172.16.220.202 to 172.16.222.13
%ASA-2-106020: Deny IP teardrop fragment (size = 160, offset = 160) from
172.16.220.203 to 172.16.222.13

```

Note The change in the default reassembly parameters simply had the purpose of making it easier to illustrate ASA (or IOS behavior). The examples just presented do not provide any guidelines or recommendations for real-life implementations.

Flexible Packet Matching

So far, this chapter has mainly presented some kinds of ACLs that can deal with specific fields of the IP and TCP headers. The current section introduces a sophisticated stateless filtering resource denominated Flexible Packet Matching (FPM).

Contrary to regular ACLs, FPM is not limited to filter only on predetermined fields. It offers the possibility of matching patterns at an arbitrary depth either in the packet header or payload area. This mechanism enables users to create customized classification criteria that may be helpful, for example, on quickly defining rules that block new viruses and worms.

Before enabling FPM, some special files called *Protocol Header Definition Files* (.phdf) need to be loaded on the router. These files can be obtained from the Cisco Connection Online (CCO) website.

Example 11-20 shows how to load the PHDF files and teaches how to display the contents of one of them. It also shows what filtering options become available inside a **class-map**, for IP and UDP, as soon as files of this kind get loaded.

Example 11-20 Basic Information About Flexible Packet Matching

```

! Loading the PHDF files
IOS-FW1(config)# load protocol flash:ip.phdf
IOS-FW1(config)# load protocol flash:tcp.phdf
IOS-FW1(config)# load protocol flash:udp.phdf
IOS-FW1(config)# load protocol flash:icmp.phdf
!
! Displaying the details of a PHDF file
IOS-FW1# show protocols phdf udp
Protocol ID: 3

```

```

Protocol name: UDP
Description: UDP-Protocol
Original file name: flash:udp.phdf
Header length: 8
Constraint(s):
Total number of fields: 5
    Field id: 0, source-port, UDP-Source-Port
        Fixed offset. offset 0
        Constant length. Length: 16
    Field id: 1, dest-port, UDP-Destination-Port
        Fixed offset. offset 16
        Constant length. Length: 16
    Field id: 2, length, UDP-Packet-Length
        Fixed offset. offset 32
        Constant length. Length: 16
    Field id: 3, checksum, UDP-Checksum
        Fixed offset. offset 48
        Constant length. Length: 16
    Field id: 4, payload-start, UDP-Payload-Start
        Fixed offset. offset 64
        Constant length. Length: 0
!
! Viewing available fields for match operations

IOS-FW1(config)# class-map type access-control match-all CLASS1
IOS-FW1(config-cmap)# match field IP ?
checksum          IP-Header-Checksum
dest-addr         IP-Destination-Address
flags            IP-Fragmentation-Flags
fragment-offset   IP-Fragmentation-Offset
identification    IP-Identification
ihl              IP-Header-Length
length           IP-Packet-Length
protocol         IP-Protocol
source-addr      IP-Source-Address
tos             IP-Type-Of-Service
ttl             IP-TTL
version          IP-Version
IOS-FW1(config-cmap)# match field UDP ?
checksum         UDP-Checksum
dest-port        UDP-Destination-Port
length          UDP-Packet-Length
source-port      UDP-Source-Port

```

Figure 11-11 registers a reference packet that is conceived to illustrate the behavior of FPM. The figure also presents two possible variants of this pseudo attack that might be caught with the same FPM rule.

Reference Packet for FPM Testing

Ethernet Packet: 80 bytes
Dest Addr: 0012.DAD2.6203, Source Addr: 0000.0000.0000
Protocol: 0x0800
IP Version: 0x4, HdrLen: 0x5, TOS: 0x40 (Prec=Immediate)
Length: 66, ID: 0x5208, Flags-Offset: 0x0000
TTL: 60, Protocol: 6 (TCP), Checksum: 0x2EC6 (OK)
Source: 172.16.210.105, Dest: 172.16.211.31
TCP Src Port: 8000, Dest Port: 600
Seq #: 0x00000000, Ack #: 0x00000000, Hdr_Len: 5
Flags: 0x02 SYN, Window: 0, Checksum: 0xB9B3 (OK)
Urgent Pointer: 0
Data:
0 : 0000 0000 0000 0000 0000 0000 0001 0108 7468 6531 the1
20 : 774F 526D 3275 wORm2u

Variant 1 (changing only the Data Portion)

Data:
0 : 0000 0000 0000 0000 0000 0000 0001 0108 774F 526D wORm
20 : 4167 6169 6E31 Again1

Variant 2 (changing only the Data Portion)

Data:
0 : 0000 0000 0000 0000 0000 0000 0001 0108 7468 656E then
20 : 6577 574F 524D ewWORM

Figure 11-11 Reference IP Packet for Flexible Packet Matching Analysis

Suppose for a while that the new threat is characterized in the following way:

- The existence of the string ‘wORm’ on the data portion of destination port TCP/600.
- The string can be present in any position from byte 16 to 25 of the TCP payload area (considering that it starts on byte 0).

Figure 11-12 shows the test topology and the configuration structure for FPM, focusing on the relationships between policy-maps and class-maps. It is critical to observe that

- A class-map of type stack matches the TCP protocol on top of IP. This special type of class-map describes which headers should be considered in the packet.

- A **class-map** of **type access-control** looks for the string that defines the threat on a given TCP port (600 in this particular example). The match process starts on a predefined position in the TCP data area and has a configurable search depth (10 bits in this case).
- FPM uses two **policy-maps** of **type access-control**, which are in charge of specifying the actions to be taken when the **class-map** is matched.

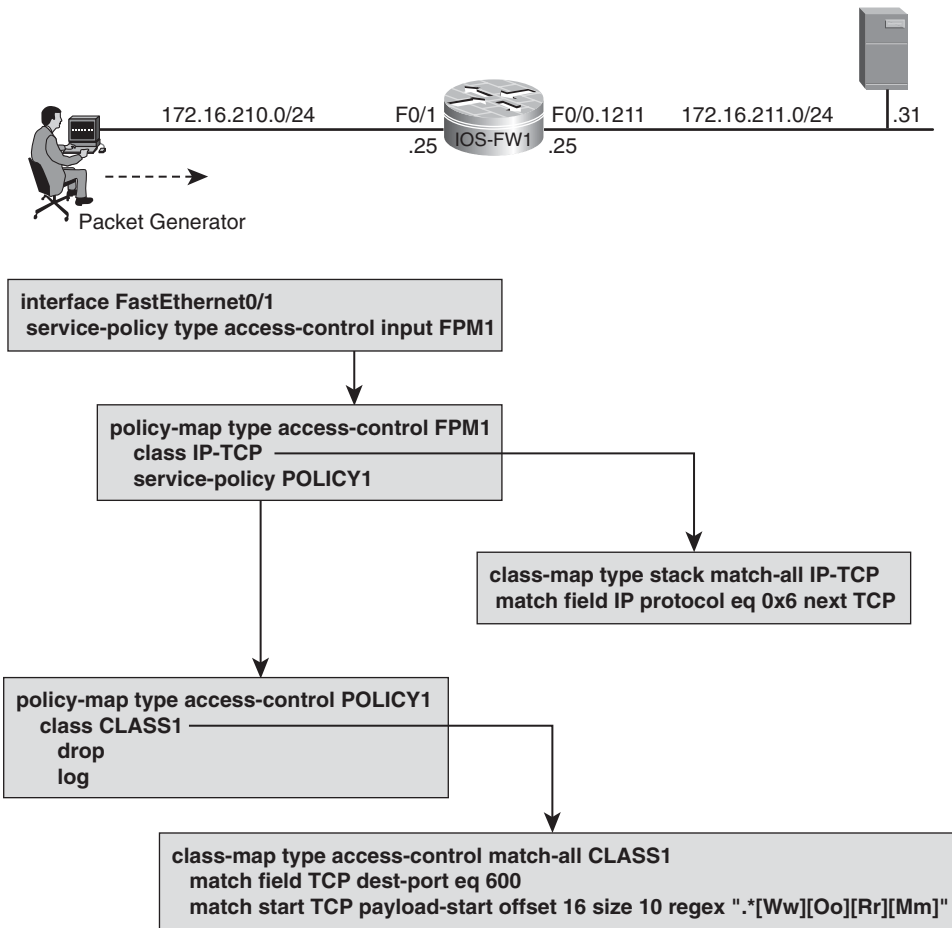


Figure 11-12 *Reference Topology for Flexible Packet Matching Analysis*

Example 11-21 documents the whole FPM configuration for the scenario in Figure 11-12 and registers the appropriate commands to verify the operation of this feature.

The **regex** defined in Example 11-21 can catch all the variants of the pseudo worm presented in Figure 11-11.

Example 11-21 *Flexible Packet Matching Example*

```

class-map type stack match-all IP-TCP
  match field IP protocol eq 0x6 next TCP
!
class-map type access-control match-all CLASS1
  match field TCP dest-port eq 600
  match start TCP payload-start offset 16 size 10 regex ".*[Ww][Oo][Rr][Mm]"
!
policy-map type access-control POLICY1
  class CLASS1
    drop
    log
!
policy-map type access-control FPM1
  class IP-TCP
    service-policy POLICY1
!
interface FastEthernet0/1
  service-policy type access-control input FPM1
!
! FPM in action ('debug fpm events' and ACL log)

fpm_cce_feature_action(): Drop FO - 0x665A572C
fpm_cce_feature_action(): Log FO 0x655A77DC
%SEC-6-IPACCESSLOGP: list CLASS1 denied tcp 172.16.210.105(8002)
(FastEthernet0/1 ) -> 172.16.211.31(600), 1 packet
!
IOS-FW1# show policy-map type access-control interface f0/1 input
FastEthernet0/1
  Service-policy access-control input: FPM1
    Class-map: IP-TCP (match-all)
      4 packets, 400 bytes
      5 minute offered rate 0 bps
      Match: field IP protocol eq 0x6 next TCP
      Service-policy access-control : POLICY1
        Class-map: CLASS1 (match-all)
          4 packets, 400 bytes
          5 minute offered rate 0 bps
          Match: field TCP dest-port eq 600
          Match: start TCP payload-start offset 16 size 10 regex
".*[Ww][Oo][Rr][Mm]"
            drop
            log
          Class-map: class-default (match-any)

```

```

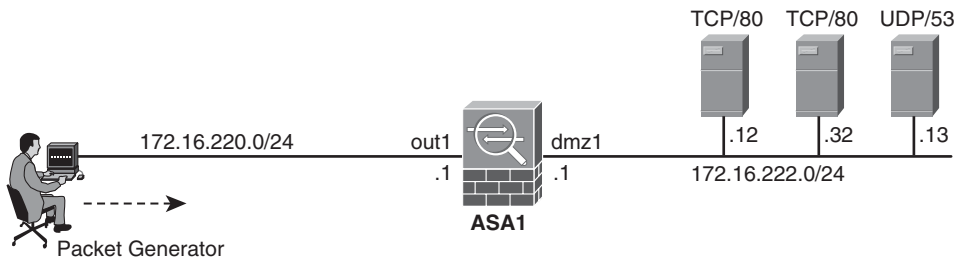
    0 packets, 0 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match: any
Class-map: class-default (match-any)
    0 packets, 0 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match: any

```

Time-Based ACLs

Time-based ACLs employ the **time-range** parameter with Access Control Entries, to activate rules on certain hours of the day or days of the week. This resource, available on both ASA and IOS, is useful on environments that have varying network usage demands according to some time-oriented information.

Figure 11-13 shows the reference topology for most of the tests in this chapter that involve ASA. The baseline topology for IOS analysis is shown in Figure 11-6.



Baseline ACL for most tests involving ASA

```

access-list OUT1 extended permit tcp 172.16.220.0 255.255.255.0 host 172.16.222.32 eq www
access-list OUT1 extended permit tcp 172.16.220.0 255.255.255.0 host 172.16.222.12 eq www
access-list OUT1 extended permit tcp 172.16.220.0 255.255.255.0 host 172.16.222.32 eq telnet
access-list OUT1 extended permit udp 172.16.220.0 255.255.255.0 host 172.16.222.13 eq domain
access-list OUT1 extended permit ip 172.16.220.0 255.255.255.0 host 172.16.222.13
access-list OUT1 extended deny tcp any any
access-list OUT1 extended deny udp any any
access-list OUT1 extended deny icmp any any
access-list OUT1 extended deny ip any any
!
access-group OUT1 in interface out1

```

Figure 11-13 Reference Topology for Tests Involving ASA

Note Some reliable source of synchronization (such as NTP) is indispensable for this feature to work properly. For a review of NTP operations on IOS or ASA, refer to Chapter 3, “Configuration Fundamentals.”

Time-Based ACLs on ASA

Example 11-22 assembles basic information about ASA time-range functionality. Some important topics documented in this example follow:

- ASA supports absolute and periodic time-ranges.
- Time-ranges can be used on ACEs with or without the **log** option.
- The **inactive** keyword appears at the end of a given ACE while the **time-range** has not become active.
- The **show time-range** command reveals what the active **time-ranges** are and whether each of them is used (on an Access Control Entry, for instance).

Example 11-23 complements Example 11-22 by showing **time-ranges** in action:

- It presents two simulations using the Packet Tracer tool: One in which the test packet is blocked because the **time-range** was inactive and a second one that enables the packet through after the **time-range** becomes active.
- It shows a sequence of connection setup attempts through ASA, performed with a packet generator (as shown in Figure 11-13). The connections are allowed through only after the appropriate **time-range** becomes active. (It is instructive to compare the hash values in the Syslog Messages with those in the ACEs).
- The **show asp drop** command unveils the packet drops related to ACL activity.

Example 11-22 Basic Information About time-range on ASA

```
! Two sample periodic time-range definitions
time-range PERIODIC1
  periodic daily 4:00 to 8:00
!
time-range PERIODIC2
  periodic Wednesday Friday 1:00 to 6:00
  periodic weekend 1:00 to 18:00
!
! Sample absolute time-range
time-range TIME1
  absolute start 08:25 27 April 2010
!
! Using the time-range parameter in the ACE definition without the 'log' option
ASA1# show access-list OUT1 | include time-
access-list OUT1 line 5 extended permit ip 172.16.220.0 255.255.255.0
host 172.16.222.13 time-range TIME1 (hitcnt=0) (inactive) 0xaeafd6cd0
!
! Adding the 'log' option to a time-based ACE
ASA1# show access-list OUT1 | include time-
```

```

access-list OUT1 line 5 extended permit ip 172.16.220.0 255.255.255.0
host 172.16.222.13 log informational interval 300 time-range TIME1
(hitcnt=0) (inactive) 0xeafd6cd0
!
! Viewing status of time-ranges

ASA1# show clock detail
08:22:18.965 BRT Tue Apr 27 2010
Time source is NTP
!
ASA1# show time-range
time-range entry: PERIODIC1 (inactive)
    periodic daily 4:00 to 8:00
time-range entry: PERIODIC2 (inactive)
    periodic Wednesday Friday 1:00 to 6:00
    periodic weekend 1:00 to 18:00
time-range entry: TIME1 (inactive)
    absolute start 08:25 27 April 2010
used in: IP ACL entry

```

Example 11-23 ASA time-ranges in Action

```

! Packet-tracer simulation while the time-range is inactive
ASA1# packet-tracer input out1 tcp 172.16.220.100 2000 172.16.222.13 22
Phase: 1
Type: FLOW-LOOKUP
[ output suppressed ]
Phase: 3
Type: ACCESS-LIST
Subtype: log
Result: DROP
Config:
access-group OUT1 in interface out1
access-list OUT1 extended deny tcp any any
Additional Information:
[ output suppressed ]
Action: drop
Drop-reason: (acl-drop) Flow is denied by configured rule
%ASA-4-106023: Deny tcp src out1:172.16.220.100/2000 dst dmz1:172.16.222.13/22 by
access-group "OUT1" [0x5884c962, 0x0]
!
! Packet-tracer simulation after the time-range 'TIME1' becomes active

ASA1# packet-tracer input out1 tcp 172.16.220.100 2000 172.16.222.13 22
Phase: 1

```

```

Type: FLOW-LOOKUP
[ output suppressed ]
Phase: 3
Type: ACCESS-LIST
Subtype: log
Result: ALLOW
Config:
access-group OUT1 in interface out1
access-list OUT1 extended permit ip 172.16.220.0 255.255.255.0 host
172.16.222.13 log time-range TIME1
[ output suppressed ]
Phase: 6
Type: FLOW-CREATION
Subtype:
Result: ALLOW
Config:
Additional Information:
New flow created with id 1709469, packet dispatched to next module
[ output suppressed ]
Action: allow
%ASA-6-302013: Built inbound TCP connection 1709469 for out1:172.16.220.100/2000
(172.16.220.100/2000) to
dmz1:172.16.222.13/22 (172.16.222.13/22)
!
! Time-range transitions to active state during packet generation session

%ASA-4-106023: Deny tcp src out1:172.16.220.237/4111 dst dmz1:172.16.222.13/22 by
access-group "OUT1" [0x5884c962, 0x0]
%ASA-4-106023: Deny tcp src out1:172.16.220.238/4112 dst dmz1:172.16.222.13/22 by
access-group "OUT1" [0x5884c962, 0x0]
%ASA-6-106100: access-list OUT1 permitted tcp out1/172.16.220.239(4113)
-> dmz1/172.16.222.13(22) hit-cnt 1 first hit [0xeafd6cd0, 0x0]
%ASA-6-302013: Built inbound TCP connection 1709451 for out1:172.16.220.239/4113
(172.16.220.239/4113) to dmz1:172.16.222.13/22 (172.16.222.13/22)
%ASA-6-106100: access-list OUT1 permitted tcp out1/172.16.220.240(4114)
-> dmz1/172.16.222.13(22) hit-cnt 1 first hit [0xeafd6cd0, 0x0]
%ASA-6-302013: Built inbound TCP connection 1709452 for out1:172.16.220.240/4114
(172.16.220.240/4114) to dmz1:172.16.222.13/22 (172.16.222.13/22)
!
ASA1# show access-list OUT1 | exclude =0
access-list OUT1; 9 elements; name hash: 0x3adf8b9f
access-list OUT1 line 5 extended permit ip 172.16.220.0 255.255.255.0
host 172.16.222.13 log informational interval 300 time-range TIME1 (hitcnt=15)
0xeafd6cd0
access-list OUT1 line 6 extended deny tcp any any (hitcnt=113) 0x5884c962
!

```

```
ASA1# show asp drop | include acl
```

```
Flow is denied by configured rule (acl-drop)
```

124

Time-Based ACLs on IOS

The concept and behavior of **time-range** on IOS can be deemed identical to that of ASA. The idea behind Example 11-24 is simply to show that IOS **time-ranges** can be associated with many different categories of ACEs (most of them already analyzed throughout this chapter).

Example 11-24 *Employing IOS Time-Ranges on ACE Definitions*

```
! Sample periodic time-range definition on IOS
time-range PERIODIC3
  periodic Saturday 8:00 to 18:00
!
! Sample absolute time-range definition on IOS
time-range TIME2
  absolute end 10:00 21 May 2010
!
! Several types of ACE statements that get activated according to time-ranges
ip access-list extended TIME-BASED
  permit icmp host 172.16.250.250 host 172.16.252.252 option record-route time-range
  PERIODIC3
  permit tcp host 172.16.250.250 host 172.16.252.252 match-all +fin +psh +urg time-
  range PERIODIC3
  permit icmp host 172.16.250.250 host 172.16.252.252 ttl lt 20 time-range PERIODIC3
  permit icmp host 172.16.250.250 host 172.16.252.252 fragments time-range PERIODIC3
  permit icmp host 172.16.250.250 host 172.16.252.252 precedence network time-range
  PERIODIC3
  permit tcp any host 172.16.252.30 eq 443
!
access-list 150 permit tcp host 172.16.250.250 host 172.16.252.252 eq 22 time-range
TIME2 log-input
!
! Viewing time-ranges and the ACLs that employ this resource

IOS-FW# show time-range
time-range entry: PERIODIC3 (inactive)
  periodic Saturday 8:00 to 18:00
  used in: IP ACL entry
  used in: IP ACL entry
  used in: IP ACL entry
  used in: IP ACL entry
  used in: IP ACL entry
time-range entry: TIME2 (active)
```

```

absolute end 10:00 21 May 2010
used in: IP ACL entry
!
IOS-FW# show access-list TIME-BASED
Extended IP access list TIME-BASED
 10 permit icmp host 172.16.250.250 host 172.16.252.252 option record-route
time-range PERIODIC3 (inactive)
 20 permit tcp host 172.16.250.250 host 172.16.252.252 match-all +fin +psh +urg
time-range PERIODIC3 (inactive)
 30 permit icmp host 172.16.250.250 host 172.16.252.252 ttl lt 20 time-range
PERIODIC3 (inactive)
 40 permit icmp host 172.16.250.250 host 172.16.252.252 time-range PERIODIC3
(inactive) fragments
 50 permit icmp host 172.16.250.250 host 172.16.252.252 precedence network time-
range PERIODIC3 (inactive)
 60 permit tcp any host 172.16.252.30 eq 443
!
IOS-FW# show access-list 150
Extended IP access list 150
 10 permit tcp host 172.16.250.250 host 172.16.252.252 eq 22 time-range TIME2
(active) log-input

```

Connection Limits on ASA

Each ASA hardware platform has its own official limit for the number of simultaneous connections. Notwithstanding, ASA administrators can specify admissible connection limits either globally or on a per-class basis.

Example 11-25 highlights the default connection limit (*Conns*) for the ASA 5510. This is the maximum acceptable number at any moment for this particular model. The command **set connection conn-max** is then applied to define a limit of 2000 concurrent connections. The choice of this value is motivated by a practical fact: the capability of my packet generation tool to create the test connections. But this is not a problem because the intent in this section is to illustrate the feature operation instead of focusing on the raw numbers.

Some additional information is documented in Example 11-25:

- A syslog message is issued when the configured limit is reached.
- Although the ACL counters are increased for every connection attempt, the exceeding packets get dropped by the **conn-max** enforcement. There are 8763 hit counts for line 4 of the ACL and 6763 ACL-related drops (refer to the output of the **show asp drop** command).
- The **show service-policy global set connection detail** command also provides insight on configured (and eventually reached) connection limits.

Example 11-25 *Limiting the Global Number of Connections*

```

! Default connection limits for an ASA 5510 appliance
ASA1# show resource usage
Resource                Current      Peak      Limit      Denied Context
Syslogs [rate]          1          503      N/A          0 System
Conns                   1         3012     130000       0 System
Hosts                   2          145      N/A          0 System
!
! Setting the global limit of simultaneous connections to 2000

policy-map global_policy
class class-default
    set connection conn-max 2000
!
ASA1# show service-policy global set connection detail
Global policy:
  Service-policy: global_policy
  Class-map: class-default
  Set connection policy: conn-max 2000
    current conns 0, drop 0
!
! Packets are generated and the configured connection limit of 2000 is reached

%ASA-3-201011: Connection limit exceeded 2000/2000 for output packet from
172.16.220.120/12000 to 172.16.222.13/53 on interface dmz1
!
ASA1# show access-list OUT1 | exclude =0
access-list OUT1; 8 elements; name hash: 0x3adf8b9f
access-list OUT1 line 4 extended permit udp 172.16.220.0 255.255.255.0 host
172.16.222.13 eq domain (hitcnt=8763) 0xacd6e0a0
!
ASA1# show asp drop | include drop|conn
Frame drop:
  Flow is denied by configured rule (acl-drop)                                8
  Connection limit reached (conn-limit)                                       6763
Flow drop:
!
ASA1# show service-policy global set connection detail
Global policy:
  Service-policy: global_policy
  Class-map: class-default
  Set connection policy: conn-max 2000
    current conns 0, drop 6763

```

Note The parameter **embryonic-conn-max** may be used with the **set connection** command to define a limit to the global number of half-open connections.

Figure 11-14 is tightly bound to Example 11-25. The figure shows Adaptive Security Device Manager (ASDM) graphs built while the packet generation tool is directing UDP connection requests through ASA. To better understand these graphs, it is convenient to be aware of the following facts:

- The default timeout for UDP connections is 2 seconds.
- The connection setup rate for the packet generation tool (displayed in Figure 11-13) is 40 connections per second (40 cps).
- The ASDM graphs display data every 10 seconds. With that said, the **conn-max** limit of 2000 is reached after 50 seconds (05 ASDM intervals).

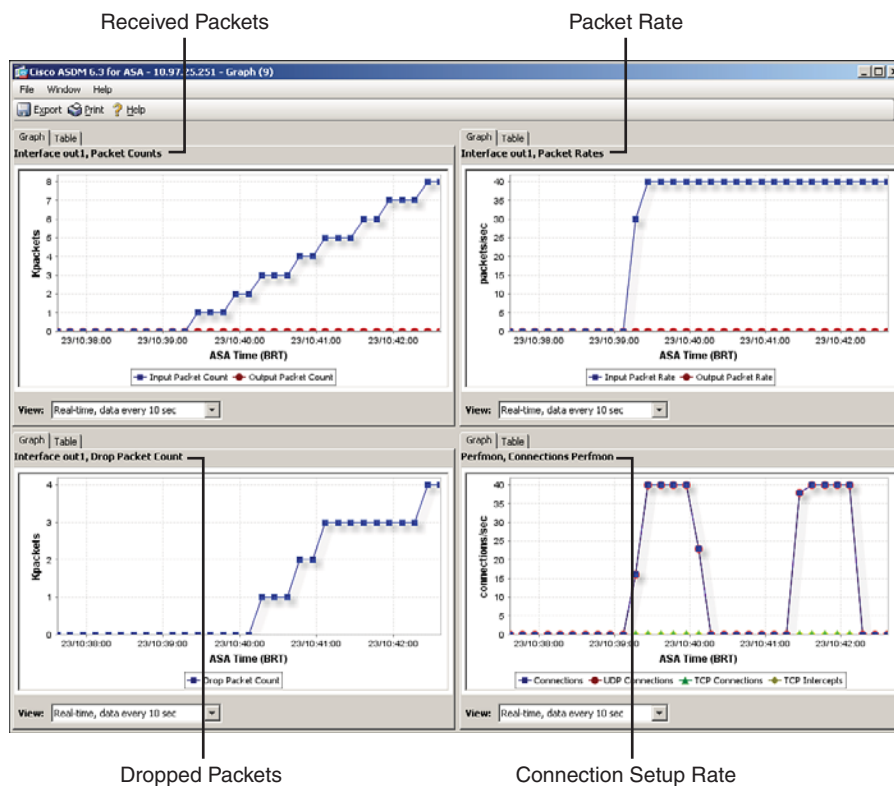


Figure 11-14 ASDM Graph for a Scenario Involving Only UDP Connections

Notice that the *Packet Rate* graph (top right) shows a constant value of 40 packets/second while the *Connection Setup Rate* graph (bottom right) shows a sudden drop in connections after 50 seconds. Two minutes after the start of packet generation, UDP connections start to timeout and make it possible for others to be accepted. Whenever the upper

The graphs on the left concentrate on the number of packets received (and eventually dropped) rather than on packet or connection rates. It is interesting to observe that packet drops start around the time 10:40:20 (coinciding with the null connection admission rate seen on the bottom-right graph).

Figure 11-15 registers two new ASDM graphs dedicated to connection and packet rates for a scenario that has a mix of UDP and TCP connections. The reference topology is still that of Figure 11-13. Some noteworthy details in this new arrangement follow:

- UDP and TCP connection setup rates are respectively 24 cps and 16 cps, summing up to 40 connections per second (and 40 packets per second).
- The UDP and TCP timeouts are set to 2 minutes and 5 minutes, respectively. After 50 seconds, the upper bound is reached and new connections cease to be admitted.
- After 2 minutes, some UDP connections are freed, but because of the larger timeout settings, the TCP-based connections are held. In this new scenario, the connection requests keep arriving in the same ratio (3 UDP/2 TCP) but the connection acceptance rate cannot reach the same value.

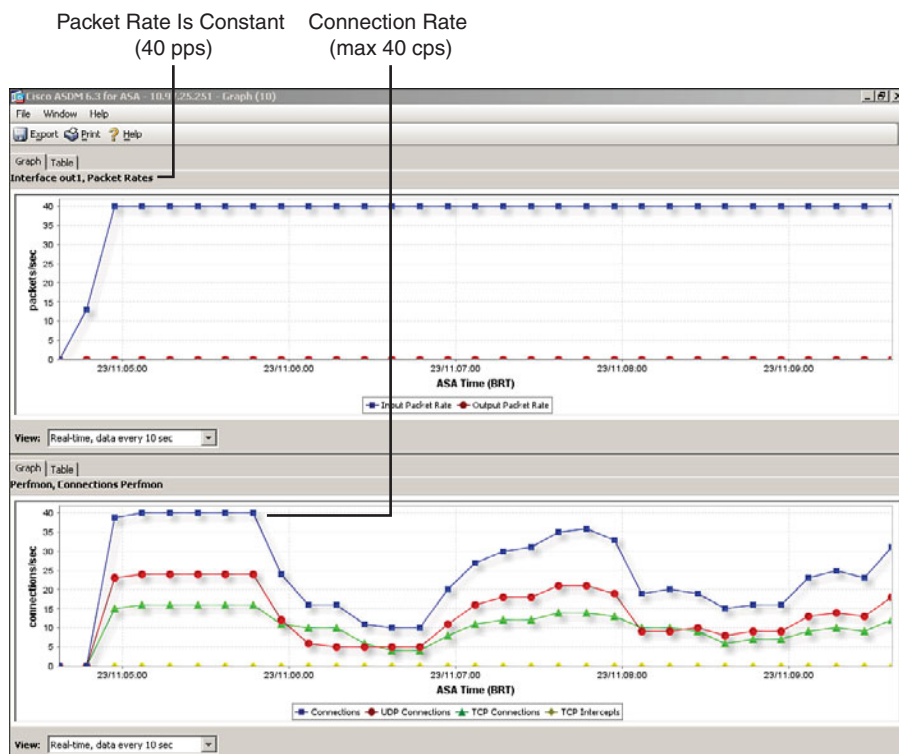


Figure 11-15 ASDM Graph for a Scenario Involving TCP and UDP Connections

Even though this new situation is a bit challenging in terms of calculations, it reproduces more properly a real-life case. (Most of the time there is a combination of TCP and UDP.)

After seeing how to establish global connection limits, it is now time to turn attention to the interesting possibility of limiting connections on a per-client basis, offered by ASA.

Example 11-26 teaches how to set a maximum of per client embryonic connections and additionally documents the verification process. The notable functionality here is the ability to establish a maximum acceptable value *per client*, without the need to know in advance the specific client addresses.

Example 11-26 *Limiting the Per-Client Number of Embryonic Connections*

```

policy-map global_policy
class class-default
  set connection per-client-embryonic-max 10
!
%ASA-6-201012: Per-client embryonic connection limit exceeded 10/10 for
input packet from 172.16.220.122/2316 to 172.16.222.12/80 on interface out1
!
ASA1# show service-policy global set connection detail
Global policy:
  Service-policy: global_policy
  Class-map: class-default
  Set connection policy: per-client-embryonic-max 10
    current conns 870, drop 0
    Per client      Embryonic      Total
    out1      172.16.220.114  10          10
    out1      172.16.220.139  10          10
    out1      172.16.220.122  10          11
    out1      172.16.220.142  10          11

```

Note The option **per-client-max** may be used with the **set connection** command to create a limit for the total number of connections on a per-client basis.

Following this brief study of connection limiting in ASA, it is worth it to recapitulate other techniques that were analyzed in previous chapters:

- Chapter 6, “Virtualization in the Firewall World,” showed how to limit resources (including simultaneous connections and CPS) on a per-context basis.
- Chapter 8, “Through ASA Using NAT,” proposed a method to limit connections for specific IP addresses while performing NAT. This is more suitable to limit connections on the server side (because their addresses are known in advance).

Combining the methods presented in this section with those of previous chapters can offer a solution that can set acceptable connection values either globally or for individual hosts (clients and servers). This group of features helps build a more robust defense against DoS attacks.

Note Example 11-28 presents a way to configure connection limits inside a **class-map**.

TCP Normalization on ASA

The ASA TCP Normalization feature is designed to identify abnormal packets and to bring the possibility of acting on them after detection. Among what is classified as *abnormal* are packets with invalid TCP Acknowledgment numbers, packets conveying data in the SYN-ACK packet, and sequences of packets that present TTL Manipulation. TCP Normalization is on by default but enables customization of some parameters using a configuration entity known as **tcp-map**.

In Example 11-27, a **tcp-map** called *TCPMAP1* is created and the default settings for the ASA Normalizer are displayed. Further, the protective actions enabled by default (such as clearing the URG flag) are highlighted.

Example 11-27 TCP Normalization Options on ASA

```
ASA1(config-tcp-map)# tcp-map TCPMAP1
ASA1(config-tcp-map)# ?
TCP-map configuration commands:
  check-retransmission    Check retransmit data, disabled by default
  checksum-verification   Verify TCP checksum, disabled by default
  default                 Set a command to its defaults
  exceed-mss              Packet that exceed the Maximum Segment Size set by
                           peer, default is to allow packet
  invalid-ack              Packets with invalid ACK, default is to drop packet
  no                      Negate a command or set its defaults
  queue-limit             Maximum out-of-order packets queued for a connection,
                           default is 0 packets
  reserved-bits           Reserved bits in TCP header are set, default is to
                           allow packet
  seq-past-window          Packets that have past-window seq numbers, default is
                           to drop packet
  syn-data                TCP SYN packets that contain data, default is to
                           allow packet
  synack-data             TCP SYN-ACK packets that contain data, default is to
                           drop packet
  tcp-options             Options in TCP header
  ttl-evasion-protection  Protection against time to live (TTL) attacks,
                           enabled by default
  urgent-flag             Urgent flag and urgent offset set, default is to
                           clear flag and offset
  window-variation        Unexpected window size variation, default is to allow
                           connection
```

Example 11-28 shows a sample customization of the normalizer for the **tcp-map** *TCPMAP1*. The configuration process builds upon the Modular Policy Framework. The actions defined in the **tcp-map** follow:

- Validating the TCP checksum.
- Dropping TCP packets whose *reserved bits* are set. The default action is to clear these bits.
- Dropping initial TCP packets (SYN packets) that carry data.
- Dropping packets that contain the TCP options with numbers 6 or 7.

The **tcp-map** is activated by the **set connection advanced-options** command (which is configured at the class level inside a **policy-map**). In this specific example, a class called *CLASS1* is defined within the *global_policy*, but it could have been bound to a **policy-map** at the interface level.

Example 11-28 *Defining a Sample tcp-map*

```
! Configuration structure for a tcp-map

access-list TCP-TRAFFIC1 extended permit tcp 172.16.220.0 255.255.255.0
172.16.222.0 255.255.255.0
!
class-map CLASS1
  match access-list TCP-TRAFFIC1
!
tcp-map TCPMAP1
  checksum-verification
  reserved-bits drop
  syn-data drop
  tcp-options range 6 7 drop
!
policy-map global_policy
  class CLASS1
    set connection conn-max 2000 per-client-embryonic-max 10
    set connection advanced-options TCPMAP1
!
service-policy global_policy global
!
! Viewing details about the new class-map and the associated tcp-map

ASA1# show service-policy set connection
Global policy:
  Service-policy: global_policy
  Class-map: CLASS1
    Set connection policy: conn-max 2000 per-client-embryonic-max 10
```

```

current conns 0, drop 0
Set connection advanced-options: TCPMAP1
Retransmission drops: 0          TCP checksum drops : 0
Exceeded MSS drops : 0          SYN with data drops: 0
Invalid ACK drops : 0          SYN-ACK with data drops: 0
Out-of-order (OoO) packets : 0  OoO no buffer drops: 0
OoO buffer timeout drops : 0    SEQ past window drops: 0
Reserved bit cleared: 0         Reserved bit drops : 0
IP TTL modified : 0            Urgent flag cleared: 0
Window varied resets: 0
TCP-options:
  Selective ACK cleared: 0      Timestamp cleared : 0
  Window scale cleared : 0
  Other options cleared: 0
  Other options drops: 0

```

Note Example 11-28 also shows how to define the connection limits studied in the previous section on a per-class basis.

Example 11-29 shows the customized **tcp-map** in action. The packet generator of Figure 11-13 was used to create TCP connections aimed to explore the parameters modified in Example 11-28. In all the tested cases, the commands **show service policy set connection** and **show asp drop** are used to reveal the reasons behind packet drops.

Example 11-29 *Sample tcp-map in Action*

```

! Dropping TCP connections whose SYN packet contains data
ASA1# show service-policy set connection | include SYN
      Exceeded MSS drops : 0          SYN with data drops: 27
      Invalid ACK drops : 0          SYN-ACK with data drops: 0
!
ASA1# show asp drop | include SYN
      TCP SYN with data (tcp-syn-data)                27
!
ASA1# show access-list OUT1 | exclude =0
access-list OUT1; 9 elements; name hash: 0x3adf8b9f
access-list OUT1 line 2 extended permit tcp 172.16.220.0 255.255.255.0 host
172.16.222.12 eq www (hitcnt=27) 0x863a5b0e
!
ASA1# show access-list TCP-TRAFFIC1
access-list TCP-TRAFFIC1; 1 elements; name hash: 0xbb80021
access-list TCP-TRAFFIC1 line 1 extended permit tcp 172.16.220.0 255.255.255.0
172.16.222.0 255.255.255.0 (hitcnt=27) 0xc7bd5925

```

```

!
! Dropping connections that contain invalid TCP Options

ASA1# show service-policy set connection | begin TCP-options
    TCP-options:
        Selective ACK cleared: 0                Timestamp cleared : 0
        Window scale cleared : 0
        Other options cleared: 0
        Other options drops: 14
        Opt 6: 14
!
ASA1# show asp drop | include option
    TCP option list invalid (tcp-bad-option-list) 14
!
! Dropping connections that contain packets with invalid TCP checksum

ASA1# show asp drop | include bad
    Bad TCP checksum (bad-tcp-cksum) 22
!
ASA1# show service-policy set connection | include checksum
    Retransmission drops: 0                TCP checksum drops : 22
!
! Dropping connections that have the TCP reserved bits set

ASA1# show service-policy set connection | include Reserved
    Reserved bit cleared: 0                Reserved bit drops : 71
!
ASA1# show asp drop | include reserved
    TCP reserved flags set (tcp-reserved-set) 71

```

Note The parameters modified in Example 11-28 and tested in Example 11-29 are by no means a recommendation for real-world scenarios. Before promoting any change in the Normalizer settings, you need to fully understand the applications involved in your particular network environment.

Threat Detection on ASA

The Threat Detection functionality consists of various levels of statistics gathering for many categories of threats. One of its special resources is the capability to identify and optionally shun a host that performs a scanning task.

This section is concerned with statistics pertaining to attack activity for the system as a whole, which is referred to as *Basic Threat Detection*.

Basic Threat Detection is enabled by default and does not have any significant impact on performance. The one restriction to be aware of is that Threat Detection is not available in Multiple Context Mode.

Example 11-30 documents the default Threat Detection settings. For each of the threat classes, there are some predefined rates automatically measured:

- Average drop rate over a 600 seconds interval and burst drop rate over a 10 seconds interval.
- Average drop rate over a 1-hour interval (3600 seconds) and burst rate over a 60 seconds interval.

Example 11-30 keeps the global limit of 2000 simultaneous connections of Example 11-25. The idea, again, is to make life easier for the nonprofessional packet generation tool shown in Figure 11-13.

Example 11-30 *Default threat-detection Settings*

```
ASA1# show running-config all threat-detection
threat-detection rate dos-drop rate-interval 600 average-rate 100 burst-rate 400
threat-detection rate dos-drop rate-interval 3600 average-rate 80 burst-rate 320
threat-detection rate bad-packet-drop rate-interval 600 average-rate 100 burst-rate
400
threat-detection rate bad-packet-drop rate-interval 3600 average-rate 80 burst-rate
320
threat-detection rate acl-drop rate-interval 600 average-rate 400 burst-rate 800
threat-detection rate acl-drop rate-interval 3600 average-rate 320 burst-rate 640
threat-detection rate conn-limit-drop rate-interval 600 average-rate 100 burst-rate
400
threat-detection rate conn-limit-drop rate-interval 3600 average-rate 80 burst-rate
320
threat-detection rate icmp-drop rate-interval 600 average-rate 100 burst-rate 400
threat-detection rate icmp-drop rate-interval 3600 average-rate 80 burst-rate 320
threat-detection rate scanning-threat rate-interval 600 average-rate 5 burst-rate 10
threat-detection rate scanning-threat rate-interval 3600 average-rate 4 burst-rate 8
threat-detection rate syn-attack rate-interval 600 average-rate 100 burst-rate 200
threat-detection rate syn-attack rate-interval 3600 average-rate 80 burst-rate 160
threat-detection rate fw-drop rate-interval 600 average-rate 400 burst-rate 1600
threat-detection rate fw-drop rate-interval 3600 average-rate 320 burst-rate 1280
threat-detection rate inspect-drop rate-interval 600 average-rate 400 burst-rate
1600
threat-detection rate inspect-drop rate-interval 3600 average-rate 320 burst-rate
1280
threat-detection rate interface-drop rate-interval 600 average-rate 2000 burst-
rate 8000
threat-detection rate interface-drop rate-interval 3600 average-rate 1600 burst-
rate 6400
```

```

threat-detection basic-threat
threat-detection statistics port
threat-detection statistics protocol
threat-detection statistics access-list
no threat-detection statistics tcp-intercept
!
! Setting the global upper bound of simultaneous connections to 2000

ASA1# show service-policy global set connection detail | include connection
Set connection policy: conn-max 2000

```

Example 11-31 illustrates the operation of Threat Detection for the *conn-limit* parameter. The default drop rates of Example 11-30 are lowered so that it becomes much easier to exceed them and show the feature in action. It is important to emphasize that the drops start to happen because the global limit of 2000 simultaneous connections is crossed. After this triggering event, as new connection requests arrive they are not allowed through and the Threat Detection feature starts to measure the current drop rate and compare it with the configured rates.

Examples 11-32 and 11-33 are analogous to 11-31, but deal respectively with drops related to a deny statement in an ACL and malformed packets (such as those that have an invalid checksum).

Example 11-31 *Defining a threat-detection Rate for the conn-limit Parameter*

```

! Modifying the default conn-limit-drop rates
threat-detection rate conn-limit-drop rate-interval 600 average-rate 30 burst-rate
60
threat-detection rate conn-limit-drop rate-interval 3600 average-rate 25 burst-rate
50
!
! Burst rate 1 (over the 600 seconds interval) is reached
%ASA-4-733100: [ Connection limit] drop rate-1 exceeded. Current burst
rate is 60 per second, max configured rate is 60; Current average
rate is 11 per second, max
configured rate is 30; Cumulative total count is 7093
!
! Burst rate 2 (over the 3600 seconds interval) is reached
%ASA-4-733100: [ Connection limit] drop rate-2 exceeded. Current burst
rate is 59 per second, max configured rate is 50; Current average
rate is 1 per second, max configured
rate is 25; Cumulative total count is 7093
!
! Dropped connections due to the 'conn-limit' value being reached

ASA1# show service-policy global set connection detail | include conn

```

```

Set connection policy: conn-max 2000
current conns 394, drop 12912
!
ASA1# show asp drop | include conn
Connection limit reached (conn-limit) 12912
!
ASA1# show threat-detection rate conn-limit-drop

```

	Average(eps)	Current(eps)	Trigger	Total events
10-min Conn limit:	21	0	5	12912
1-hour Conn limit:	3	28	1	12912

Example 11-32 Defining a threat-detection Rate for the acl-drop Parameter

```

! Modifying the default acl-drop rates
threat-detection rate acl-drop rate-interval 600 average-rate 50 burst-rate 100
threat-detection rate acl-drop rate-interval 3600 average-rate 40 burst-rate 80
!
! Burst rate 1 (over the 600 seconds interval) is reached
%ASA-4-733100: [ ACL drop] drop rate-1 exceeded. Current burst rate
is 120 per second,
max configured rate is 100; Current average rate is 14 per second, max configured
rate is 50; Cumulative total count is 8959
!
! Burst rate 2 (over the 3600 seconds interval) is reached
%ASA-4-733100: [ ACL drop] drop rate-2 exceeded. Current burst rate is 84 per
second,
max configured rate is 80; Current average rate is 2 per second, max configured
rate is 40; Cumulative total count is 10167
!
! Dropped connections due to acl-drop limit being reached

ASA1# show access-list OUT1 | exclude =0
access-list OUT1; 8 elements; name hash: 0x3adf8b9f
access-list OUT1 line 2 extended permit tcp 172.16.220.0 255.255.255.0 host
172.16.222.12 eq www (hitcnt=971) 0x863a5b0e
access-list OUT1 line 6 extended deny udp any any (hitcnt=11658) 0x5721eb54
!
ASA1# show asp drop | include acl
Flow is denied by configured rule (acl-drop) 11662
!
ASA1# show threat-detection rate acl-drop

```

	Average(eps)	Current(eps)	Trigger	Total events
10-min ACL drop:	19	0	4	11662
1-hour ACL drop:	3	2	1	11662


```
ASA1# show service-policy global set connection detail | include conn
Set connection policy: conn-max 2000
current conns 0, drop 0
```

Example 11-33 *Defining a threat-detection Rate for the bad-packet Parameter*

```
! Modifying the default bad-packet-drop rates
threat-detection rate bad-packet-drop rate-interval 600 average-rate
50 burst-rate 100
threat-detection rate bad-packet-drop rate-interval 3600 average-rate
40 burst-rate 80
!
%ASA-4-733100: [ Bad pkts] drop rate-1 exceeded. Current burst rate
is 120 per second, max configured rate is 100; Current average rate
is 14 per second, max configured rate
is 50; Cumulative total count is 8654
%ASA-4-733100: [ Bad pkts] drop rate-2 exceeded. Current burst rate
is 82 per second, max configured rate is 80; Current average rate is
2 per second, max configured rate
is 40; Cumulative total count is 9862
!
! Drops caused by the bad-packet-rate being reached

ASA1# show asp drop
Frame drop:
  Invalid UDP Length (invalid-udp-length)                10877
  Flow is denied by configured rule (acl-drop)             4
Last clearing: 00:51:54 BRT Apr 24 2010 by enable_15
Flow drop:
Last clearing: 00:51:54 BRT Apr 24 2010 by enable_15
!
ASA1# show threat-detection rate bad-packet-drop

```

	Average(eps)	Current(eps)	Trigger	Total events
10-min Bad pkts:	18	0	4	10877
1-hour Bad pkts:	3	90	1	10877

Summary

This chapter analyzed some protection mechanisms that operate up to Layer 4 of the OSI model and can be used to complement the basic resource of stateful inspection, adding considerable value to the overarching security architecture.

More precisely, this chapter covered the following:

- Two important antispoofing techniques for ASA and IOS: ACLs employed according to RFC 2827 and the unicast Reverse Path Forwarding verification (uRPF).
- Special-purpose IOS ACLs that filter based on specific fields of the IP and TCP headers: TCP Flags, IP options, TTL Field, and IP Fragmentation fields.
- The Virtual Fragmentation Reassembly mechanism, which enables Cisco Network Firewalls (ASA or IOS) to establish a limit of simultaneous fragments being reassembled, to define a maximum number of acceptable fragments per chain and so on.
- A powerful IOS filtering method named Flexible Packet Matching, which enables a filter based on any field of the IP and L4 headers, with search depth defined by the firewall administrator.
- Time-based ACLs.
- How to establish ASA connection limits either at the global level or on a per-client basis.
- TCP Normalization on ASA.
- Threat Detection statistics gathering on ASA.

The next chapter focuses on inspecting application-level protocols on ASA and IOS, complementing the security resources studied so far.

Further Reading

TCP/IP Illustrated, Volume 1: The Protocols (W. Richard Stevens)

<http://www.pearsonhighered.com/educator/product/TCPIP-Illustrated-Volume-1-The-Protocols/9780201633467.page>

TCP/IP Illustrated, Volume 2: The Implementation (Gary R. Wright, W. Richard Stevens)

<http://www.pearsonhighered.com/educator/product/TCPIP-Illustrated-Volume-2-The-Implementation/9780201633542.page>

This page intentionally left blank

Chapter 12

Application Inspection

This chapter covers the following topics:

- Inspection Capabilities in the Classic IOS Firewall
- Application Inspection in the Zone Policy Firewall
- DNS Inspection in the Zone Policy Firewall
- FTP Inspection in the Zone Policy Firewall
- HTTP Inspection in the Zone Policy Firewall
- IM Inspection in the Zone Policy Firewall
- Overview of ASA Application Inspection
- DNS Inspection in ASA
- FTP Inspection in ASA
- HTTP Inspection in ASA
- Inspection of IM and Tunneling Traffic in ASA
- Botnet Traffic Filtering in ASA

“If I have seen farther it is by standing on the shoulders of giants.”—Isaac Newton

Chapters 7, “Through ASA Without NAT” through Chapter 10, “IOS Zone Policy Firewall Overview,” promoted a detailed analysis of generic stateful inspection, the core protection mechanism employed by network firewalls. Generic inspection works fine for well-behaved applications that employ *single channel* sessions but is not sufficient for protocols that either negotiate secondary channels or include IP address information inside the data portion of the IP packet.

Deep Packet Inspection (DPI) is the technique employed by Cisco Firewalls to adapt to the particularities of such application protocols and remove the challenges they would face when crossing a stateless packet filter or a stateful firewall limited to Layer 4.

One bonus that comes from the investment on DPI is that the application knowledge can also be leveraged for more sophisticated filtering activities. Nevertheless, do not lose sight that although HTTP is largely customizable, inspection of most of the protocols does not go that far.

This chapter starts by briefly registering, as a follow-up to Chapter 9, “Classic IOS Firewall Overview,” the original inspection features of the Classic IOS Firewall. These resources are limited and there are no plans for further development. Even if you have not migrated yet to IOS versions that support the Zone Policy Firewall approach, this functionality might still be helpful.

This chapter then shifts to largely exemplifying the Modular Policy Framework (MPF) configuration philosophy, which is similar in IOS Zone-based Policy Firewall and firewalls based on the Adaptive Security Algorithm (ASA).

Chapter 13, “Inspection of Voice Protocols” uses the concepts presented in this chapter with the specific objective of inspecting the protocols that relate to Unified Communications.

Inspection Capabilities in the Classic IOS Firewall

This section extends the Context Based Access Control (CBAC) philosophy initially looked at in Chapter 9 to protocols that need special handling when crossing Layer 3 devices. The Classic IOS Firewall solution essentially focuses on fixing up misbehaved protocols, rather than using application awareness for advanced filtering. The main exception to this rule is HTTP, which somewhat enables customization.

Figure 12-1 depicts the reference scenario for the CBAC examples that follow. Example 12-1 assembles the commands used to implement a policy that permits the setup of outbound DNS, FTP, and HTTP sessions. The **ip inspect** rules are not L4-based as those employed in Chapter 9. They make direct reference to the application protocols themselves, instead of being TCP- or UDP-based. Some additional aspects explored in this example deserve special mention:

- The audit-trail option has been turned on to provide visibility about session setup and termination. When audit-trail comes into play, it shows start http session rather than start tcp session. Compare this output with those shown in Chapter 9.
- The **access-list** named INSIDE is used to limit the protocols enabled to cross the CBAC Firewall. Protocols other than DNS, FTP, and HTTP are blocked by this ACL. Packets permitted by the ACL are then inspected.
- The access-list called OUTSIDE blocks all inbound connections in the topology of Figure 12-1. The permissions for return packets are created on demand in this ACL by means of the **ip inspect** functionality.

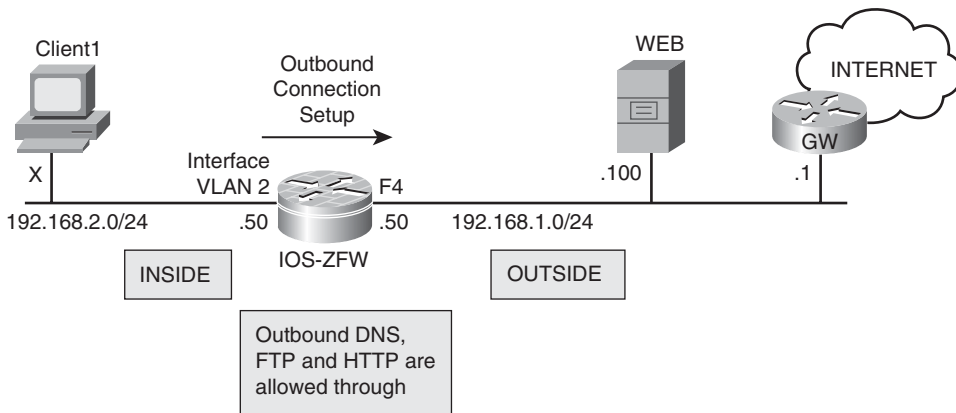


Figure 12-1 Reference Topology for CBAC Application Inspection

Example 12-1 Simple CBAC Policy in Action

```
! CBAC global inspection parameters
ip inspect log drop-pkt
ip inspect tcp block-non-session
!
! Defining a CBAC inspection policy called 'CBAC-L7'
ip inspect name CBAC-L7 ftp alert on audit-trail on
ip inspect name CBAC-L7 http alert on audit-trail on
ip inspect name CBAC-L7 dns alert on audit-trail on
!
! Access-list to define protocols that may be used through the firewall
ip access-list extended INSIDE
 permit tcp 192.168.2.0 0.0.0.255 any eq www
 permit tcp 192.168.2.0 0.0.0.255 any eq ftp
 permit udp 192.168.2.0 0.0.0.255 any eq domain
!
! Access-list to block connections initiated from the outside world
ip access-list extended OUTSIDE
 deny ip any any
!
interface FastEthernet4
 ip address 192.168.1.50 255.255.255.0
 ip access-group OUTSIDE in
!
! Inside interface - incoming packets allowed by the ACL are inspected by CBAC
interface Vlan2
 ip address 192.168.2.50 255.255.255.0
```

```

ip access-group INSIDE in
ip inspect CBAC-L7 in
!
! Sample HTTP connection (setup and termination)

FIREWALL* OBJ_CREATE: Pak 83BCB43C sis 8469CD40
initiator_addr (192.168.2.61:4565) responder_addr (192.168.1.100:80)
initiator_alt_addr (192.168.2.61:4565) responder_alt_addr (192.168.1.100:80)
%FW-6-SESS_AUDIT_TRAIL_START: Start http session: initiator (192.168.2.61:4565) -
responder (192.168.1.100:80)
FIREWALL OBJ_CREATE: sid 846AEDD8 acl OUTSIDE Prot: tcp
Src 192.168.1.100 Port [80:80]
Dst 192.168.2.61 Port [4565:4565]
FIREWALL OBJ_CREATE: create host entry 846814C8 addr 192.168.1.100 bucket 13 (vrf
0:0)
insp_cb 0x84DA54E4
!
IOS-FW# show ip inspect session
Established Sessions
Session 8469CD40 (192.168.2.61:4565)=>(192.168.1.100:80) http SIS_OPEN
!
FIREWALL OBJ_DELETE: delete host entry 846814C8 addr 192.168.1.100
FIREWALL OBJ_DELETE: delete sis 8469CD40
%FW-6-SESS_AUDIT_TRAIL: Stop http session: initiator (192.168.2.61:4565) sent 402
bytes - responder (192.168.1.100:80) sent 1894 bytes
FIREWALL OBJ_DELETE: sid 846AEDD8 on acl OUTSIDE Prot: tcp
Src 192.168.1.100 Port [80:80]
Dst 192.168.2.61 Port [4565:4565]

```

Example 12-2 displays the options available for the **appfw** command. After configuration, the **appfw** policy must be associated with a CBAC inspection rule (which is then bound to a particular firewall interface).

By showing some simple **appfw** definitions in action for HTTP traffic, Example 12-3 becomes the natural companion of Example 12-2. The settings in this example were chosen with the sole objective to easily illustrate application awareness of HTTP. Notice that CBAC gains visibility of information such as the selected HTTP request method or even the URI length. These are typical L7 parameters that are not accessible to the firewall where only generic (L4) inspection is configured.

Example 12-2 Application Firewall Resources Available for CBAC

```

! 'appfw' command and correspondent options for HTTP inspection
IOS-FW(config)# appfw ?
policy-name Name of the Application Firewall policy

```

```

IOS-FW(config)# appfw policy-name APPS1
IOS-FW(cfg-appfw-policy)#?
Application Firewall Policy configuration command:
  application Application to be inspected
  default      Set a command to its defaults
  exit         Exit from appfw policy configuration mode
  no          Negate a command or set its defaults
IOS-FW(cfg-appfw-policy)# application ?
  http HTTP application inspection
  im   Instant Messaging application inspection
IOS-FW(cfg-appfw-policy)# application http
IOS-FW(cfg-appfw-policy-http)#?
HTTP Policy configuration commands:
  audit-trail          Application HTTP audit-trail
  content-length       Specify the range of content length
  content-type-verification Content-type inspection
  default              Set a command to its defaults
  exit                Exit from http-policy configuration mode
  max-header-length    Maximum header size inspection
  max-uri-length       Maximum URI length inspection
  no                  Negate a command or set its defaults
  port-misuse          HTTP port misuse inspection
  request-method       Request method inspection
  strict-http          Strict HTTP Compliance
  timeout              Application HTTP Timeout
  transfer-encoding    Transfer Encoding inspection

```

Example 12-3 Building a Simple Application Firewall Policy with CBAC

```

! Sample appfw policy for use with CBAC
appfw policy-name APPS1
  application http
    max-uri-length 40 action reset alarm
    port-misuse default action reset alarm
    request-method rfc get action allow alarm
    audit-trail on
ip inspect name CBAC-L7 appfw APPS1
!
! Registering the usage of the "GET" request method

06:09:43: %FW-6-SESS_AUDIT_TRAIL_START: Start http session: initiator
(192.168.2.63:1079) -- responder (192.168.1.100:80)
06:09:43: %APFW-4-HTTP_REQ_METHOD_RFC: Sig:3 HTTP RFC method illegal - 'GET' from
192.168.2.63:1079 to 192.168.1.100:80

```



```

!
! Blocking a request containing an URI longer than the allowed value

06:17:39: %FW-6-SESS_AUDIT_TRAIL_START: Start http session: initiator
(192.168.2.63:1108) -- responder (200.221.31.136:80)
06:17:39: %APFW-4-HTTP_MAX_URI_LEN: Sig:12 HTTP URI length exceeded.
Received 48 byte of URL - Reset - URI length exceeded from
192.168.2.63:1108 to 200.221.31.136:80

```

Application Inspection in the Zone Policy Firewall

Having paid tribute to CBAC in the introductory section, it is now time to concentrate on the Zone Policy Firewall (ZFW), the recommended method for deploying IOS-based Firewall solutions.

Top-level **class-maps** and **policy-maps**, which constitute the main building blocks for the ZFW Modular Policy Framework (MPF), were examined in Chapter 10 in the context of generic stateful inspection. To go a bit further and gain L7 visibility, some new concepts need to be introduced:

- **Application-specific class-maps:** These allow traffic classification based on the attributes of a certain application protocol. They are configured by combining the **class-map type inspect** command with a keyword that defines the application protocol under analysis. For instance, the statement **class-map type inspect http HTTP1** is used to classify packets using criteria that pertain to the HTTP protocol.
- **Application-specific policy-maps:** These are configured with the **policy-map type inspect** command and a keyword that refers to a given application protocol. This special type of **policy-map** is used to define actions for each application-specific **class-map** and complements (as a *child policy*) the **inspect** action under a top-level **policy-map**.

Figure 12-2 summarizes the MPF configuration structure as follows:

- Application-specific policy-maps determine the actions that should be taken by the ZFW (reset, log, allow) for each of its application-specific class-maps.
- A top-level **policy-map** defines the appropriate actions for each top-level **class-map**. An application-specific **policy-map** may be optionally applied (with the aid of the **service-policy** command) under each top-level **class-map**.
- The top-level **policy-map** is then attached to a **zone-pair** to implement a zone-based policy. An application-specific **policy map** cannot be directly associated to a **zone-pair**.

Figure 12-2 includes a table that documents not only the supported protocol groups (used in application-specific **policy-maps**) but also the corresponding protocols in each group (used to define application-specific **class-maps**).

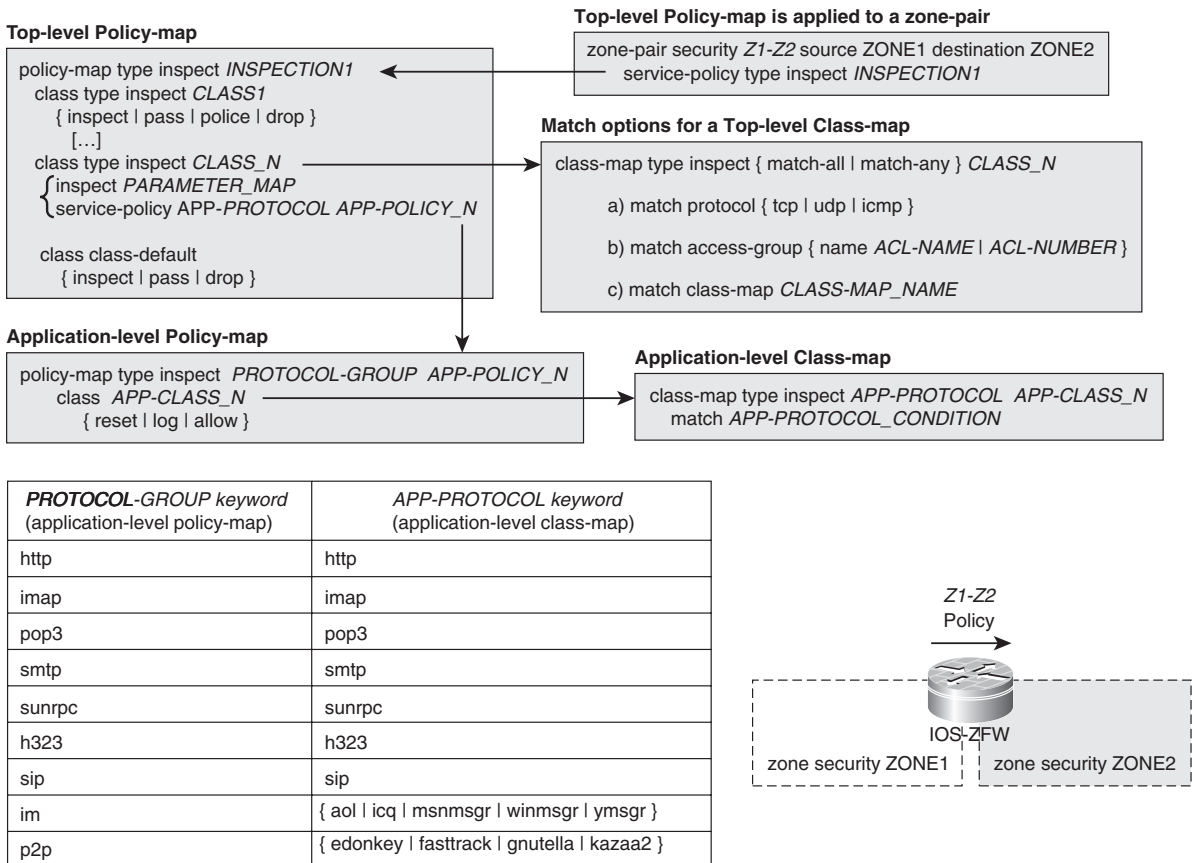


Figure 12-2 *Modular Policy Framework for the Zone Policy Firewall*

Note The ZFW **show** commands used throughout this chapter are those of the 12.4T train. As studied in the final part of Chapter 10, version 15.X offers a new set of commands that have a simpler syntax (**show policy- firewall**). A good exercise, if you already deployed a 15.X release is to repeat the exercises in this chapter using these new commands.

DNS Inspection in the Zone Policy Firewall

Name queries from DNS clients contribute with a large volume of connections to the overall traffic of any typical IP internetwork. Given its importance for the infrastructure of the Internet, it is convenient to differentiate DNS as much as possible from other UDP-based protocols.

IOS enables the DNS timeout to be chosen independently of the global UDP timeout. This approach results in the possibility of rapidly terminating DNS sessions and consequently freeing the connection resources to the protocols that rely on the name resolution services provided by DNS.

Example 12-4 shows the default DNS timeout of 5 seconds, which is much lower than the default value for UDP (and TCP) timeouts (30 seconds). The `dns-timeout` command might be invoked (under a `parameter-map type inspect`) to change the original value of 5 seconds.

Figure 12-3 shows the topology related to Example 12-4 and the corresponding MPF configuration structure for DNS Inspection. DNS is not among the protocols listed in the table of Figure 12-2 and as such does not enable policy actions based on DNS-related parameters.

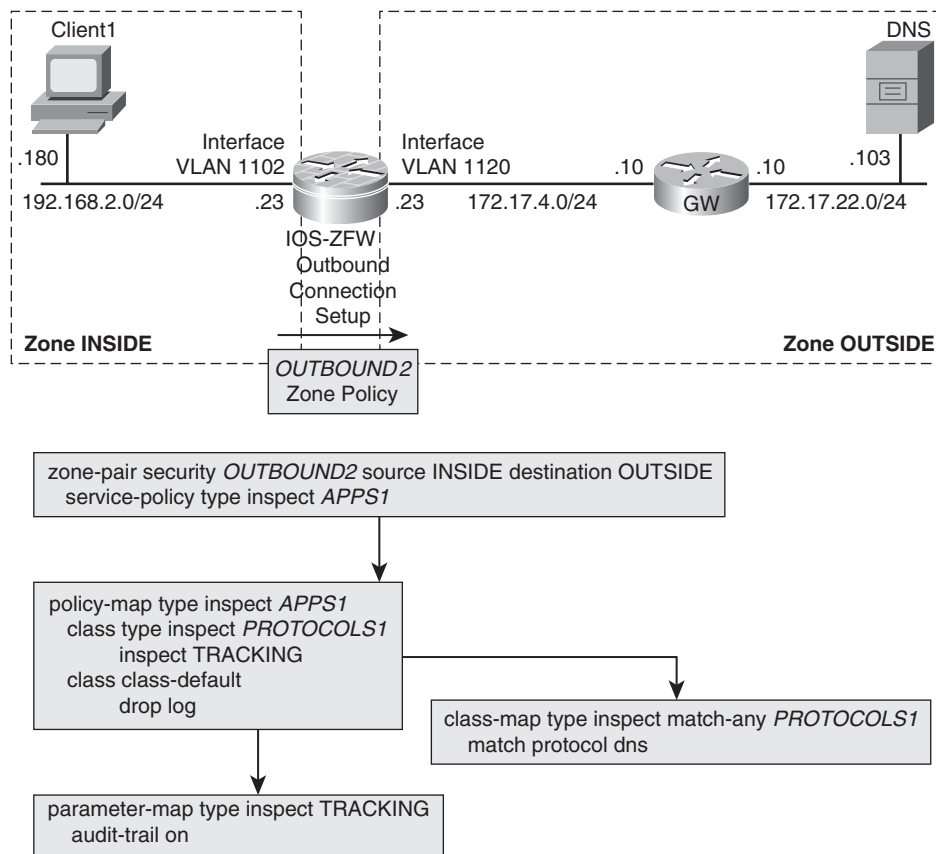


Figure 12-3 Reference Topology for DNS Inspection in the Zone Policy Firewall

Note The **parameter-map** called **TRACKING** shown in Figure 12-3 is referenced throughout the ZFW sections to enable the **audit-trail** function in the application inspection engines. For simplicity, it will be omitted in the remaining ZFW figures.

Example 12-4 Changing the DNS Timeout on ZFW

```
! The default dns-timeout for IOS ZFW is 05 seconds
IOS-ZFW# show parameter-map type inspect default | include dns
      dns-timeout 5
!
! Setup and termination of a DNS session (highlighting timing information)
06:18:54: %FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND2:PROTOCOLS1):
Start dns session: initiator (192.168.2.180:18180) -- responder (172.17.22.103:53)
06:18:59: %FW-6-SESS_AUDIT_TRAIL: (target:class)-(OUTBOUND2:PROTOCOLS1):
Stop dns session: initiator (192.168.2.180:18180) sent 33 bytes -- responder
(172.17.22.103:53) sent 0 bytes
```

FTP Inspection in the Zone Policy Firewall

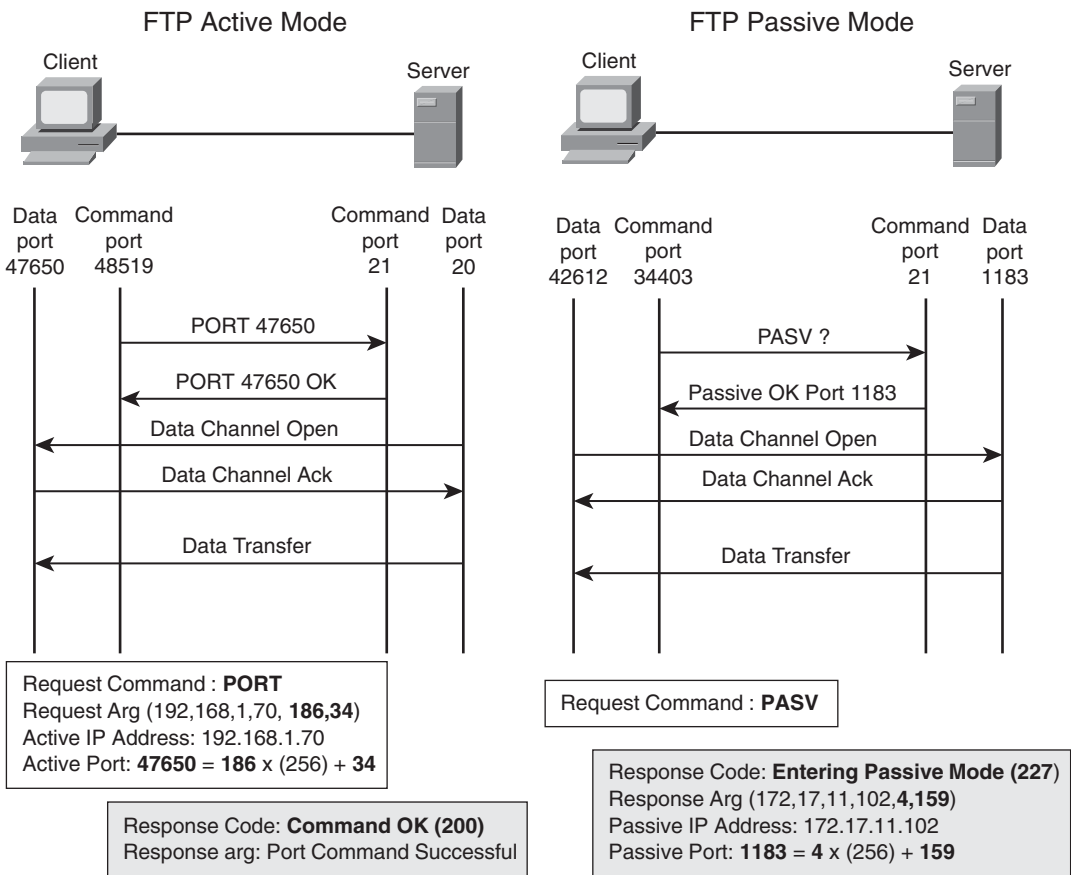
Figure 12-4 revisits the two modes of operation for the File Transfer Protocol (FTP):

- **Active mode:** The client opens a control connection on TCP port 21 and negotiates a data port using the **PORT** command. After accepting the port proposed by the client, the server opens a data connection to it (using TCP/20 as source port).
- **Passive mode:** The client requests Passive mode operation by issuing the **PASV** command over the control connection on TCP port 21. The server suggests a data port, to which the client must connect (using a second source port, randomly selected).

In either operation mode, FTP dynamically selects a data port inside the control channel. If the firewall does not understand the protocol commands, it cannot open the pertinent data channels.

Example 12-5 registers a typical Active FTP session, between the client 192.168.2.72 and the server 172.17.11.102, being established through the Zone-Based Policy Firewall. The detailed visibility provided in this example derives from the usage of the **debug policy-firewall protocol ftp** command.

Although the control session lasts for long, several data connections are dynamically created. (Each FTP command such as *bin*, *ls*, and *get* results in a **PORT** negotiation and in the creation of the associated data session). Another noteworthy fact in the **audit-trail** logs is that the data connections are started by the FTP server (refer to the *initiator* parameter).



```

FIREWALL* sis 8428B960:FTP-Client: PASS xxxxxxx
FIREWALL* sis 8428B960:FTP-Server: 230 Logged on--
FIREWALL* sis 8428B960: User authenticated
FIREWALL* sis 8428B960:FTP-Client: SYST--
FIREWALL* sis 8428B960:FTP-Server: 215 UNIX emulated by FileZilla--
FIREWALL* sis 8428B960:FTP-Client: TYPE I--
FIREWALL* sis 8428B960:FTP-Server: 200 Type set to I--
!
! Client lists directories on the FTP server ( a data-session is created)

FIREWALL sis 84280440:FTP-Client: TYPE A--
FIREWALL sis 84280440:FTP-Server: 200 Type set to A--
FIREWALL* sis 8428B960:FTP-Client: PORT 192,168,2,72,187,150--
FIREWALL* sis 8428B960: Handle PORT command 192.168.2.72:48022
FIREWALL sis 8428B960:FTP-Client: PORT 192,168,2,72,187,150--
FIREWALL sis 8428B960: Handle PORT command 192.168.2.72:48022
FIREWALL* sis 8428B960:FTP-Server: 200 Port command successful--
FIREWALL* sis 8428B960:FTP-Client: LIST--

17:56:36: %FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:PROTOCOLS1):
Start ftp-data session: initiator (172.17.11.102:20) -- responder
(192.168.2.72:48022)
FIREWALL* sis 8428B960:FTP-Server: 150 Opening data channel for directory list.--
FIREWALL* sis 8428B960:FTP-Server: 226 Transfer OK--
17:56:38: %FW-6-SESS_AUDIT_TRAIL: (target:class)-(OUTBOUND1:PROTOCOLS1):
Stop ftp-data session: initiator (172.17.11.102:20) sent 350 bytes -- responder
(192.168.2.72:48022) sent 0 bytes
!
! Client retrieves file 'Reg1.exe' ('get' command) - another data session is
created
FIREWALL* sis 8428B960:FTP-Client: TYPE I --
FIREWALL* sis 8428B960:FTP-Server: 200 Type set to I--
FIREWALL* sis 8428B960:FTP-Client: PORT 192,168,2,72,205,240--
FIREWALL* sis 8428B960: Handle PORT command 192.168.2.72:52720
FIREWALL sis 8428B960:FTP-Client: PORT 192,168,2,72,205,240--
FIREWALL sis 8428B960: Handle PORT command 192.168.2.72:52720
FIREWALL* sis 8428B960:FTP-Server: 200 Port command successful--
FIREWALL* sis 8428B960:FTP-Client: RETR Reg1.exe--
17:57:10: %FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:PROTOCOLS1):
Start ftp-data session: initiator (172.17.11.102:20) -- responder
(192.168.2.72:52720)
FIREWALL* sis 8428B960:FTP-Server: 150 Opening data channel for file transfer.--
FIREWALL* sis 8428B960:FTP-Server: 226 Transfer OK --
17:57:12: %FW-6-SESS_AUDIT_TRAIL: (target:class)-(OUTBOUND1:PROTOCOLS1):
Stop ftp-data session: initiator (172.17.11.102:20) sent 477 bytes -- responder
(192.168.2.72:52720) sent 0 bytes

```

```

!
! Client executes 'quit' command (the FTP control session is closed)

FIREWALL* sis 8428B960:FTP-Client: QUIT --
FIREWALL* sis 8428B960:FTP-Server: 221 Goodbye--
17:57:24: %FW-6-SESS_AUDIT_TRAIL: (target:class)-(OUTBOUND1:PROTOCOLS1):
Stop ftp session: initiator (192.168.2.72:58275) sent 135 bytes -- responder
(172.17.11.102:21) sent 417 bytes

```

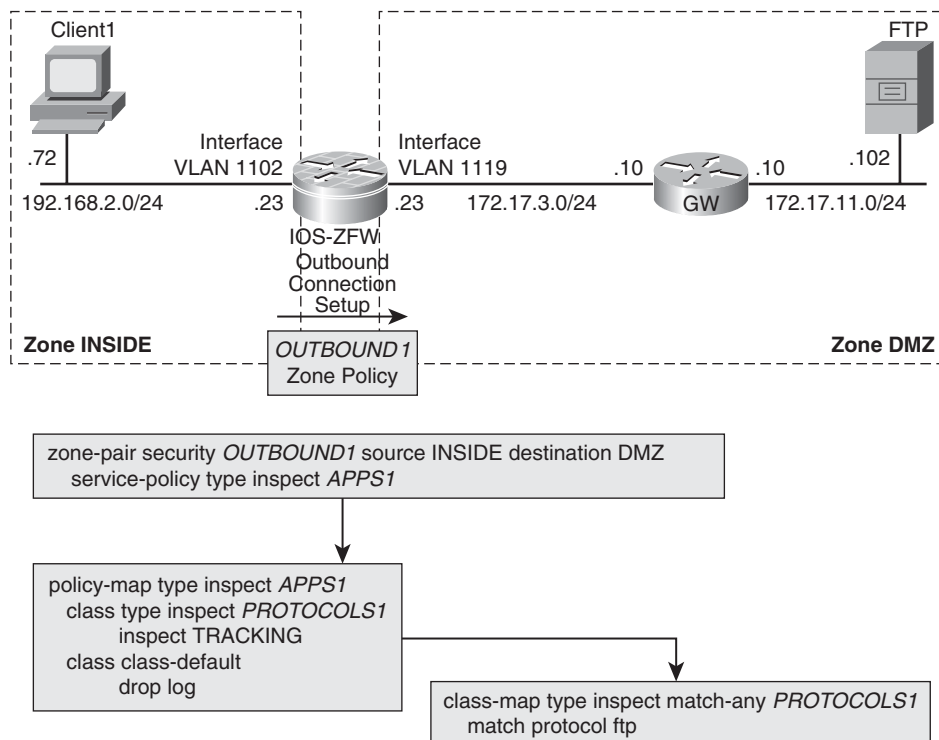


Figure 12-5 Reference Topology for FTP Inspection in the ZFW

Note Although ZFW application knowledge is enough for dynamically opening and closing the data sessions, its FTP inspection capabilities are not customizable.

Example 12-6 illustrates a situation in which regular NAT functionality interacts with the FTP application-level inspection provided by the ZFW. The underlying motivation for this example is to show that the ZFW inspection engine translates the IP address that is carried inside FTP, thus complementing the Layer 3 translation performed by the NAT

feature. Egress and Ingress Netflow not only provide a straightforward method of viewing the IP addresses in each side of the firewall but also unveil the L4 ports being used ($0x14 = 20$ and $0x15 = 21$).

Figure 12-6 depicts the topology associated with Example 12-6 and includes the server-side NAT configuration (`ip nat outside source static` command).

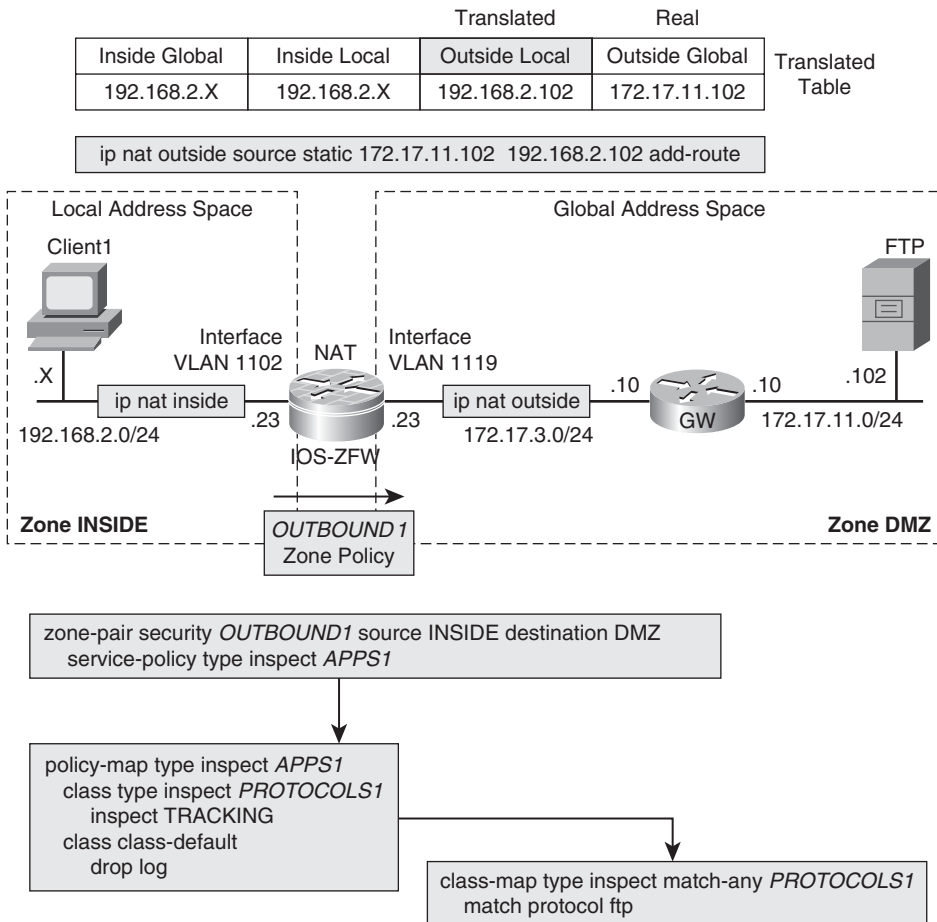


Figure 12-6 FTP Inspection with NAT in the Zone Policy Firewall

Example 12-6 FTP Inspection, NAT, and Flow Accounting

```
! Applying ZFW, NAT and Netflow definitions to relevant interfaces
interface Vlan1102
 ip address 192.168.2.23 255.255.255.0
 ip flow egress
 ip nat inside
```



```

zone-member security INSIDE
!
interface Vlan1119
 ip address 172.17.3.23 255.255.255.0
 ip flow ingress
 ip nat outside
 zone-member security DMZ
!
! FTP Client initiates session to 192.168.2.102 (translated address of FTP server)
1d22h: %IPNAT-6-CREATED: tcp 192.168.2.72:36886 192.168.2.72:36886 192.168.2.102:21
172.17.11.102:21
1d22h: %FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:PROTOCOLS1):
Start ftp session: initiator (192.168.2.72:36886) -- responder (172.17.11.102:21)
!
! Each data connection creates a TCP-based NAT entry
1d22h: %IPNAT-6-CREATED: tcp 192.168.2.72:51974 192.168.2.72:51974 192.168.2.102:20
172.17.11.102:20
1d22h: %FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:PROTOCOLS1):
Start ftp-data session: initiator (172.17.11.102:20) -- responder
(192.168.2.72:51974)
1d22h: %FW-6-SESS_AUDIT_TRAIL: (target:class)-(OUTBOUND1:PROTOCOLS1):
Stop ftp-data session: initiator (172.17.11.102:20) sent 350 bytes -- responder
(192.168.2.72:51974) sent 0 bytes
!
! Displaying active flows just after retrieving a file using FTP

IOS-ZFW# show ip cache flow | begin Src

```

SrcIf	SrcIPaddress	DstIf	DstIPaddress	Pr	SrcP	DstP	Pkts
Vl1119	192.168.2.102	Vl1102*	192.168.2.72	06	0014	9B2A	844
Vl1119	172.17.11.102	Null	192.168.2.72	06	0014	9B2A	853
Vl1119	172.17.11.102	Null	192.168.2.72	06	0015	9016	4

```

!
! Viewing statistics for a given Zone-based Policy

IOS-ZFW# show policy-map type inspect zone-pair OUTBOUND1
policy exists on zp OUTBOUND1
Zone-pair: OUTBOUND1
Service-policy inspect : APPS1
Class-map: PROTOCOLS1 (match-any)
  Match: protocol ftp
    1 packets, 40 bytes
    30 second rate 0 bps
Inspect
  Packet inspection statistics [process switch:fast switch]
  tcp packets: [554:849]
  Session creations since subsystem startup or last reset 3

```

```

Current session counts (estab/half-open/terminating) [1:0:0]
Maxever session counts (estab/half-open/terminating) [2:1:1]
Last session created 00:00:07
Last statistic reset 00:01:21
Last session creation rate 3
Maxever session creation rate 3
Last half-open session total 0
Class-map: class-default (match-any)
  Match: any
  Drop
    0 packets, 0 bytes

```

HTTP Inspection in the Zone Policy Firewall

Although HTTP is not on its own a *misbehaved* protocol, its virtual omnipresence as the enabler of web-based access justifies a special treatment. For instance, the HTTP engine can limit the request methods employed, establish the acceptable request and response headers, or even look for patterns inside the response body, just to name a few possibilities. The engine is also useful for detecting applications that try to disguise their presence inside HTTP, such as Instant Messengers.

This section registers a set of examples that illustrate some of the inspection mechanisms available for HTTP in the ZFW. After becoming acquainted with these resources, they are be revisited in next section with the specific goal to block IM applications.

Example 12-7 refers to the network shown in Figure 12-7. In this case, the **ip port-map** command was employed to instruct IOS to promote HTTP inspection on nonstandard TCP ports 2002 and 2003. It is not just generic TCP on the specified ports but rather flows on such ports are treated as true HTTP.

Example 12-7 Inspecting HTTP on Nonstandard TCP Ports

```

! Requests to TCP ports 2002 and 2003 directed to 172.17.11.102 should be deemed
HTTP
access-list 1 permit 172.17.11.102
ip port-map http port tcp from 2002 to 2003 list 1
!
IOS-ZFW# show ip port-map http
Default mapping:  http          tcp port 80          system defined
Host specific:   http          tcp port 2002-2003   in list 1   user defined
!
! Standard HTTP port and user-defined HTTP ports are inspected by this ZFW policy

class-map type inspect match-any PROTOCOLS1
  match protocol http
!

```

```

policy-map type inspect APPS1
  class type inspect PROTOCOLS1
    inspect TRACKING
  class class-default
    drop log
!
zone-pair security OUTBOUND1 source INSIDE destination DMZ
  service-policy type inspect APPS1
!
! HTTP Sessions on TCP port 2002 (rather than generic TCP sessions)

FIREWALL* sis 84294160: Session Created
FIREWALL* sis 84294160: Pak 83CBFCFC
init_addr (192.168.2.70:2468) resp_addr (172.17.11.102:2002)
init_alt_addr (192.168.2.70:2468) resp_alt_addr (172.17.11.102:2002)
FIREWALL* sis 84294160: FO cls 0x84F8EB80 clsgrp 0x10000000, target
0xA0000000, FO 0x849600E0, alert = 1, audit_trail = 1, L7 = http,
PAMID = 5
FIREWALL* sis 84294160: Allocating L7 sis extension L4 = tcp, L7 = http, PAMID = 5

%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-
(OUTBOUND1:PROTOCOLS1):Start http session: initiator
(192.168.2.70:2468) -- responder (172.17.11.102:2002)
!
IOS-ZFW# show policy-map type inspect zone-pair OUTBOUND1
policy exists on zp OUTBOUND1
Zone-pair: OUTBOUND1
Service-policy inspect : APPS1
Class-map: PROTOCOLS1 (match-any)
  Match: protocol http
        10 packets, 280 bytes
        30 second rate 0 bps

```

Tip The `ip port-map` command is also available for use with CBAC policies.

Figure 12-8 registers the network topology and the corresponding MPF configuration structure that first exemplifies the usage of application-specific **class-maps** and **policy-maps**. In this particular case, the L7 attribute that served for packet classification was the presence of the *ser-cookie* header in the HTTP response. The connections matching this protocol condition (for packets coming from the web server) are reset and logged as determined by the application-specific **policy-map** called WEB1. These two actions are performed by the application class HTTP1, as shown in Example 12-8.

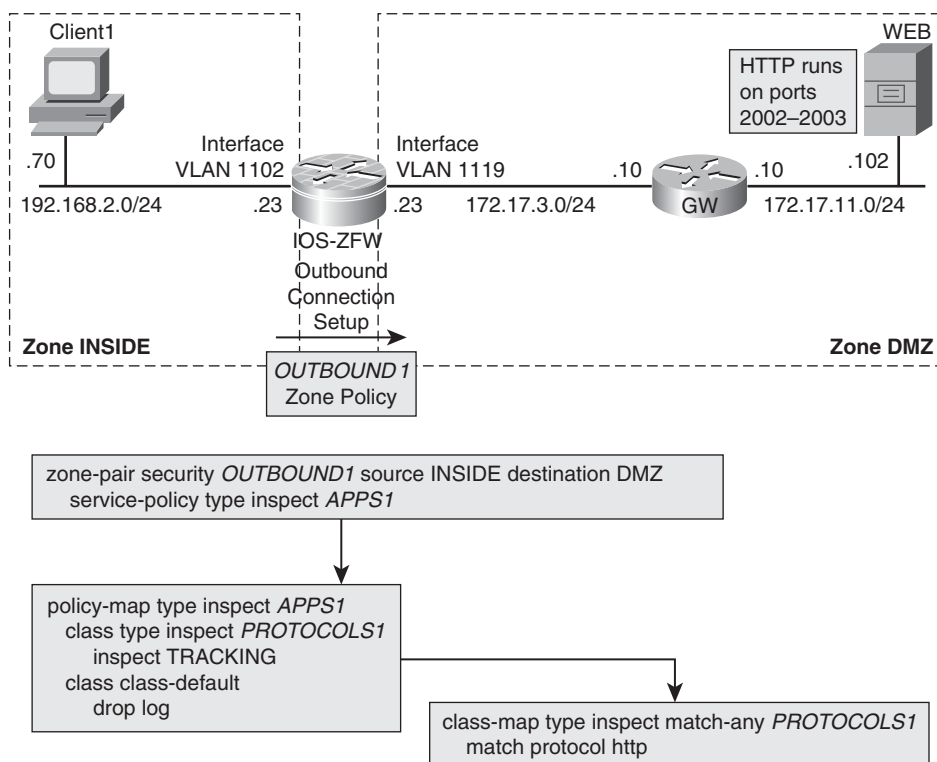


Figure 12-7 HTTP Inspection on Nonstandard TCP Ports

Example 12-8 HTTP Deep Inspection with ZFW - Example 1

```

! Sample session reset and logged by the MPF policy of Figure 12-8
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:HTTP-
CLASS):Start http session: initiator (192.168.2.74:12000) --
responder (172.17.3.151:80)
FIREWALL* sis 84283A40: match-info token in cce_sb 849BA240 - class
3221225494; filter 31; val1 0; val2 0; str set-cookie, log on, reset
on
% APPFW-4-HTTP_HDR_FIELD_REGEX_MATCHED: Header field (set-cookie)
matched - resetting session 172.17.3.151:80 192.168.2.74:12000 on
zone-pair OUTBOUND1 class HTTP-CLASS app1-class HTTP1

```

Example 12-9 builds on the topology shown in Figure 12-8, with just a few modifications in the classification criteria. In this new example, a pattern match against the HTTP *User-agent* request header is performed. Packets classified by the HTTP1 class are then subject to reset and log actions, as established by the **policy-map** WEB1.

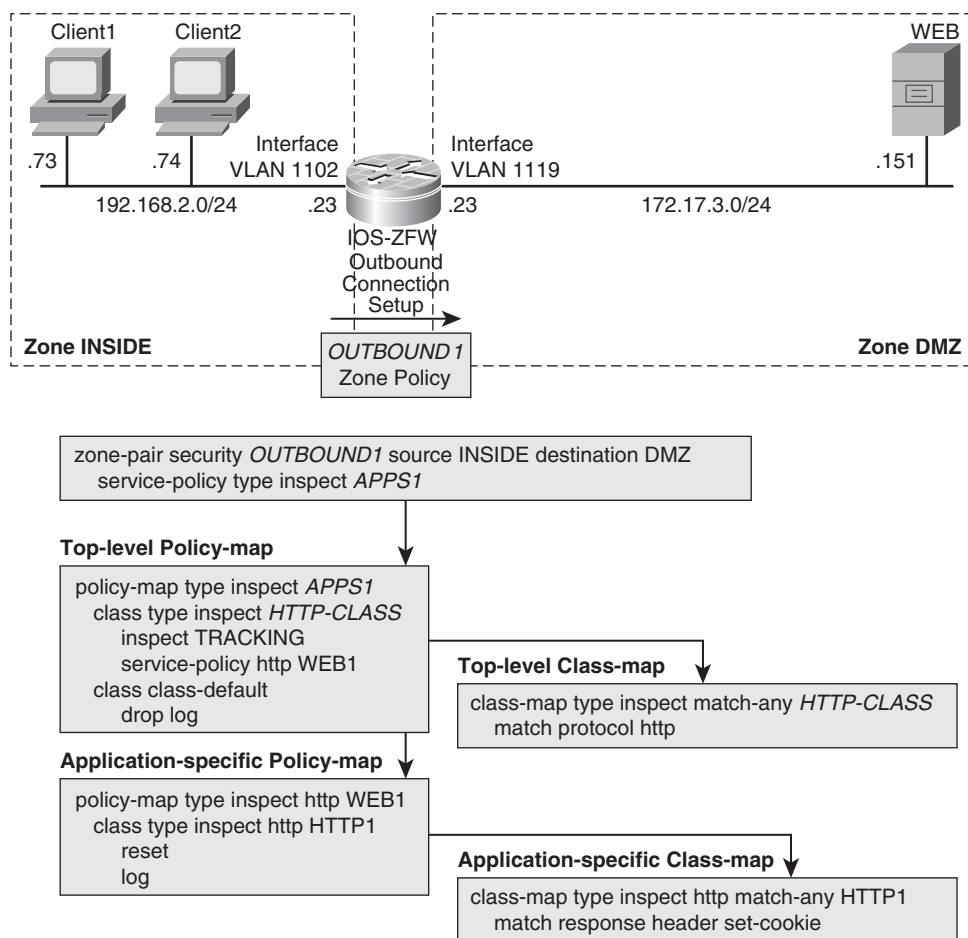


Figure 12-8 Reference Topology for HTTP Deep Packet Inspection

Example 12-9 HTTP Deep Inspection with ZFW - Example 2

```

! Defining a Regular Expression for use within an application-specific class-map
parameter-map type regex AGENT1
pattern .*[0o][Pp][Ee][Rr][Aa]
!
class-map type inspect http match-any HTTP1
match request header user-agent regex AGENT1
!
class-map type inspect match-all HTTP-CLASS
match protocol http
!
policy-map type inspect http WEB1

```

```

class type inspect http HTTP1
  reset
  log
!
policy-map type inspect APPS1
class type inspect HTTP-CLASS
  inspect TRACKING
  service-policy http WEB1
class class-default
  drop log
!
zone-pair security OUTBOUND1 source INSIDE destination DMZ
  service-policy type inspect APPS1
!
FIREWALL* sis 84283240: match-info token in cce_sb 849BA240 -
class 3221225494; filter 34; val1 0; val2 0; str ^[Uu][Ss][Ee][Rr][-
][Aa][Gg][Ee][Nn][Tt]:.*[Oo][Pp][Ee][Rr][Aa], log on, reset on
% APPFW-4-HTTP_HDR_FIELD_REGEX_MATCHED: Header field
(^[Uu][Ss][Ee][Rr][ ][Aa][Gg][Ee][Nn][Tt]:.*[Oo][Pp][Ee][Rr][Aa])
matched - resetting session 192.168.2.73:5000 172.17.3.151:80 on
zone-pair OUTBOUND1 class HTTP-CLASS
appl-class HTTP1

```

Example 12-10 relates to the scenario portrayed in Figure 12-9. The idea is to supplement the material studied so far by providing an inspection policy that involves more protocols simultaneously. Some facts that deserve special mention follow:

- The policy-map WEB-POLICY1 specifies that HTTP requests containing a certain pattern within the URI must be reset and logged. The same policy actions are prescribed for connections that present signs of port-misuse, such as carrying P2P traffic inside HTTP. There are two application-level class-maps inside the application-level policy-map.
- There are two top-level **class-maps** inside the top-level **policy-map** named APPS1. DNS and HTTPS packets are subject to regular inspection, whereas HTTP traffic undergoes Deep Packet Inspection.

Example 12-10 Sample ZFW Policy Involving More Application Protocols

```

! The protocols matched by this class-map are subject to regular inspection
class-map type inspect match-any PROTOCOLS1
  match protocol dns
  match protocol https
!
! HTTP receives a special treatment and is matched by a dedicated class-map
class-map type inspect match-all HTTP-CLASS

```

```

match protocol http
!
! Application-specific class-map for HTTP
class-map type inspect http match-any VIOLATIONS
  match request port-misuse any
!
! Regular Expression matching based on the URI parameter

parameter-map type regex YAHOO-SITES
  pattern .*[.][Yy][Aa][Hh][Oo][Oo][.]
!
class-map type inspect http match-any YAH001
  match request uri regex YAH00-SITES
!
! Application-specific policy-map for HTTP

policy-map type inspect http WEB-POLICY1
  class type inspect http YAH001
    log
    reset
  class type inspect http VIOLATIONS
    log
    reset
!
! Top-level policy-map and corresponding zone-pair definition

policy-map type inspect APPS1
  class type inspect HTTP-CLASS
    inspect
    service-policy http WEB-POLICY1
  class type inspect PROTOCOLS1
    inspect
  class class-default
    drop log
!
zone-pair security OUTBOUND1 source INSIDE destination OUTSIDE
  service-policy type inspect APPS1
!
! ZFW policy in action

% APPFW-4-HTTP_URI_REGEX_MATCHED: URI regex
(*[.][Yy][Aa][Hh][Oo][Oo][.]) matched - resetting
session 192.168.2.61:2195 200.152.168.178:80 on zone-pair OUTBOUND1 class
HTTP-CLASS appl-class YAH001

```

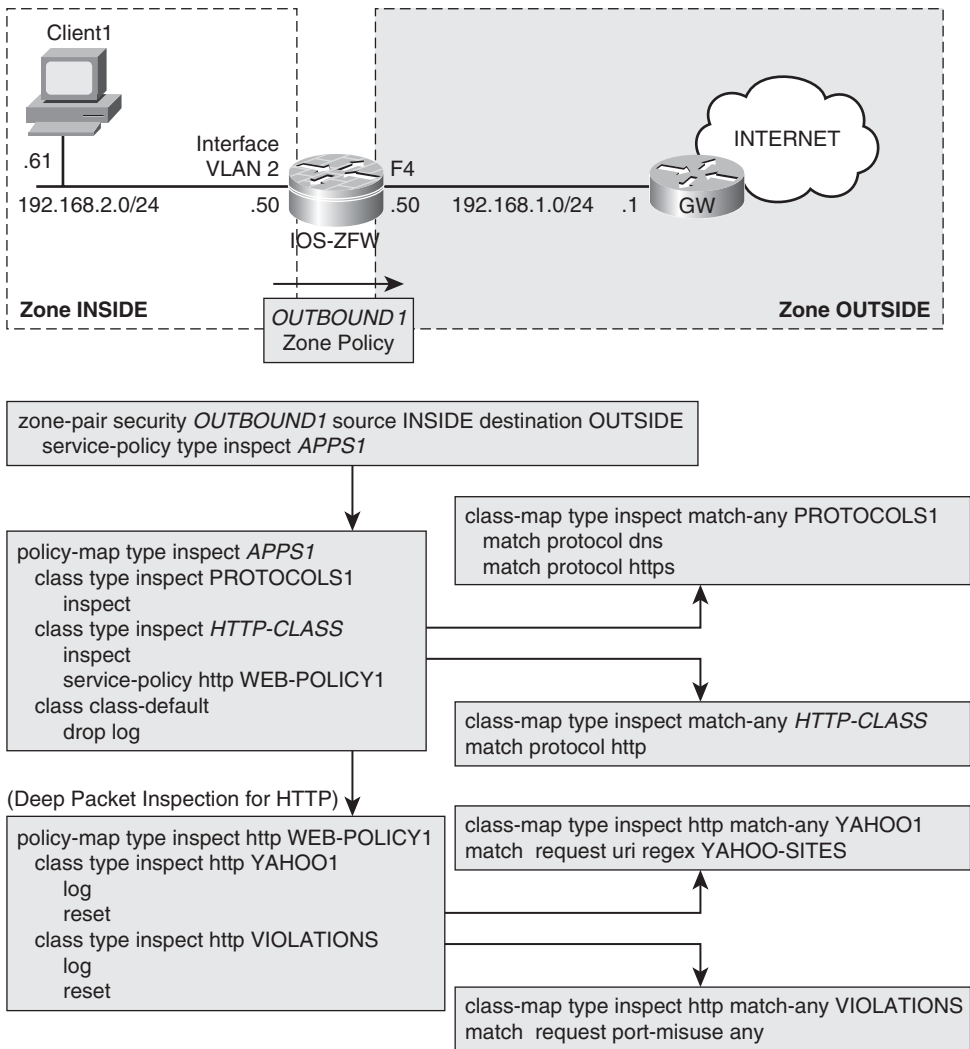


Figure 12-9 Reference Topology for Complex Inspection Policy

Note The Regular expressions presented are merely illustrative. Of course there is no suggestion of any nature against *Yahoo* or *Opera*.

IM Inspection in the Zone Policy Firewall

The topology represented in Figure 12-9 serves as the backdrop for a discussion for the mutable nature of Instant Messaging applications.

Example 12-11 begins by providing a way to visualize the default ports for *Yahoo* and *MSN* messengers. It later employs Netflow to obtain clues about the addresses involved in a *Yahoo Messenger* login sequence. Many of the Yahoo servers are called on TCP ports 80 (0x50) and 443 (0x1BB), meaning that the protocol does not limit its activity to the originally assigned port (TCP/5050).

One possibility of controlling this type of traffic is to filter based on the **server ip** (or **server name**) **protocol-info** inside a **parameter-map**. Such an approach blocks not only the requests directed over port 5050 but also those on ports 80 and 443, as shown in the Syslog messages.

Example 12-11 Blocking Instant Messengers (1)

```
! Default L4 ports used for Yahoo Messenger and MSN Messenger
IOS-FW# show ip port-map ymsgsr
Default mapping: ymsgsr          tcp port 5050          system defined
IOS-FW# show ip port-map msnmsgr
Default mapping: msnmsgr        tcp port 1863          system defined
!
! Some IP Addresses seen when an internal host logs to Yahoo Messenger

IOS-FW# show ip cache flow | begin Src
SrcIf      SrcIPAddress  DstIf      DstIPAddress  Pr SrcP DstP  Pkts
Fa4        209.191.93.250 V12*       192.168.2.61  06 0050 0558   4
Fa4        98.136.48.32  V12*       192.168.2.61  06 0050 054F   1
Fa4        209.191.92.114 V12*       192.168.2.61  06 01BB 0553   6
Fa4        98.137.130.31  V12*       192.168.2.61  06 01BB 0556   4
Fa4        200.221.31.134 V12*       192.168.2.61  06 0050 0564  23
Fa4        200.221.31.136 V12*       192.168.2.61  06 0050 0565   3
Fa4        67.195.187.211  V12*       192.168.2.61  06 0050 0551  10
Fa4        200.152.175.146 V12*       192.168.2.61  06 0050 055E   3
Fa4        98.138.47.195   V12*       192.168.2.61  06 0050 0566   1
[ output suppressed ]
!
C:\Documents and Settings\amoraes.CISCO>ping -a 209.191.93.250
Pinging yts1.ab.vip.mud.yahoo.com [209.191.93.250] with 32 bytes of data:
Reply from 209.191.93.250: bytes=32 time=211ms TTL=56
C:\Documents and Settings\amoraes.CISCO>ping -a 209.191.92.114
Pinging 12.login.vip.mud.yahoo.com [209.191.92.114] with 32 bytes of data:
Reply from 209.191.92.114: bytes=32 time=207ms TTL=55
C:\Documents and Settings\amoraes.CISCO>ping -a 98.136.48.32
```

```

Pinging vcs3.msg.vip.sp1.yahoo.com [98.136.48.32] with 32 bytes of data:
Reply from 98.136.48.32: bytes=32 time=254ms TTL=55
!
parameter-map type protocol-info YAHOO-SRV
  server ip 209.191.92.114
  server ip 209.191.93.250
  server ip 98.136.48.32
  server ip range 67.195.187.200 67.195.187.219
!
class-map type inspect match-any YAHOO-MSG
  match protocol ymsg YAHOO-SRV
!
class-map type inspect match-any PROTOCOLS2
  match protocol http
  match protocol https
  match protocol dns
!
policy-map type inspect APPS2
  class type inspect YAHOO-MSG
    drop log
  class type inspect PROTOCOLS2
    inspect
  class class-default
    drop log
!
zone-pair security OUTBOUND2 source INSIDE destination OUTSIDE
  service-policy type inspect APPS2
!
! Requests to servers in the parameter-map are dropped irrespectively
of dest-ports
%FW-6-DROP_PKT: Dropping im-yahoo session 192.168.2.61:2940
98.136.48.32:80 on zone-pair OUTBOUND2 class YAHOO-MSG due to DROP
action found in policy-map with ip ident 0
%FW-6-LOG_SUMMARY: 1 packet were dropped from 192.168.2.61:2940 =>
98.136.48.32:80 (target:class)-(OUTBOUND2:YAHOO-MSG)
%FW-6-LOG_SUMMARY: 3 packets were dropped from 192.168.2.61:2943 =>
209.191.92.114:443 (target:class)-(OUTBOUND2:YAHOO-MSG)

```

Another method to deny traffic for a given IM is to block access to its default assigned port and look for some kind of protocol signature inside HTTP requests. This approach is registered in Example 12-12 for Yahoo Messenger.

Example 12-12 *Another Approach for Blocking Instant Messengers*

```

parameter-map type regex YAH001
  pattern .*[Yy][Mm][Ss][Gg]
!
class-map type inspect http match-any YAH00-WEB
  match request body regex YAH001
!
policy-map type inspect http WEB-YAH00
  class type inspect http YAH00-WEB
    reset
    log
!
policy-map type inspect APPS1
  class type inspect HTTP-CLASS
    inspect
  service-policy http WEB-YAH00
  class type inspect PROTOCOLS1
    inspect
  class class-default
    drop log
!
zone-pair security OUTBOUND1 source INSIDE destination OUTSIDE
  service-policy type inspect APPS1

```

Overview of ASA Application Inspection

This section outlines the inspection functionality available in ASA, which is built around the Modular Policy Framework (MPF), a structured way to build policies. You have already seen the ASA MPF way of thinking in topics such as the following:

- Instructing the ASA to decrement the TTL field (Chapter 7)
- TCP Sequence Number Randomization (Chapter 7)
- TCP Normalization (Chapter 11, “Additional Protection Mechanisms”)

An ASA-based firewall has many preconfigured inspection settings, which are applied at the global level, as shown in Example 12-13. It is relevant to emphasize that these rules are effective independently of the particular interface where packets arrive.

Example 12-13 also reveals the preconfigured class *inspection_default* acts upon a set of protocol ports grouped as the *default-inspection-traffic*.

The class named *class-default* appears at the end of top-level **policy-maps** and matches any traffic that was not previously classified within the given **policy-map**. It simply instructs ASA not to perform any application-specific inspection action for this remaining traffic.

Example 12-13 *Default Inspection Settings in ASA*

```

! Default service-policy (named 'global_policy' and applied at the global level)
ASA1# show running-config service-policy
service-policy global_policy global
!
! Details about the default policy-map
ASA1# show running-config all policy-map global_policy
policy-map global_policy
  class inspection_default
    inspect dns preset_dns_map
    inspect ftp
    inspect h323 h225 _default_h323_map
    inspect h323 ras _default_h323_map
    inspect netbios
    inspect rsh
    inspect rtsp
    inspect skinny
    inspect esmtp _default_esmtp_map
    inspect sqlnet
    inspect sunrpc
    inspect tftp
    inspect sip
    inspect xdmcp
  class class-default
!
ASA1# show running-config all class-map inspection_default
class-map inspection_default
  match default-inspection-traffic
!
ASA1# show running-config all class-map class-default
class-map class-default
  match any
!
! Displaying default inspection traffic

ASA1(config)# class-map CLASS1
ASA1(config-cmap)# match ?
mpf-class-map mode commands/options:
  access-list          Match an Access List
  any                  Match any packet
  default-inspection-traffic Match default inspection traffic:
                        ctique---tcp--2748      dns-----udp--53
                        ftp-----tcp--21        gtp-----udp--2123,3386
                        h323-h225-tcp--1720      h323-ras--udp--1718-1719

```

http-----tcp--80	icmp-----icmp
ils-----tcp--389	mgcp-----udp--2427,2727
netbios---udp--137-138	radius-acct---udp--1646
rpc-----udp--111	rsh-----tcp--514
rtsp-----tcp--554	sip-----tcp--5060
sip-----udp--5060	skinny---tcp--2000
smtp-----tcp--25	sqlnet---tcp--1521
tftp-----udp--69	waas-----tcp--1-65535
xmcp-----udp--177	

Figure 12-10 summarizes the MPF hierarchy for ASA, whose usage is largely illustrated throughout the rest of this chapter. Some important concepts present in this figure are:

- Deep Packet Inspection is materialized by invoking application-specific **policy-maps**, inside a top-level **class-map**. These special **policy-maps** can establish match conditions for the application protocols represented as a **CONFIGURABLE-PROTOCOL** in the table. The other protocols shown in Example 12-13 (default-inspection-traffic) are not customizable and were referred to as a **FIXED-PROTOCOL** in the schematic drawing of Figure 12-10.
- Although ASA supports application-specific **class-maps** for a handful of protocols (refer to Figure 12-11), these MPF components are not strictly necessary. As shown in Figure 12-10, many **match** criteria (L7-related) are directly supported inside an application-specific **policy-map**.
- The top-level **policy-map** is associated to a particular ASA interface. Even though the interface-oriented **policy-map** takes precedence for connections initiated through this interface, the global policy is still there.

Figure 12-11 brings an Adaptive Security Device Manager (ASDM) screenshot that might be considered a good starting point for the configuration of ASA inspection activities:

- ASDM refers to application-specific **policy-maps** as *inspect-maps*.
- ASDM treats application-specific **class-maps** and *inspect-maps* as policy objects (which might be later reused for policy construction).
- The figure shows both the *global_policy* (default) and a sample user-defined policy. The HTTP-specific policy named WEB1 is part of the top-level **policy-map** called INSPECTION1.
- The **Edit Service Policy Rule** window is shown, highlighting several configuration components related to application inspection.

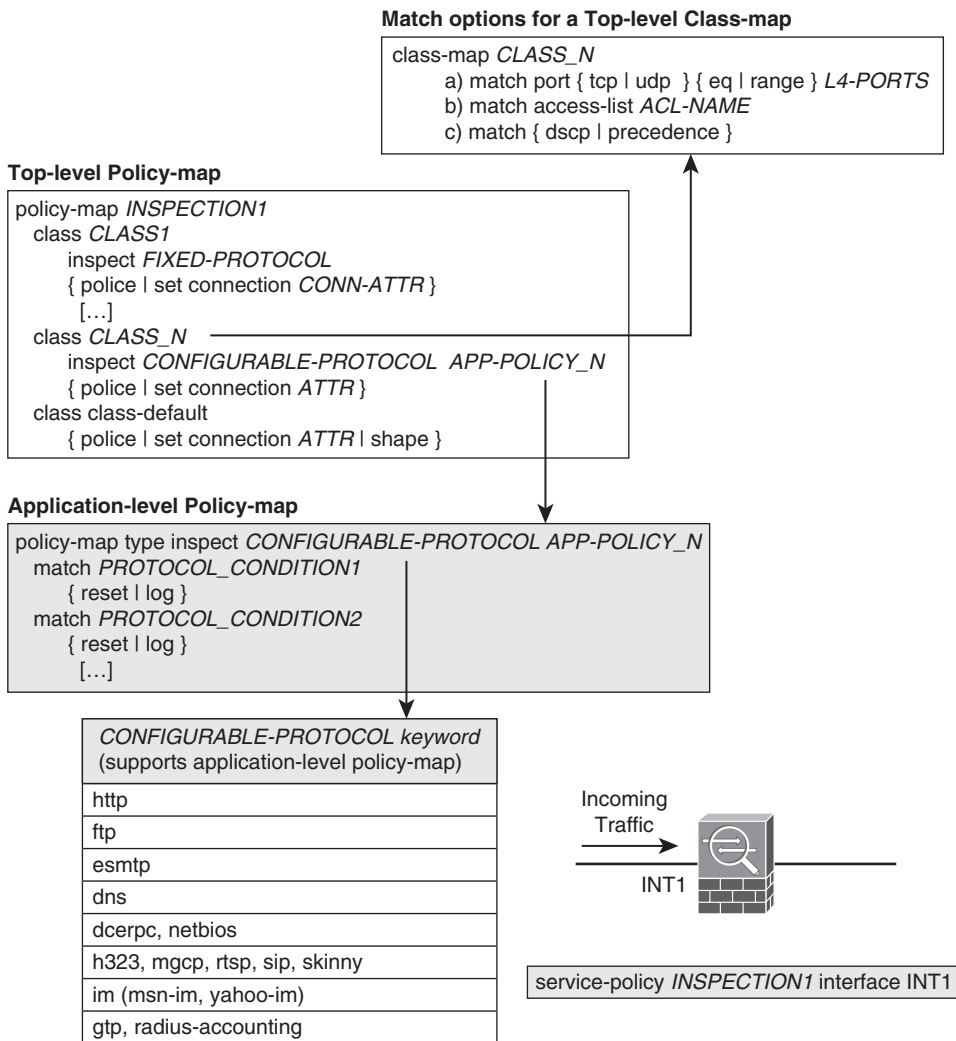


Figure 12-10 Modular Policy Framework for ASA

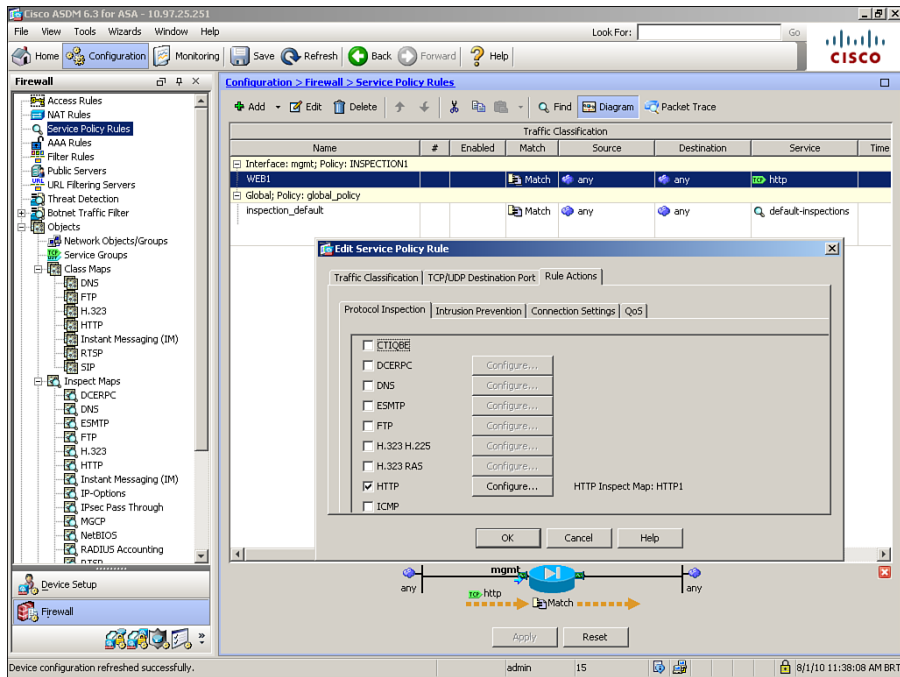


Figure 12-11 ASDM - Path to Service-Policy Definitions

Figure 12-12 shows another ASDM screenshot that enables the configuration of ESMTP inspection attributes. The figure also shows how to access other important policy objects:

- **Regular expressions:** Used inside application-specific policy-maps.
- **TCP-Maps:** Used for TCP Normalization (covered in Chapter 11).

Throughout the rest of the chapter, many ASA inspection examples are presented, but always using the CLI. This quick visit to ASDM was just a humble attempt to provide basic guidance about how to start using this tool for dealing with inspection concepts. One interesting exercise for you is to rebuild the examples presented here with the aid of ASDM.

DNS Inspection in ASA

To better understand ASA's potential to deal with DNS, start by reviewing the main DNS message formats, which are summarized in Figure 12-13.

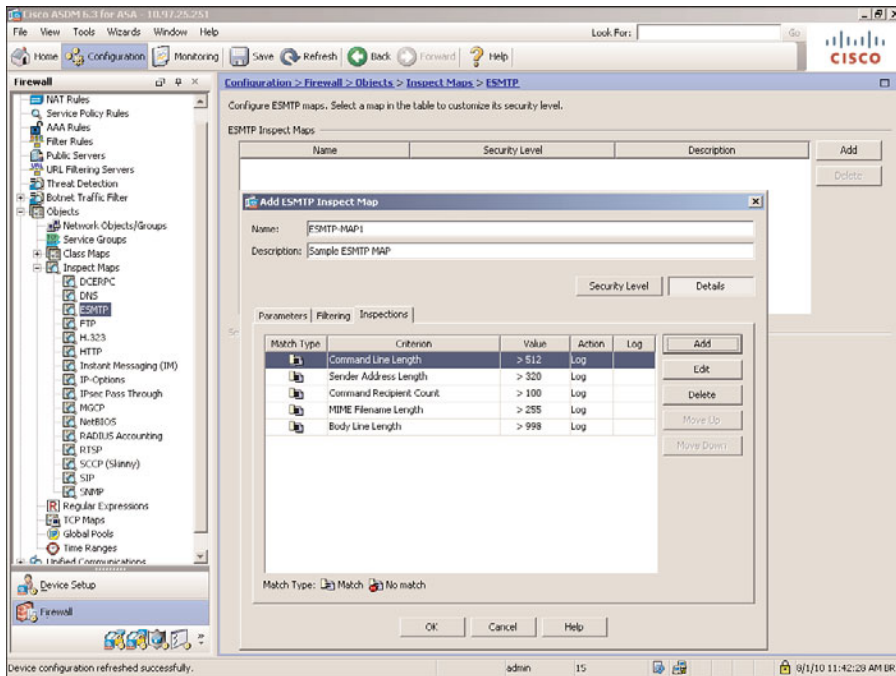


Figure 12-12 ASDM - Displaying Options for the ESMTP Inspect Maps

Figure 12-14 shows a sample DNS response message, which serves to exemplify many of the contents assembled in Figure 12-13. This DNS reply has some noteworthy features:

- The Flags field reveals that QR = 1 and OPCODE = 0, meaning that this is a reply to a standard query.
- The value 1 for bit AA in the Flags field, indicates that this DNS Server is authoritative for the domain.
- The Answer Count field contains a value of 2 because the queried name corresponds to an alias record (CNAME type). One of the answers is the 'A' record (real name) bound to the alias, and the second is the IP address associated to the real name. To be more specific, the queried name *webtop.mylab1.com* is an alias corresponding to *web1.mylab1.com* (real host), which resolves to the IP address 172.17.4.150.

Note Figure 12-15 registers another example of a DNS reply message. In this scenario, the queried name is directly the real name *web1.mylab1.com*, for which an 'A' record exists in the server. The Flags field is identical to that in Figure 12-14, but the Answer Count field now assumes the value 1.

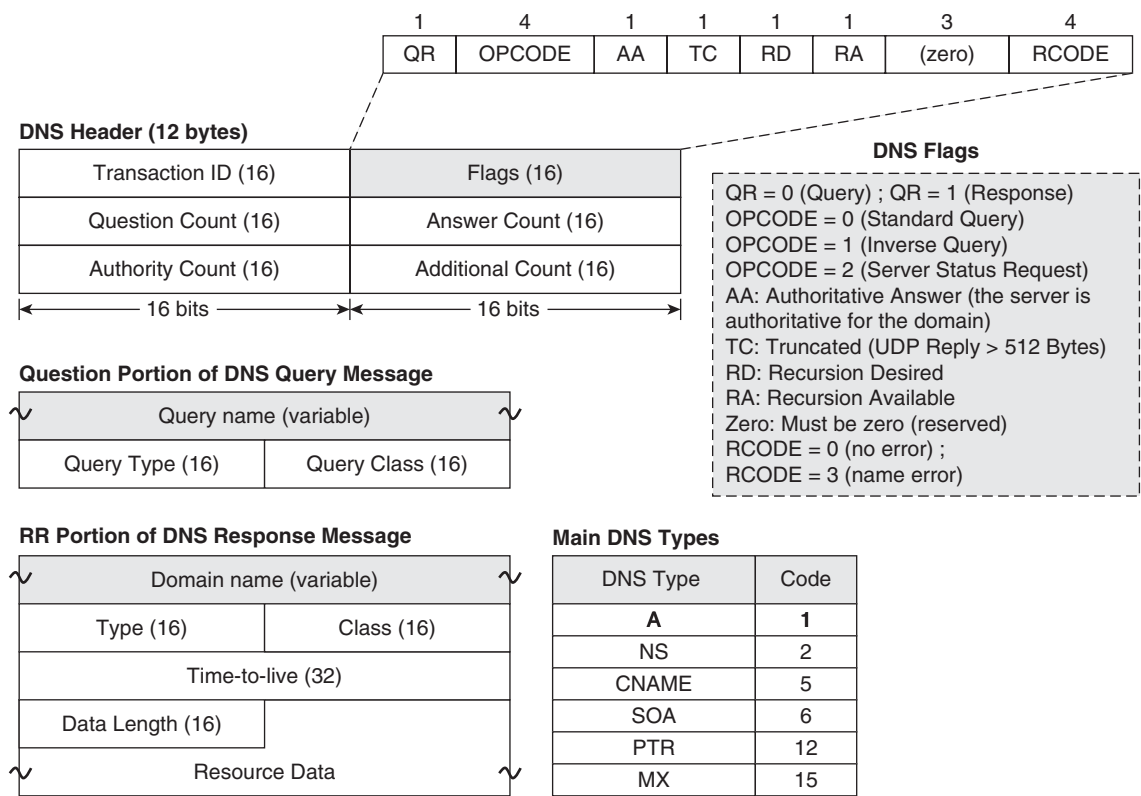


Figure 12-13 The format of DNS Messages

DNS Guard

Example 12-14 registers the setup and termination of a DNS session through ASA, when inspection of this protocol is disabled, meaning that DNS is treated just as any other UDP application. The syslog messages characterize that the regular timeout is 2 minutes.

Given the importance of DNS for the infrastructure of the Internet and the typical amount of connections for this protocol, it is worth it to invest in solutions that can differentiate DNS from other UDP-based applications. A firewall that understands DNS operations can detect that a certain transaction has ended and terminate the underlying UDP connection, without the need to wait 2 minutes (or at least 1 minute, the minimum allowed timeout for generic UDP). L4 connections are freed earlier with such an approach and the resources become available for other application protocols.

With that demand in mind, Cisco created the *DNS Guard* feature, a mechanism that closes a DNS session as soon as a response for a given DNS query is seen by ASA.

Example 12-15 was conceived to illustrate the DNS Guard functionality. First, a **policy-map** of type inspect that logs every question is configured to provide greater visibility of DNS queries and the respective responses. The parameter that ties a query to a response is the Transaction ID, previously shown in Figures 12-13 and 12-14.

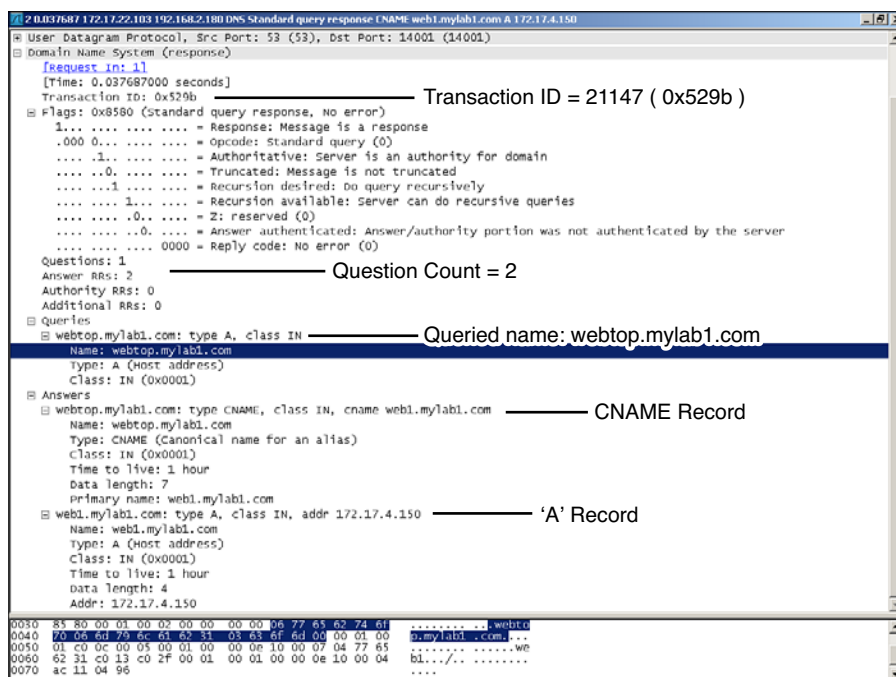


Figure 12-14 Sample DNS Reply Message

Example 12-14 DNS Inspection Disabled

```
! The default timeout for DNS is the UDP timeout of 02 minutes
15:59:14: %ASA-6-302015: Built outbound UDP connection 1752008 for
out:172.17.22.103/53 (172.17.22.103/53) to inside:192.168.2.71/56194
(192.168.2.71/56194)
!
ASA1# show conn detail | begin out
UDP out:172.17.22.103/53 inside:192.168.2.71/56194,
    flags -, idle 1m53s, uptime 1m53s, timeout 2m0s, bytes 82
!
16:01:17: %ASA-6-302016: Teardown UDP connection 1752008 for out:172.17.22.103/53
to inside:192.168.2.71/56194 duration 0:02:02 bytes 82
```

Example 12-15 shows how DNS inspection follows connection creation. Just after receipt of the reply from the DNS server, ASA tears down the UDP connection. The good news is that the elapsed time between DNS request and reply is around 1 second (instead of the original 2 minutes).

Example 12-15 *DNS Inspection and DNS Guard*

```

class-map DNS-UDP
  match port udp eq domain
!
! Policy-map action used to log DNS requests and responses
policy-map type inspect dns DNS1
  match question
    log
!
policy-map INSPECTION2
  class DNS-UDP
    inspect dns DNS1
!
service-policy INSPECTION2 interface inside
!
! Sample session with DNS Guard enabled
16:18:57: %ASA-6-302015: Built outbound UDP connection 1752011 for
out:172.17.22.103/53 (172.17.22.103/53) to inside:192.168.2.180/12000
(192.168.2.180/12000)
16:18:57: %ASA-6-410004: DNS Classification: Received DNS request (id
60675) from inside:192.168.2.180/12000 to out:172.17.22.103/53;
matched Class 27: match question
16:18:57: %ASA-6-410004: DNS Classification: Received DNS reply (id
60675) from out:172.17.22.103/53 to inside:192.168.2.180/12000;
matched Class 27: match question
16:18:58: %ASA-6-302016: Teardown UDP connection 1752011 for
out:172.17.22.103/53 to inside:192.168.2.180/12000 duration 0:00:00
bytes 82
!
ASA1# show service-policy interface inside inspect dns
Interface inside:
  Service-policy: INSPECTION2
    Class-map: DNS-UDP
      Inspect: dns DNS1, packet 4, drop 0, reset-drop 0
        dns-guard, count 2

```

Note DNS Guard comes into play whenever DNS inspection is enabled. As shown in Example 12-13, DNS is part of the default inspection list.

Note The log option configured within Example 12-15 is used throughout the remaining DNS examples. The purpose is to provide better visibility at the application level.

DNS Doctoring

Figure 12-15 depicts an ASA inserted in a three interfaces scenario in which an *inside* client needs to access a Web server (web1.mylab1.com) located at the *dmz* interface. The challenge here is that the DNS server (reachable through the *out* interface) replies with the IP address 172.17.4.150 (the published address of the server *web1* and not its real address 172.17.3.150). ASA permits access only to a published address on a given interface and, as such, a ping from the inside to 172.17.4.150 fails (as shown in Example 12-16).

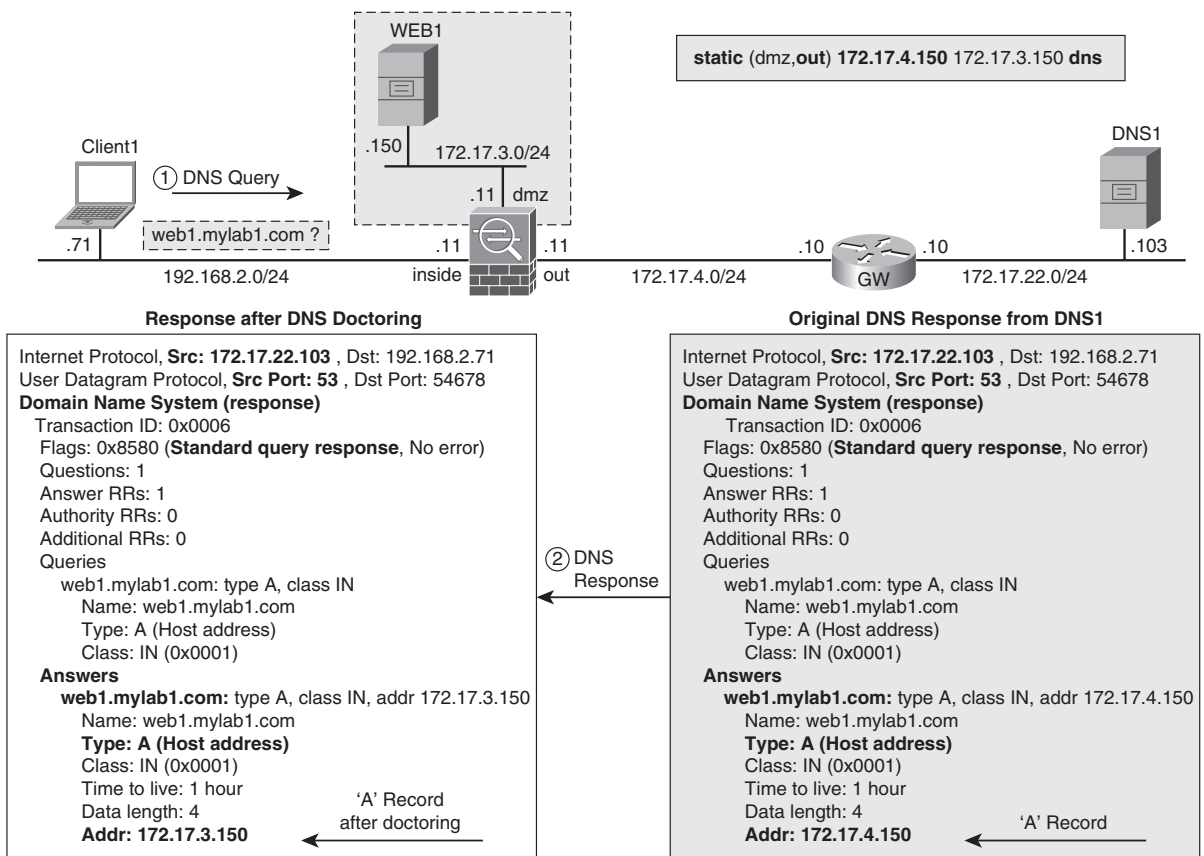


Figure 12-15 DNS doctoring in action

Example 12-16 *Name Resolution Without DNS Doctoring*

```

! Connectivity from inside to 'dmz' uses PAT
nat (inside) 1 192.168.2.0 255.255.255.0
global (dmz) 1 172.17.3.251
!
! Connectivity from 'inside' to 'out' uses static NAT ( 1 to 1 mapping)
static (inside,out) 192.168.2.0 192.168.2.0 netmask 255.255.255.0
!
! Real server address 172.17.3.150 is published on 'out' as 172.17.4.150
static (dmz,out) 172.17.4.150 172.17.3.150 netmask 255.255.255.255
!
! Pings from CLIENT1 to host web1.mylab1.com fail

CLIENT1# ping web1.mylab1.com repeat 2
Translating "web1.mylab1.com"...domain server (172.17.22.103) [OK]
Success rate is 0 percent (0/2)
!
! The PING fails because the address 172.17.4.150 cannot be reached
from the inside
%ASA-6-302015: Built outbound UDP connection 1751954 for
out:172.17.22.103/53 (172.17.22.103/53) to inside:192.168.2.71/56140
(192.168.2.71/56140)
%ASA-6-410004: DNS Classification: Received DNS request (id 3) from
inside:192.168.2.71/56140 to out:172.17.22.103/53; matched Class 27:
match question
%ASA-6-410004: DNS Classification: Received DNS reply (id 3) from
out:172.17.22.103/53 to inside:192.168.2.71/56140; matched Class 27:
match question
%ASA-6-302020: Built outbound ICMP connection for faddr
172.17.4.150 /0 gaddr 192.168.2.71/34 laddr 192.168.2.71/34
!
! Without DNS Doctoring, no nat-rewrite is performed

ASA1# show service-policy interface inside inspect dns
Interface inside:
  Service-policy: INSPECTION2
    Class-map: DNS-UDP
      Inspect: dns DNS1, packet 2, drop 0, reset-drop 0
        dns-guard, count 1
        protocol-enforcement, drop 0
        nat-rewrite, count 0
        match question

```

```

log, packet 2
!
! DNS information seen on CLIENT1
CLIENT1#show host
Default domain is not set
Name/address lookup uses domain service
Name servers are 172.17.22.103
Codes: UN - unknown, EX - expired, OK - OK, ?? - revalidate
      temp - temporary, perm - permanent
      NA - Not Applicable None - Not defined
Host          Port  Flags      Age Type  Address(es)
web1.mylab1.com  None (temp, OK) 0  IP    172.17.4.150

```

Example 12-17 presents the solution for the challenge raised in Example 12-16. The `dns` keyword in the `static` command instructs ASA to translate the address in the 'A' record that is part of the DNS response. Example 12-17 shows that the host CLIENT1 receives 172.17.3.150 after name resolution of `web1.mylab1.com` and the `ping` then succeeds.

The intelligent translation of the DNS response (known as the *DNS doctoring*) is detailed in Figure 12-15. A little reflection helps to conclude that this resource becomes even more meaningful when the Web server has a private address (in the sense of RFC 1918) on the `dmz` that translates to a valid address (publicly routable) on the `out` interface.

Example 12-17 Name Resolution Using DNS Doctoring

```

! Including the 'dns' keyword on the static NAT definition
static (dmz,out) 172.17.4.150 172.17.3.150 netmask 255.255.255.255 dns
!
ASA1# show xlate debug
2 in use, 13 most used
Flags: D - DNS, d - dump, I - identity, i - dynamic, n - no random,
      r - portmap, s - static
NAT from inside:192.168.2.0 to out:192.168.2.0 flags s idle 0:32:14
timeout 0:00:00
NAT from dmz:172.17.3.150 to out:172.17.4.150 flags sD idle 0:32:14
timeout 0:00:00
!
! Successful ping from CLIENT1 (192.168.2.71) to web1.mylab1.com (172.17.3.150)

%ASA-6-302015: Built outbound UDP connection 1751997 for out:172.17.22.103/53
(172.17.22.103/53) to inside:192.168.2.71/54678
(192.168.2.71/54678)
%ASA-6-410004: DNS Classification: Received DNS request (id 6) from
inside: 192.168.2.71/54678 to out:172.17.22.103/53; matched Class 27:
match question

```

```
%ASA-6-410004: DNS Classification: Received DNS reply (id 6) from
out:172.17.22.103/53 to inside:192.168.2.71/54678; matched Class 27:
match question
%ASA-6-305011: Built dynamic ICMP translation from inside:192.168.2.71/37 to
dmz:172.17.3.251/47036
%ASA-6-302020: Built outbound ICMP connection for faddr
172.17.3.150 /0 gaddr 172.17.3.251/47036 laddr 192.168.2.71/37
!
ASA1# show service-policy interface inside inspect dns
Interface inside:
  Service-policy: INSPECTION2
  Class-map: DNS-UDP
    Inspect: dns DNS1, packet 2, drop 0, reset-drop 0
      dns-guard, count 1
      protocol-enforcement, drop 0
      nat-rewrite, count 1
!
! CLIENT1 sees the real address 172.17.3.150 as a result of DNS doctoring

CLIENT1# show host | begin Host
Host          Port  Flags      Age Type  Address(es)
web1.mylab1.com      None (temp, OK)  0  IP    172.17.3.150
```

DNS Inspection Parameters

Example 12-18 illustrates two DNS protection mechanisms provided by ASA:

- An ID mismatch between DNS response and query caused the response to be dropped by ASA. This is configured under the parameters section in a DNS policy-map. This resource contributes to mitigate DNS spoofing attacks.
- In the second case, a malformed DNS reply is dropped by ASA, because the *label length* exceeds the limit defined in the protocol specification.

Example 12-18 *Some DNS Protocol-Enforcement Examples*

```
! Specifying the log rate for the occurrence of ID mismatch
policy-map type inspect dns DNS1
parameters
  id-mismatch count 1 duration 1 action log
!
! Mismatch between DNS request ID and response ID

%ASA-6-302015: Built outbound UDP connection 1744446 for
out:172.17.22.104/53 (172.17.22.104/53) to inside:192.168.2.185/2001
(192.168.2.185/2001)
```

```
%ASA-2-410002: Dropped 1 DNS responses with mis-matched id in the
past 1 second(s): from out:172.17.22.104/53 to
inside:192.168.2.185/2001
!
! Malformed DNS packet (label length exceeds acceptable value) is dropped

%ASA-6-302015: Built outbound UDP connection 1744560 for
out:172.17.22.104/53 (172.17.22.104/53) to inside:192.168.2.180/12000
(192.168.2.180/12000)
%ASA-4-410001: Dropped UDP DNS reply from out:172.17.22.104/53 to
inside:192.168.2.180/12000; label length 172 bytes exceeds protocol
limit of 63 bytes
!
! Viewing information about DNS inspection and eventual drops

ASA1# show service-policy interface inside inspect dns
Interface inside:
  Service-policy: INSPECTION2
    Class-map: DNS-UDP
      Inspect: dns DNS1, packet 16, drop 8, reset-drop 0
        dns-guard, count 0
        protocol-enforcement, drop 4
        id-mismatch count 1 duration 1, log 0
    !
ASA1# show asp drop | include dns
DNS Inspect invalid domain label (inspect-dns-invalid-domain-label)          4
DNS Inspect id not matched (inspect-dns-id-not-matched)                    4
```

Example 12-19 illustrates a situation in which ASA is configured to randomize the DNS transaction ID, another resource useful for protection against spoofing of DNS replies. Figure 12-14, as shown previously, complements Example 12-19 by displaying the packet that corresponds to the DNS reply. It is important to notice that

- The DNS request ID shown in the Syslog message registers the post randomization ID value (21147).
- The DNS server replies to the randomized ID (21147), as characterized by the packet registered in Figure 12-14.
- The DNS Reply ID (22000) in the Syslog message corresponds to the original DNS ID selected by the client (in the DNS query).

Example 12-19 *Instructing ASA to Randomize the DNS Transaction ID*

```
policy-map type inspect dns DNS1
  parameters
```


id-randomization

```

!
%ASA-6-302015: Built outbound UDP connection 1751809 for
out:172.17.22.103/53 (172.17.22.103/53) to inside:192.168.2.180/14001
(192.168.2.180/14001)
%ASA-6-410004: DNS Classification: Received DNS request (id 21147)
from inside: 192.168.2.180/14001 to out:172.17.22.103/53; matched
Class 27: match question
%ASA-6-410004: DNS Classification: Received DNS reply (id 22000) from
out: 172.17.22.103/53 to inside:192.168.2.180/14001; matched Class
27: match question
!
ASA1# show service-policy interface inside inspect dns
Interface inside:
  Service-policy: INSPECTION2
    Class-map: DNS-UDP
      Inspect: dns DNS1, packet 12, drop 0, reset-drop 0
        dns-guard, count 6
        protocol-enforcement, drop 0
        id-randomization, count 6

```

Note Although randomization of the DNS transaction ID enhances security, it makes troubleshooting more difficult. This is a situation in which more security means more complex management and operation. An interesting intermediate option is to disable the feature if you need to troubleshoot DNS-related issues.

Example 12-20 documents a practical situation in which two DNS packets (a request and a reply) are dropped as a result of exceeding the configured length of 512 bytes.

Example 12-20 *Filtering Based on DNS Message Length*

```

policy-map type inspect dns DNS1
  parameters
    message-length maximum 512
!
%ASA-4-410001: Dropped UDP DNS request from
inside:192.168.2.180/16500 to out:172.17.22.104/53; packet length 600
bytes exceeds configured limit of 512 bytes
%ASA-4-410001: Dropped UDP DNS reply from out:172.17.22.104/53
to inside:
192.168.2.182/16502; packet length 800 bytes exceeds configured limit
of 512 bytes
!

```

```

ASA1# show service-policy interface inside inspect dns
Interface inside:
  Service-policy: INSPECTION2
    Class-map: DNS-UDP
      Inspect: dns DNS1, packet 15, drop 2, reset-drop 0
        message-length maximum 512, drop 0
        dns-guard, count 0
        protocol-enforcement, drop 0
    !
ASA1# show asp drop | include dns
DNS Inspect packet too long (inspect-dns-pak-too-long)

```

2

Some Additional DNS Inspection Capabilities

Example 12-21 shows a way to block DNS packets that carry a particular value for the DNS Type parameter (in this case, a CNAME record similar to the one shown in Figure 12-14).

Example 12-22 characterizes ASA's capability to filter based on a specific value in the DNS Flags field, which was documented in Figure 12-13. In this particular example, a response from a server that is not authoritative (AA =0) is blocked.

Example 12-21 *Filtering Based on the DNS Resource Record Type*

```

policy-map type inspect dns DNS1
  match dns-type eq CNAME
  drop-connection log
!
%ASA-4-410003: DNS Classification: Dropped DNS reply (id 22000) from
out: 172.17.22.103/53 to inside:192.168.2.180/12000; matched Class
26: match dns-type eq CNAME
%ASA-4-507003: udp flow from inside:192.168.2.180/12000 to
out:172.17.22.103/53
terminated by inspection engine, reason - inspector disconnected,
dropped packet.
!
ASA1# show service-policy interface inside inspect dns
Interface inside:
  Service-policy: INSPECTION2
    Class-map: DNS-UDP
      Inspect: dns DNS1, packet 6, drop 3, reset-drop 0
        dns-guard, count 3
        protocol-enforcement, drop 0
        match dns-type eq CNAME
        drop-connection log, packet 3
    !

```

Example 12-22 *Filtering Based on the Flags field of the DNS Header*

```

! DNS Response (QR =1) from a server that is not authoritative (AA = 0)
class-map type inspect dns match-all DNS-HEADER1
  match header-flag eq QR RD RA
!
policy-map type inspect dns DNS1
  class DNS-HEADER1
    drop-connection log
!
%ASA-6-302015: Built outbound UDP connection 1751925 for
out:172.17.22.104/53 (172.17.22.104/53) to inside:192.168.2.180/15000
(192.168.2.180/15000)
%ASA-4-410003: DNS Classification: Dropped DNS reply (id 25000) from
out:172.17.22.104/53 to inside:192.168.2.180/15000; matched Class 26:
DNS-HEADER1
%ASA-4-507003: udp flow from inside:192.168.2.180/15000 to
out:172.17.22.104/53
terminated by inspection engine, reason - inspector disconnected,
dropped packet.
!
ASA1# show service-policy interface inside inspect dns
Interface inside:
  Service-policy: INSPECTION2
    Class-map: DNS-UDP
      Inspect: dns DNS1, packet 6, drop 3, reset-drop 0
      dns-guard, count 3
      protocol-enforcement, drop 0
      class DNS-HEADER1
        drop-connection log, packet 3

```

Besides reviewing some characteristics of DNS messages, this section covered several resources that provide application visibility and filtering capabilities. Explore not only other CLI options but also the configuration using the ASDM graphical approach (refer to Figure 12-11).

FTP Inspection in ASA

This chapter already promoted a quick review of FTP modes of operation during the study of the IOS ZFW. This section focuses on ASA's inspection capabilities for this application protocol.

Example 12-23 simply characterizes that ASA has visibility of events such as file retrieval using FTP Active mode. It is interesting to observe that the Syslog messages reveal the creation of both the control and data connections, respectively on TCP ports 21 and 20.

Example 12-24 registers the task of storing a file (PUT command) using FTP Active mode through ASA. This example includes both the ASA's and the FTP client's perspectives.

Example 12-23 *Downloading a File with Active FTP*

```
! MS-DOS Client 192.168.1.70 connects to FTP Server 172.17.11.102
%ASA-6-302013: Built outbound TCP connection 416281 for
svcs:172.17.11.102/ 21 (172.17.11.102/21) to mgmt:192.168.1.70/48519
(192.168.1.70/48519)
%ASA-6-302013: Built outbound TCP connection 416283 for
svcs:172.17.11.102/ 20 (172.17.11.102/20) to mgmt:192.168.1.70/47650
(192.168.1.70/47650)
%ASA-6-303002: FTP connection from mgmt:192.168.1.70/48519 to
out:172.17.11.102/21,
user user2 Retrieved file attempts1_user5.htm
```

Example 12-24 *Uploading a File with Active FTP*

```
! Client Perspective
amoraes-lnx:/home/cisco# ftp 172.17.11.102
Connected to 172.17.11.102.
[ output suppressed ]
230 Logged on
Remote system type is UNIX.
ftp> put File1
local: File1 remote: File1
200 Port command successful
150 Opening data channel for file transfer.
226 Transfer OK
11 bytes sent in 0.00 secs (91.8 kB/s)
!
! What is seen on ASA
%ASA-6-302013: Built outbound TCP connection 417309 for
svcs:172.17.11.102/ 21 (172.17.11.102/21) to mgmt:192.168.1.70/38411
(192.168.1.70/38411)
%ASA-6-302013: Built outbound TCP connection 417310 for
svcs:172.17.11.102/ 20 (172.17.11.102/20) to
mgmt:192.168.1.70/42464(192.168.1.70/42464)
%ASA-6-303002: FTP connection from mgmt:192.168.1.70/38411 to
svcs:172.17.11.102/21,
user user1 Stored file File1
```

Example 12-25 displays a Passive Mode FTP file download through ASA. An instructive exercise in this case is to correlate the server perspective with ASA's. (Figure 12-4 is a good source for calculating the value of the Passive Port from the Response Arg).

Example 12-25 *Downloading a File with Passive FTP*

```

! Passive FTP session - server perspective
17:14:22 PM - user1 (192.168.1.70)> PASV
17:14:22 PM - user1 (192.168.1.70)> 227 Entering Passive Mode (172,17,11,102,4,141)
17:14:22 PM - user1 (192.168.1.70)> LIST
17:14:22 PM - user1 (192.168.1.70)> 150 Connection accepted
17:14:22 PM - user1 (192.168.1.70)> 226 Transfer OK
17:15:04 PM - user1 (192.168.1.70)> PASV
17:15:04 PM - user1 (192.168.1.70)> 227 Entering Passive Mode (172,17,11,102,4,142)
17:15:04 PM - user1 (192.168.1.70)> NLST *
17:15:04 PM - user1 (192.168.1.70)> 150 Connection accepted
17:15:04 PM - user1 (192.168.1.70)> 226 Transfer OK
17:15:13 PM - user1 (192.168.1.70)> PASV
17:15:13 PM - user1 (192.168.1.70)> 227 Entering Passive Mode (172,17,11,102,4,143)
17:15:13 PM - user1 (192.168.1.70)> RETR file_win32.zip
17:15:13 PM - user1 (192.168.1.70)> 150 Connection accepted
17:15:13 PM - user1 (192.168.1.70)> 226 Transfer OK
!
! ASA's perspective

%ASA-6-302013: Built outbound TCP connection 416426 for
svcs:172.17.11.102/ 21 (172.17.11.102/21) to
mgmt:192.168.1.70/47629(192.168.1.70/47629)
%ASA-6-302013: Built outbound TCP connection 416427 for
svcs:172.17.11.102/ 1165 (172.17.11.102/1165) to
mgmt:192.168.1.70/41001 (192.168.1.70/41001)
%ASA-6-302013: Built outbound TCP connection 416431 for
svcs:172.17.11.102/ 1166 (172.17.11.102/1166) to
mgmt:192.168.1.70/45475 (192.168.1.70/45475)
%ASA-6-302014: Teardown TCP connection 416431 for
svcs:172.17.11.102/1166 to mgmt:192.168.1.70/45475 duration 0:00:00
bytes 46 TCP FINs
%ASA-6-302013: Built outbound TCP connection 416432 for
svcs:172.17.11.102/ 1167 (172.17.11.102/1167) to
mgmt:192.168.1.70/58854 (192.168.1.70/58854)
%ASA-6-303002: FTP connection from mgmt:192.168.1.70/47629 to
svcs:172.17.11.102/21, user user1 Retrieved file file_win32.zip

```

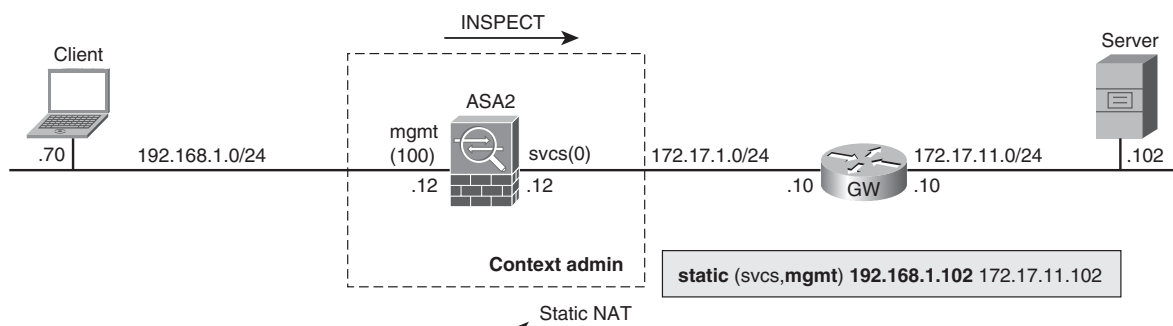
Example 12-26 relates to the scenario portrayed in Figure 12-16, for which a server-side static translation exists. Instead of accessing the server's real address, the client sees only the translated address, 192.168.1.102. ASA's knowledge of FTP ensures that the *passive address* (embedded in the FTP Response) is also translated (meaning that the NAT functionality is not limited to Layer 3).

① FTP Request (PASV command)

Internet Protocol, Src: 192.168.1.70 , **Dst: 192.168.1.102**
 Transmission Control Protocol, Src Port: 34403 , **Dst Port: 21**
 Seq: 2391549414, Ack: 3058237284, Len: 6
File Transfer Protocol (FTP)
 PASV\r\n
Request command: PASV

② FTP Request (PASV command) – post NAT

Internet Protocol, Src: 192.168.1.70 , **Dst: 172.17.11.102**
 Transmission Control Protocol, Src Port: 34403 , **Dst Port: 21**
 Seq: 2983412563, Ack: 2528907714, Len: 6
File Transfer Protocol (FTP)
 PASV\r\n
Request command: PASV



④ FTP Response (Client Perspective)

Internet Protocol, **Src: 192.168.1.102**, Dst: 192.168.1.70
 Transmission Control Protocol, **Src Port: 21**, Dst Port: 34403
 Seq: 3058237284, Ack: 2391549420, Len: 49
File Transfer Protocol (FTP)
 227 Entering Passive Mode (192,168,1,102,4,159)\r\n
 Response code: Entering Passive Mode (227)
 Response arg: Entering Passive Mode (192,168,1,102,4,159)
Passive IP address: 192.168.1.102
 Passive port: 1183

③ FTP Response (Passive Mode – OK)

Internet Protocol, **Src: 172.17.11.102**, Dst: 192.168.1.70
 Transmission Control Protocol, **Src Port: 21**, Dst Port: 34403
 Seq: 2528907714, Ack: 2983412569, Len: 49
File Transfer Protocol (FTP)
 227 Entering Passive Mode (172,17,11,102,4,159)\r\n
 Response code: Entering Passive Mode (227)
 Response arg: Entering Passive Mode (172,17,11,102,4,159)
Passive IP address: 172.17.11.102
 Passive port: 1183

Figure 12-16 FTP Passive Mode with Server-Side NAT

Example 12-26 Passive FTP with Server-Side Static NAT

```
! Static Configuration for the FTP Server with Real address 172.17.11.102
static (svcs,mgmt) 192.168.1.102 172.17.11.102 netmask 255.255.255.255
!
! FTP Passive Mode - client perspective

amoraes-lnx:/home/cisco# ftp 192.168.1.102
Connected to 192.168.1.102.
220-FileZilla Server version 0.9.34 beta
Name (192.168.1.102:root): user1
331 Password required for user1
Password: *****
230 Logged on
Remote system type is UNIX.
ftp> passive
```

```

Passive mode on.
ftp> ls
227 Entering Passive Mode (192,168,1,102,4,159)
150 Connection accepted
-rwxr-xr-x 1 ftp ftp          743469 Oct 13  2005 file_win32.exe
226 Transfer OK
ftp> mget *
mget file_win32.exe? yes
227 Entering Passive Mode (192,168,1,102,4,161)
150 Connection accepted
226 Transfer OK
743469 bytes received in 0.19 secs (3754.8 kB/s)
!
! What is seen on ASA

%ASA-6-302013: Built outbound TCP connection 416582 for
svcs:172.17.11.102 /21 (192.168.1.102 /21) to mgmt:192.168.1.70/34403
(192.168.1.70/34403)
%ASA-6-302013: Built outbound TCP connection 416583 for
svcs:172.17.11.102/ 1183 (192.168.1.102/1183) to
mgmt:192.168.1.70/42612 (192.168.1.70/42612)
%ASA-6-302014: Teardown TCP connection 416583 for
svcs:172.17.11.102/1183 to mgmt:192.168.1.70/42612 duration 0:00:00
bytes 144 TCP FINs
%ASA-6-302013: Built outbound TCP connection 416584 for
svcs:172.17.11.102/1184 (192.168.1.102/1184) to
mgmt:192.168.1.70/44773 (192.168.1.70/44773)
%ASA-6-302014: Teardown TCP connection 416584 for
svcs:172.17.11.102/1184 to mgmt:192.168.1.70/44773 duration 0:00:00
bytes 46 TCP FINs
%ASA-6-302013: Built outbound TCP connection 416585 for
svcs:172.17.11.102/1185 (192.168.1.102/1185) to
mgmt:192.168.1.70/56342 (192.168.1.70/56342)
%ASA-6-303002: FTP connection from mgmt:192.168.1.70/34403 to
svcs:172.17.11.102/21, user user1 Retrieved file file_win32.exe

```

Example 12-27 combines server-side static NAT with client-side PAT for a scenario in which an Active FTP session is built through ASA. Figure 12-17 and the following list help unveil what is going on behind the scenes:

- ASA performs PAT not only for the command channel but also for the dynamically negotiated data channel. (Please compare the Active IP Address and Active Port for stages 1 and 2 in Figure 12-17).
- Figure 12-17 registers only the first PORT command issued by the FTP client. The other PORT negotiations generated by FTP commands such as *ls* and *get* are shown in the Syslog messages.
- Each PORT negotiation requires a new dynamic TCP translation to be created.

Example 12-27 Active FTP with Client-Side PAT and Server-Side Static NAT

```

! Dynamic PAT rule for clients on segment 192.168.1.0/24 (mgmt)
nat (mgmt) 1 192.168.1.0 255.255.255.0
global (svcs) 1 172.17.1.120
!
! Command channel (connection is created on top of dynamic
translation)
%ASA-6-305011: Built dynamic TCP translation from
mgmt:192.168.1.70/39011 to svcs:172.17.1.120/18468
%ASA-6-302013: Built outbound TCP connection 416719 for
svcs:172.17.11.102/ 21 (192.168.1.102/21) to mgmt:192.168.1.70/39011
(172.17.1.120/18468)

! First PORT negotiation ('bin' command)
%ASA-6-305011: Built dynamic TCP translation from
mgmt:192.168.1.70/53441 to svcs:172.17.1.120/41093
%ASA-6-302013: Built outbound TCP connection 416720 for
svcs:172.17.11.102/ 20 (192.168.1.102/20) to mgmt:192.168.1.70/53441
(172.17.1.120/41093)
%ASA-6-302014: Teardown TCP connection 416720 for
svcs:172.17.11.102/20 to mgmt:192.168.1.70/53441 duration 0:00:00
bytes 144 TCP FINs

! Second PORT negotiation ('ls' command)
%ASA-6-305011: Built dynamic TCP translation from
mgmt:192.168.1.70/45879 to svcs:172.17.1.120/53347
%ASA-6-302013: Built outbound TCP connection 416721 for
svcs:172.17.11.102/ 20 (192.168.1.102/20) to mgmt:192.168.1.70/45879
(172.17.1.120/53347)
%ASA-6-302014: Teardown TCP connection 416721 for
svcs:172.17.11.102/20 to mgmt:192.168.1.70/45879 duration 0:00:00
bytes 46 TCP FINs

! Third PORT Negotiation ('get' command)
%ASA-6-305011: Built dynamic TCP translation from

```



```
mgmt:192.168.1.70/58632 to svcs:172.17.1.120/30952
%ASA-6-302013: Built outbound TCP connection 416722 for
svcs:172.17.11.102/ 20 (192.168.1.102/20) to
mgmt:192.168.1.70/58632(172.17.1.120/30952)
%ASA-6-303002: FTP connection from mgmt:192.168.1.70/39011 to
svcs:172.17.11.102/21,
user user1 Retrieved file file_win32.exe
```

① FTP Request (PORT command)

Internet Protocol, **Src: 192.168.1.70**, Dst: 192.168.1.102
Transmission Control Protocol, **Src Port: 39011**, Dst Port: 21
Seq: 1645816088, Ack: 3742210643, Len: 27
File Transfer Protocol (FTP)
PORT 192,168,1,70,208,193\r\n
Request command: PORT
Request arg: 192,168,1,70,208,193
Active IP address: 192.168.1.70
Active port: 53441

② FTP Request (PORT command) –post NAT

Internet Protocol, **Src: 172.17.1.120**, Dst: 172.17.11.102
Transmission Control Protocol, Src Port: 18468, Dst Port: 21
Seq: 2275911749, Ack: 2899545585, Len: 27
File Transfer Protocol (FTP)
PORT 172,17,1,120,160,133\r\n
Request command: PORT
Request arg: 172,17,1,120,160,133
Active IP address: 172.17.1.120
Active port: 41093

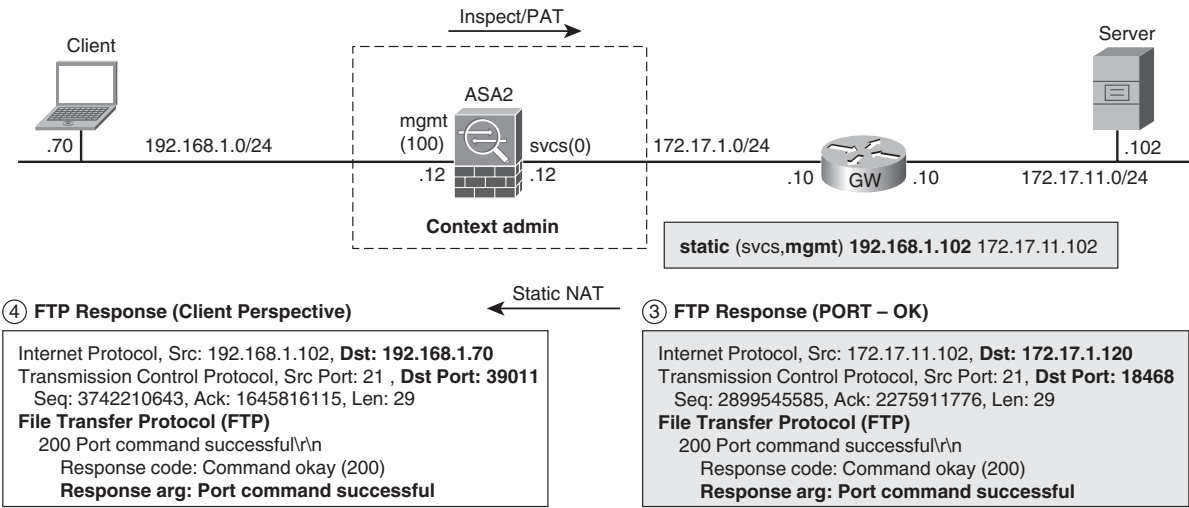


Figure 12-17 FTP Active Mode with Client-Side PAT and Server-Side Static NAT

Note After studying DNS doctoring and these examples that involve application-level address translation for FTP, you might ask why DNS needs a special handling for NAT. The conditional translation performed by DNS doctoring is used as an auxiliary feature to other protocols and not to solve any issue of NAT with DNS itself. With the aid of the DNS inspection capability, ASA enables the same application server to respond to client requests sourced either from public or private addresses.

Example 12-28 reveals what happens when FTP inspection is disabled in ASA.

- For an Active mode connection in the outbound direction, the attempt to open the data channel (inbound) is blocked. (Assuming that the default deny inbound policy is in place).
- For a Passive mode connection in the outbound direction, ASA performs PAT for the control channel (TCP port 21) but ignores the need to fix the embedded IP address in FTP. As a result, the client tries to open a data connection to the server's real address (172.17.11.102, rather than the originally called 192.168.1.102 address).

Example 12-28 *Disabling FTP Inspection*

```
! Active Mode (outbound)
%ASA-6-302013: Built outbound TCP connection 416861 for
svcs:172.17.11.102/21 (192.168.1.102/21) to mgmt:192.168.1.70/43007
(172.17.1.120/7617)
%ASA-4-106023: Deny tcp src svcs:172.17.11.102/20 dst
mgmt:192.168.1.70/ 56148 by access-group "SVCS" [0x0, 0x0]
!
! Server perspective
18:40:41 PM - user1 (172.17.1.120)> PORT 192,168,1,70,219,84
18:40:41 PM - user1 (172.17.1.120)> 200 Port command successful
18:40:41 PM - user1 (172.17.1.120)> LIST
18:40:41 PM - user1 (172.17.1.120)> 150 Opening data channel for
directory list.
18:40:52 PM - user1 (172.17.1.120)> 425 Can't open data connection.
!
! Passive Mode (outbound)

nat (mgmt) 1 192.168.1.0 255.255.255.0
global (svcs) 1 172.17.1.120
!
! Client Perspective (the real address of the server is not translated inside FTP)
amoraes-lnx:/home/cisco# ftp 192.168.1.102
Connected to 192.168.1.102.
220-FileZilla Server version 0.9.34 beta
Name (192.168.1.102:root): user1
331 Password required for user1
Password:
230 Logged on
Remote system type is UNIX.
ftp> passive
Passive mode on.
ftp> ls
227 Entering Passive Mode(172,17,11,102,4,252)
```

```

ftp: connect: Connection timed out
!
! PAT for the control channel goes on normally
%ASA-6-305011: Built dynamic TCP translation from
mgmt:192.168.1.70/41746 to svcs:172.17.1.120/22018
%ASA-6-302013: Built outbound TCP connection 417184 for
svcs:172.17.11.102/ 21 (192.168.1.102/21) to mgmt:192.168.1.70/41746
(172.17.1.120/22018)
!
! The client tries to connect to the PASV port (1276) using the real
address
%ASA-6-305011: Built dynamic TCP translation from
mgmt:192.168.1.70/39067 to svcs:172.17.1.120/4762
%ASA-3-305005: No translation group found for tcp src
mgmt:192.168.1.70/39067 dst
svcs:172.17.11.102/ 1276

```

The examples presented so far characterize ASA's knowledge of FTP behavior. The goal for the new set of examples is to take advantage of ASA's protocol awareness to materialize more sophisticated filtering resources.

Example 12-29 teaches how to mask the FTP server banner and the server reply to the SYST command. Figure 12-18 documents the effects of such a masking from the client's perspective. The less information you share about your server, the harder it is to search for potential vulnerabilities and related exploits.

Example 12-29 *Masking Server Information in the FTP Response*

```

class-map FTP1
  match port tcp eq ftp
!
policy-map type inspect ftp FTPMAP1
  parameters
    mask-banner
    mask-syst-reply
!
policy-map INSPECTION1
  class FTP1
    inspect ftp strict FTPMAP1
!
service-policy INSPECTION1 interface inside

```

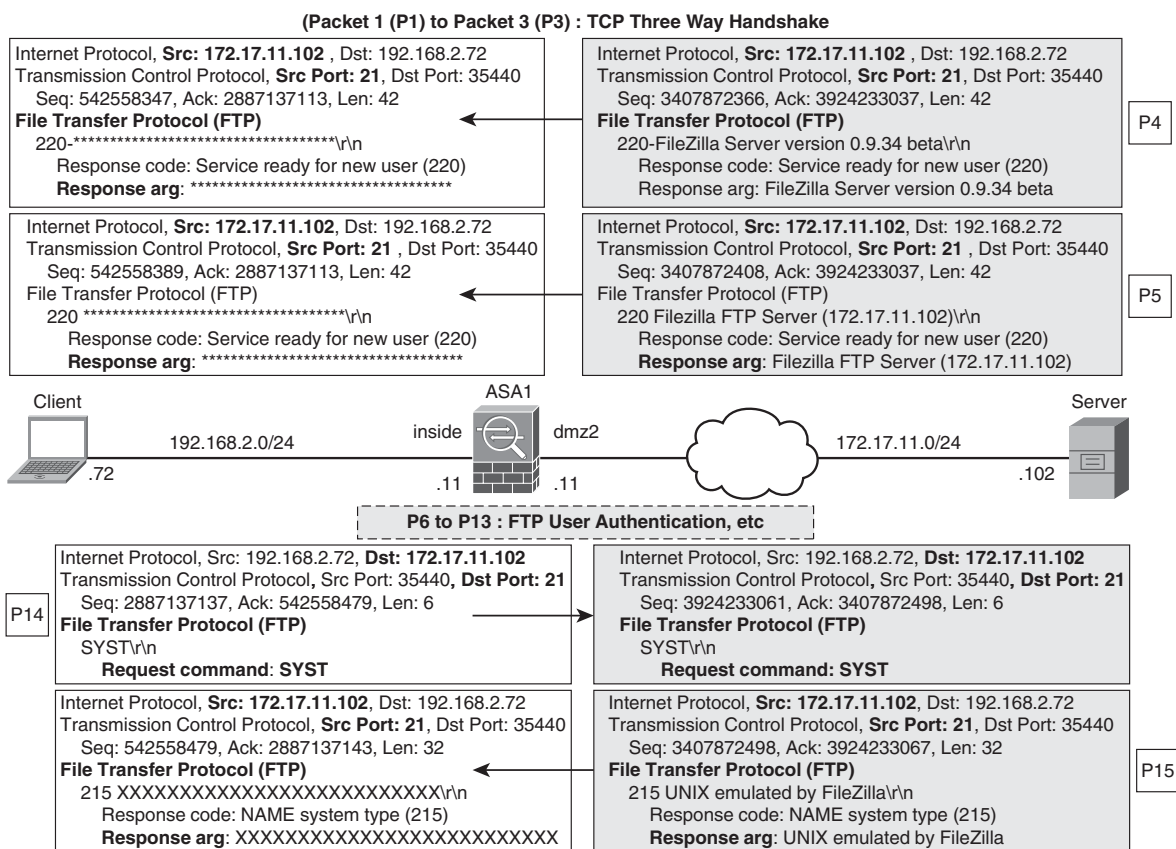


Figure 12-18 Masking Server Information Contained in the FTP Response

Example 12-30 registers the configuration to build a sample policy that blocks a certain set of FTP request commands. The example also presents the client's perspective, just after execution of a forbidden command.

Example 12-30 *Filtering Some FTP Request Commands*

```
class-map FTP1
  match port tcp eq ftp
!
class-map type inspect ftp match-any FTPCMD1
  match request-command put
  match request-command dele
  match request-command rmd
!
```

```

policy-map type inspect ftp FTPMAP1
  parameters
    class FTPCMD1
      reset log
!
policy-map INSPECTION1
  class FTP1
    inspect ftp strict FTPMAP1
!
service-policy INSPECTION1 interface mgmt
!
! Sample attempt of uploading a file - client perspective

amoraes-lnx:/home/cisco# ftp 192.168.1.102
Connected to 192.168.1.102.
[ output suppressed ]
230 Logged on
Remote system type is UNIX.
ftp> put File1
local: File1 remote: File1
200 Port command successful
421 Service not available, remote server has closed connection

%ASA-5-303005: Strict FTP inspection matched Class 26: FTPCMD1 in
policy-map FTPMAP1, Reset connection from mgmt:192.168.1.70/55990 to
svcs:172.17.11.102/21
%ASA-4-507003: tcp flow from mgmt:192.168.1.70/55990 to
svcs:172.17.11.102/21
terminated by inspection engine, reason - inspector reset
unconditionally.

ASA2/admin# show service-policy interface mgmt inspect ftp
Interface mgmt:
  Service-policy: INSPECTION1
    Class-map: FTP1
      Inspect: ftp strict FTPMAP1, packet 50, drop 0, reset-drop 2
        class FTPCMD1 (match-any)
          Match: request-command put , 1 packets
          Match: request-command dele , 0 packets
          Match: request-command rmd , 1 packets
          reset log, packet 2

```

Example 12-31 displays a sample policy designed to prevent FTP operations that involve a certain file type. Figure 12-19 relates to this example and presents the client and server perspectives for a series of FTP command execution attempts through ASA.

Example 12-31 *Filtering Based on the FTP File Type*

```

regex FILETYPE1 ".*\[Ee\][Xx][Ee]"
!
class-map FTP1
  match port tcp eq ftp
!
class-map type inspect ftp match-all FTP-FILE1
  match filetype regex FILETYPE1
!
policy-map type inspect ftp FTPMAP1
  parameters
    class FTP-FILE1
      reset log
!
policy-map INSPECTION1
  class FTP1
    inspect ftp strict FTPMAP1
!
service-policy INSPECTION1 interface inside
!
! ASA in action

%ASA-6-302013: Built outbound TCP connection 1756951 for
dmz2:172.17.11.102/21 (172.17.11.102/21) to inside:192.168.2.72/40330
(192.168.2.72/40330)
[ output suppressed ]
%ASA-6-303002: FTP connection from inside:192.168.2.72/40330 to
dmz2:172.17.11.102/21, user user1 Retrieved file exe1.doc
%ASA-6-303002: FTP connection from inside:192.168.2.72/40330 to
dmz2:172.17.11.102/21, user user1 Retrieved file Reg1.key
%ASA-5-303005: Strict FTP inspection matched Class 22: FTP-FILE1 in
policy-map FTPMAP1, Reset connection from inside:192.168.2.72/40330
to dmz2:172.17.11.102/21
%ASA-4-507003: tcp flow from inside:192.168.2.72/40330 to
dmz2:172.17.11.102/21 terminated by inspection engine, reason -
inspector reset unconditionally.
%ASA-6-302014: Teardown TCP connection 1756951
for dmz2:172.17.11.102/21 to inside:192.168.2.72/40330 duration
0:00:43 bytes 720 Flow closed by inspection

```

Client Perspective

```

amoraes-lnx:~# ftp 172.17.11.102
Connected to 172.17.11.102.
[ output suppressed ...]
ftp> ls
200 Port command successful
150 Opening data channel for directory list.
-rw-r--r-- 1 ftp ftp      0 Jul 18 17:55 exe1.doc
-rwxr-xr-x 1 ftp ftp      477 Jan 29 2008 Reg1.exe
-rw-r--r-- 1 ftp ftp      477 Jan 29 2008 Reg1.key
drwxr-xr-x 1 ftp ftp      0 Jul 18 17:55 Registry1
-rwxr-xr-x 1 ftp ftp     1206366 Jan 29 2008 war.exe
226 Transfer OK
ftp> get exe1.doc
local: exe1.doc remote: exe1.doc
200 Port command successful
150 Opening data channel for file transfer.
226 Transfer OK
ftp> get Reg1.key
local: Reg1.key remote: Reg1.key
200 Port command successful
150 Opening data channel for file transfer.
226 Transfer OK
477 bytes received in 0.00 secs (137.2 kB/s)
ftp> get Reg1.exe
local: Reg1.exe remote: Reg1.exe
200 Port command successful
421 Service not available, remote server has closed connection
ftp>

```

Server Perspective

```

(not logged in) (192.168.2.72)> Connected, sending welcome message...
(not logged in) (19.168.2.72)> 220-FileZilla Server version 0.9.34 beta
(not logged in) (192.168.2.72)> 220 Filezilla FTP Server (172.17.11.102)

```

[output suppressed ...]

```

user1 (192.168.2.72)> PORT 192,168,2,72,137,97
user1 (192.168.2.72)> 200 Port command successful
user1 (192.168.2.72)> LIST
user1 (192.168.2.72)> 150 Opening data channel for directory list.
user1 (192.168.2.72)> 226 Transfer OK
user1 (192.168.2.72)> TYPE I
user1 (192.168.2.72)> 200 Type set to I
user1 (192.168.2.72)> PORT 192,168,2,72,172,226
user1 (192.168.2.72)> 200 Port command successful
user1 (192.168.2.72)> RETR exe1.doc
user1 (192.168.2.72)> 150 Opening data channel for file transfer.
user1 (192.168.2.72)> 226 Transfer OK
user1 (192.168.2.72)> PORT 192,168,2,72,223,46
user1 (192.168.2.72)> 200 Port command successful
user1 (192.168.2.72)> RETR Reg1.key
user1 (192.168.2.72)> 150 Opening data channel for file transfer.
user1 (192.168.2.72)> 226 Transfer OK
user1 (192.168.2.72)> PORT 192,168,2,72,203,248
user1 (192.168.2.72)> 200 Port command successful
user1 (192.168.2.72)> disconnected.

```

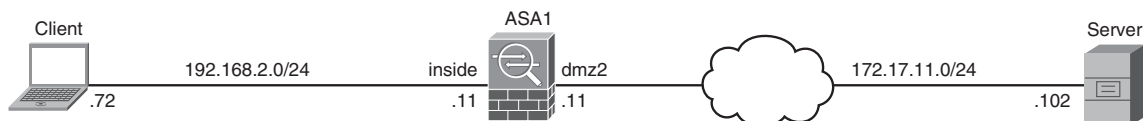


Figure 12-19 Sample FTP Session to Which File Type Filtering Is Applied

Tip It is always interesting to use the **test regex** command before applying a Regular Expression to a **class-map**. For instance, the two following tests illustrate a situation in which the regex “***\.[Ee][Xx][Ee]**” fails to match a sample string and another in which it succeeds.

```

ASA1# test regex exe.doc ".*\.[Ee][Xx][Ee]"
INFO: Regular expression match failed.
ASA1# test regex doc1.exe ".*\.[Ee][Xx][Ee]"
INFO: Regular expression match succeeded.

```

HTTP Inspection in ASA

HTTP inspection is not enabled by default in ASA, which was earlier shown in Example 12-13. The first bonus from implementing this feature is that ASA starts logging the URLs accessed, even in scenarios that involve NAT. This is shown in Example 12-32.

Example 12-32 Basic HTTP Inspection

```
! Enabling HTTP inspection at the global level (default global_policy)
ASA1(config)# policy-map global_policy
ASA1(config-pmap)# class inspection_default
ASA1(config-pmap-c)# inspect http
!
ASA1# show running-config policy-map global_policy | include http
inspect http
!
%ASA-6-302013: Built outbound TCP connection 503 for
dmz:172.16.222.22/80 (172.16.222.22/80) to mgmt:192.168.1.250/4610
(192.168.1.250/4610)
%ASA-5-304001: 192.168.1.250 Accessed URL 172.16.222.22:/
!
ASA1# show service-policy global inspect http
Global policy:
  Service-policy: global_policy
  Class-map: inspection_default
    Inspect: http, packet 22, drop 0, reset-drop 0
!
! Inspection and NAT

static (dmz,mgmt) 192.168.1.22 172.16.222.22 netmask 255.255.255.255
access-list MGMT extended permit tcp 192.168.1.0 255.255.255.0 host
192.168.1.22 eq www
access-group MGMT in interface mgmt
!
! Client accesses the server on 192.168.1.22 (whose real address is
172.16.222.22)
%ASA-6-302013: Built outbound TCP connection 33 for
dmz:172.16.222.22/80 (192.168.1.22/80) to mgmt:192.168.1.15/2331
(192.168.1.15/2331)
%ASA-5-304001: 192.168.1.15 Accessed URL 172.16.222.22:/
```


Note The `packet-tracer` utility shows information about the order of operations when ACLs, NAT, and Application Inspection are configured. You are encouraged to perform, for instance, the following simulation:

```
■ packet-tracer input mgmt tcp 192.168.1.15 2000 192.168.1.22 80
```

Example 12-33 proposes a Deep Packet Inspection policy that is capable of blocking the `Post` request method. Figure 12-20 enhances this example, by presenting not only the correspondent topology but also the whole MPF structure.

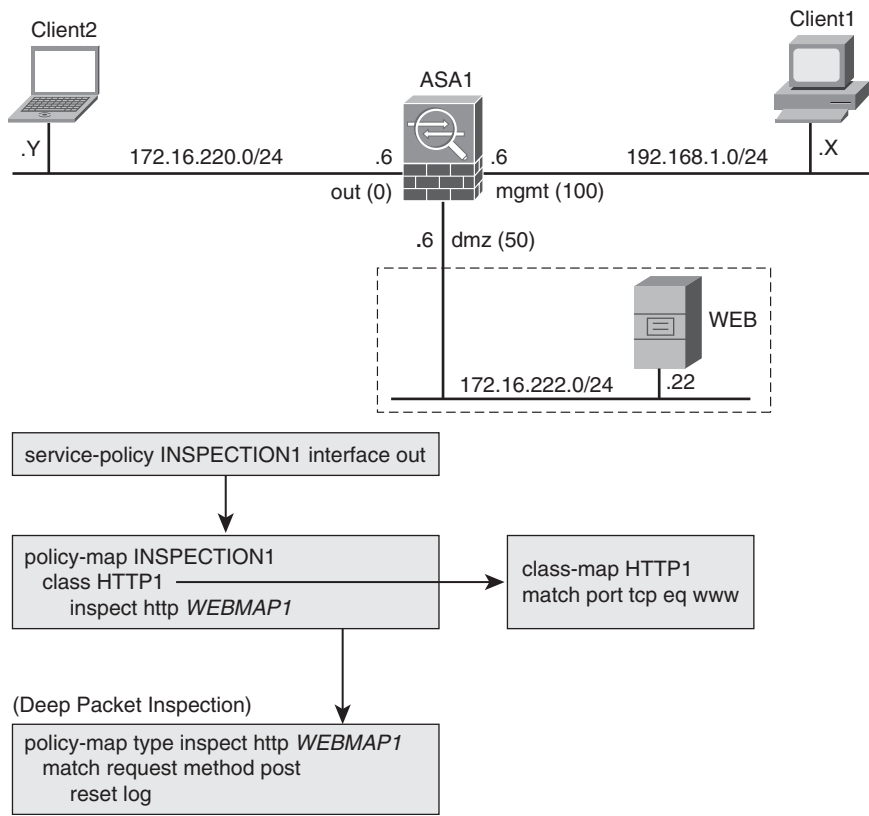


Figure 12-20 Reference Topology for HTTP Inspection and MPF Structure

Example 12-33 Matching on the HTTP Request Method

```
class-map HTTP1
  match port tcp eq www
!
policy-map type inspect http WEBMAP1
```

```

match request method post
  reset log
!
policy-map INSPECTION1
  class HTTP1
    inspect http WEBMAP1
!
service-policy INSPECTION1 interface out
!
! Inspection Policy in action
%ASA-6-302013: Built inbound TCP connection 1712795 for
out:172.16.220.100/10000 (172.16.220.100/10000) to
dmz:172.16.222.22/80 (172.16.222.22/80)
%ASA-5-415009: HTTP - matched request method post in policy-map
WEBMAP1, method
matched - Resetting connection from out:172.16.220.100/10000 to
dmz:172.16.222.22/80
%ASA-4-507003: tcp flow from out:172.16.220.100/10000 to dmz:172.16.222.22/80
terminated by inspection engine, reason - reset unconditionally.
%ASA-6-302014: Teardown TCP connection 1712795 for out:172.16.220.100/10000 to
dmz:172.16.222.22/80 duration 0:00:00 bytes 0 Flow closed by inspection
!
! Viewing information about the inspection policy
ASA1# show service-policy interface out inspect http
Interface out:
  Service-policy: INSPECTION1
    Class-map: HTTP1
      Inspect: http WEBMAP1, packet 4, drop 1, reset-drop 1
        protocol violations
          packet 0
          match request method post
            reset log, packet 1

```

Example 12-34 illustrates a policy that can reset and log HTTP connections for which the *set-cookie* header is set in the server response. The associated topology and the MPF structure are essentially identical to those represented in Figure 12-20, with the distinction that the match criterion now refers to a *response header*, instead of a request header.

Example 12-35 also builds on Figure 12-20 but centers on enforcing coherence between HTTP requests and responses, in what concerns the presence of the *Content-type* header. The associated Syslog message, 415014, provides detailed information about the events that triggered the ASA actions.

Example 12-34 *Matching the set-cookie Response Header*

```

class-map HTTP1
  match port tcp eq www
!
policy-map type inspect http WEBMAP1
  match response header set-cookie length gt 1
  reset log
!
policy-map INSPECTION1
  class HTTP1
    inspect http WEBMAP1
!
service-policy INSPECTION1 interface out
!
! Inspection policy in action

%ASA-6-415002: HTTP - matched response header set-cookie length gt 1
in policy-map WEBMAP1, header field length exceeded - Resetting
connection from out:172.16.220.100/5000 to dmz:172.16.222.22/80

```

Example 12-35 *content-type Mismatch Between HTTP Request and Response Headers*

```

class-map HTTP1
  match port tcp eq www
!
! Enforcing coherence between content-types within the request and response headers
policy-map type inspect http WEBMAP1
  match req-resp content-type mismatch
  reset log
!
policy-map INSPECTION1
  class HTTP1
    inspect http WEBMAP1
!
service-policy INSPECTION1 interface out
!
! Viewing the service-policy statements matched by a given flow

ASA1# show service-policy flow tcp host 172.16.222.100 eq 2000 host 172.16.222.22
eq 80
Global policy:
  Service-policy: global_policy
  Class-map: class-default
  Match: any

```

```

    Action:
        Output flow:
Interface out:
    Service-policy: INSPECTION1
    Class-map: HTTP1
        Match: port tcp eq www
        Action:
            Input flow: inspect http WEBMAP1
        Class-map: class-default
            Match: any
            Action:
!
! Inspection policy in action

%ASA-6-302013: Built inbound TCP connection 330183 for
out:172.16.220.100/5000 (172.16.220.100/5000) to dmz:172.16.222.22/80
(172.16.222.22/80)
%ASA-5-304001: 172.16.220.100 Accessed URL 172.16.222.22:80/
%ASA-5-415014: HTTP - matched req-resp content-type mismatch in policy-map WEBMAP1,
Mime-type in response wasn't found in the accept-types of the request -
Resetting connection from out:172.16.220.100/5000 to dmz:172.16.222.22/80

```

Example 12-36 displays a sample policy for matching on the *User-agent* request header in HTTP connections. One noticeable aspect is that ASA enables the use of a **match not** statement. In this particular example, this construction means that, if the string defined by the regex BROWSER1 is not found in the User-agent header, the underlying connection must be dropped and logged. This approach to deny one specific situation and enable the rest might be convenient because of its potential to save some lines under a **policy-map** in many practical scenarios.

Example 12-36 also registers the type of information that may be revealed by the **debug appfw** event command.

Example 12-36 *User-Agent Matching*

```

! Defining a Regular Expression to match a given pattern
regex BROWSER1 ".*[Mm][Oo][Zz][Ii][Ll][Ll][Aa]*"
!
class-map HTTP1
    match port tcp eq www
!
! Denying connections whose 'User-agent' values do not comply with
regex 'BROWSER1'
policy-map type inspect http WEBMAP1
    match not request header user-agent regex BROWSER1

```

```

    drop-connection log
!
policy-map INSPECTION1
  class HTTP1
    inspect http WEBMAP1
!
service-policy INSPECTION1 interface mgmt
!
! HTTP request started by an Internet Explorer browser is permitted

D7E9B2E0:request:before deobfuscation httpState=sHTTP_START
saveState=sHTTP_START matchState=0
  data|GET / HTTP/1.1..Accept: image/gif, image/jpeg, image/pjpeg,
image/pjpeg,
application/x-shockwave-flash, /*..Accept-Language: en-us..User-
Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1;
Trident/4.0)..Accept-Encoding: gzip, deflate..Host: 172.16.222.22..Connection:
Keep-Alive....| Len:288 Req#:0

! HTTP request started by a Firefox browser is allowed through

D7E9B2E0:request:before deobfuscation httpState=sHTTP_START
saveState=sHTTP_START matchState=0
  data|GET / HTTP/1.1..Host: 172.16.222.22..User-Agent: Mozilla/5.0
(Windows; U; Windows
NT 5.1; en-US; rv:1.9.0.10) Gecko/2009042316 Firefox/3.0.10..Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8..Accept-
Language: en-us,en;q=0.5..Accept-Encoding: gzip,deflate..Accept-
Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7..Keep-Alive: 300..Connection:
keep-alive....| Len:370 Req#:0

! HTTP request started by an Opera browser is dropped

D7E9B2E0:request:before deobfuscation httpState=sHTTP_START
saveState=sHTTP_START matchState=0
  data|GET / HTTP/1.1..User-Agent: Opera/9.80 (Windows NT 5.1; U; en)
Presto/2.5.24 Version/10.53..Host: 172.16.222.22..Accept: text/html,
application/xml;q=0.9,
application/xhtml+xml, image/png, image/jpeg, image/gif, image/x-
bitmap, /*;q=0.1..Accept-Language: en-US,en;q=0.9..Accept-Charset:
iso-8859-1, utf-8, utf-16, *;q=0.1..Accept-Encoding: deflate, gzip,
x-gzip, identity, *;q=0..Connection: Keep-Alive....| Len:407 Req#:0
[ output suppressed ]
D7E9B2E0:request:>>>Silently drop & disconnect
inspectHttp: ===== Closing Connection D7E9B2E0 =====!

```

```

!
! Syslog message revealing connection drop performed by HTTP Inspection
%ASA-5-415008: HTTP - matched not request header user-agent regex BROWSER1
in policy-map WEBMAP1, header matched - Dropping connection from
mgmt:192.168.1.250/1333
to out: 172.16.222.22/80

```

Note The topology in Figure 12-20 also relates to Example 12-36. In this case, though, the **service-policy** is applied to interface *mgmt*, rather than to interface *out*.

Example 12-37 starts by characterizing that ASA's HTTP inspection is aware of Java URLs accessed. Drawing on that knowledge, a policy is built to block Java content that moves between a source/destination pair of IP addresses on specific TCP ports (range 2002-2003 in this case). The Java content removal is accomplished using the **mask** action under the application-specific **policy-map** WEBMAP1.

Figure 12-21 shows not only the topology but also the schematic MPF policy bound to Example 12-37. In this scenario, ASA is looking for signs of Java applets in the range TCP 2002-2003 and removes the Java code eventually found.

Example 12-37 Removing Java applets from the HTTP Response Body

```

! HTTP inspection provides visibility about the JAVA URLs accessed
%ASA-5-304001: 192.168.1.138 Accessed URL 172.17.11.102:/
[ output suppressed ]
%ASA-5-304001: 192.168.1.138 Accessed JAVA URL 172.17.11.102:/login_applet.class
%ASA-5-304001: 192.168.1.138 Accessed JAVA URL 172.17.11.102:/MD5.class
%ASA-5-304001: 192.168.1.138 Accessed JAVA URL 172.17.11.102:/MessageDigest.class
%ASA-5-304001: 192.168.1.138 Accessed JAVA URL 172.17.11.102:/login_applet.class
%ASA-5-304001: 192.168.1.138 Accessed JAVA URL 172.17.11.102:/tabComponent.class
!
! Building an inspection policy that removes Java content

access-list JAVA1 extended permit tcp 192.168.1.0 255.255.255.0
172.17.11.0 255.255.255.0 range 2002 2003
!
class-map ACS1
 match access-list JAVA1
!
policy-map type inspect http WEBMAP1
 match response body java-applet
 mask
!

```

```

policy-map INSPECTION1
  class ACS1
    inspect http WEBMAP1
  !
service-policy INSPECTION1 interface mgmt
!
! Inspection policy in action

%ASA-6-302013: Built outbound TCP connection 382595
for svcs:172.17.11.102/2002 (172.17.11.102/2002) to
mgmt:192.168.1.138/3857 (192.168.1.138/3857)
%ASA-5-304001: 192.168.1.138 Accessed URL 172.17.11.102:/
%ASA-5-500002: Java content modified: src 192.168.1.138 dest
172.17.11.102 on interface svcs

ASA2/admin# show service-policy interface mgmt inspect http
Interface mgmt:
  Service-policy: INSPECTION1
  Class-map: ACS1
    Inspect: http WEBMAP1, packet 50, drop 0, reset-drop 0
             tcp-proxy: bytes in buffer 0, bytes dropped 0
             protocol violations
             packet 0
             match response body java-applet
             mask, packet 1

```

Note The method just presented can be used in a similar way for Microsoft Activex filtering. You just need to replace the **java-applet** keyword with the **active-x** keyword.

Example 12-38 registers the original way of filtering java applets in ASA. Although this might seem simpler, the method shown in Example 12-37 is the recommended one for the reasons specified in the following:

- Removing java-applets from the HTTP response-body might be just one of several possible actions defined in an application-specific policy-map.
- As shown in Example 12-37, the **service-policy** approach quantifies the occurrences of a given match, for instance with the **show service-policy inspect http** command.
- The **filter java** command was designed for outbound connections, meaning that the HTTP response body is analyzed only when the return packet arrives on an interface that is on a lower sec-lvl than the interface where the request was originated. The type of access (outbound or inbound) is not an issue when using a **service-policy**.

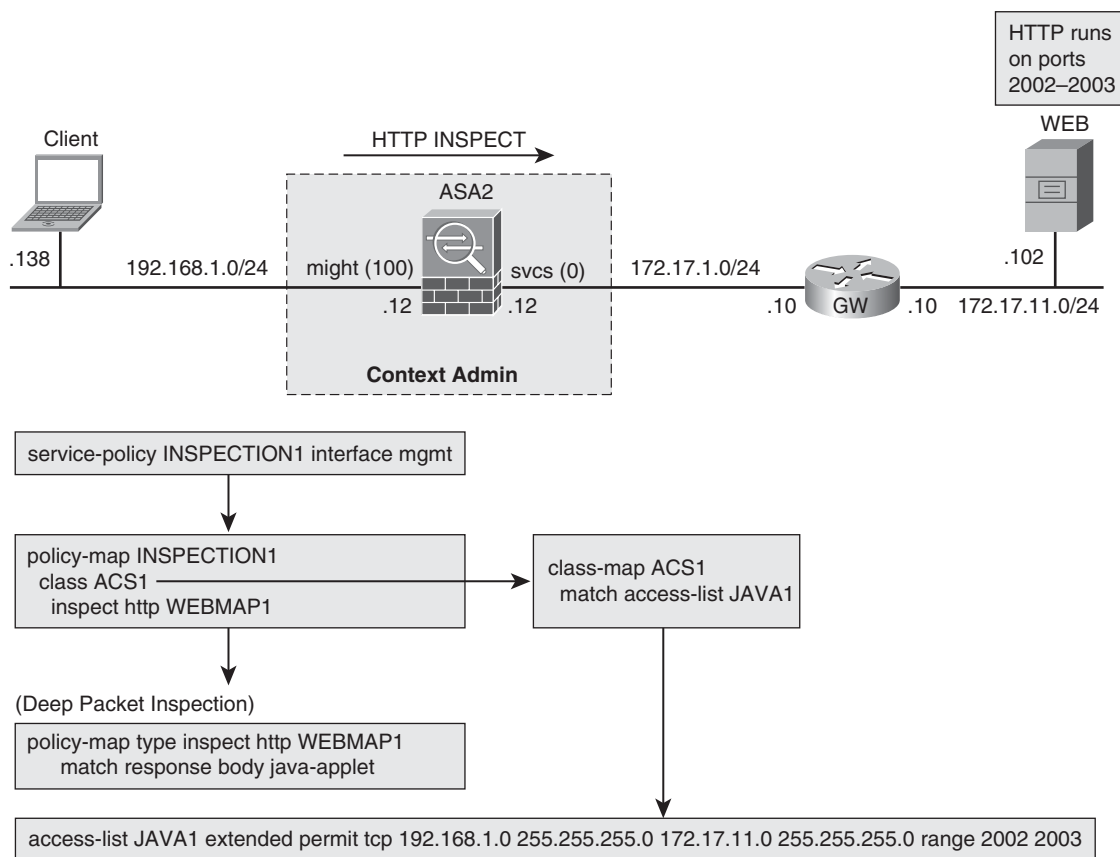


Figure 12-21 Reference Topology for Java Filtering in ASA

Example 12-38 Removing Java Applets with the filter java Command

```
! The 'filter java' command removes java applets
filter java 2002-2003 192.168.1.0 255.255.255.0 172.17.11.102
255.255.255.255
!
%ASA-6-302013: Built outbound TCP connection 98 for
svcs:172.17.11.102/2002 (172.17.11.102/2002) to
mgmt:192.168.1.138/1343 (192.168.1.138/1343)
%ASA-5-500002: Java content modified: src 192.168.1.138 dest
172.17.11.102 on interface svcs
```


Note The original method of removing Microsoft ActiveX dynamic content uses the **filteractivex** command in an analogous fashion to the **filter java** option.

In the two previous examples, the Java content removal takes place on the interface 'svcs' and not on the interface over which the client request arrives ('mgmt'). This happens because the filtering action is focused on the HTTP response body from the Web server, which is reachable via the 'svcs' interface.

Note Figures 12-17 and 12-21 show an ASA security context instead of a dedicated appliance. The purpose of such a choice is to emphasize that application inspection is equally supported when the firewall operates in Multiple Context mode.

Inspection of IM and Tunneling Traffic in ASA

As previously discussed for IOS ZFW, the Instant Messaging (IM) applications have a highly mutable network behavior. Whenever their officially assigned L4 ports are blocked, they promptly search for other available ports (mainly TCP/80), following a path that is typical of tunneling protocols.

This section quickly presents ASA resources for dealing with the *MSN messenger*, which is somewhat representative of how IM and tunneling families of protocols behave. Figure 12-22 depicts the reference topology for this study, which was built using Transparent mode. The goal was to emphasize that the same inspection features are available when ASA is inserted in the network environment as a L2 bridge.

Example 12-39 relates to Figure 12-22 and registers some typical IM operations as seen from ASA's standpoint. This visibility that goes beyond the Syslog messages is provided by the **debug im** and **debug appfw event** commands.

Example 12-40 shows the available match criteria for use inside an IM inspect map. For the scenario of Figure 12-22, the **match protocol im-msn** was employed. One of the most interesting option resides in the usage of the **match service** statement. For instance, it is possible to build an IM policy that enables only the **chat** and **conference** services, while blocking more dangerous activities such as file transfer.

If you want to simply deny the flow of IM protocols through your ASA-based firewall, another approach is to block the pre-assigned IM ports and look for traits of IM activity inside HTTP connections.

ASA default configuration includes a set of protocol signatures (mainly related to IM and tunneling applications) that can be displayed using either the **show running-config all regex** command or following the ASDM path **Configuration > Firewall > Objects > Regular Expressions** (Figure 12-12). The preconfigured HTTP class-maps that employ these Regular Expressions are visible using the **show running-config all class-map** or

navigating to ASDM menu **Configuration > Firewall > Objects > Class Maps > HTTP** (Figure 12-11).

Specific actions can then be configured for each of these HTTP class-maps inside HTTP inspect maps.

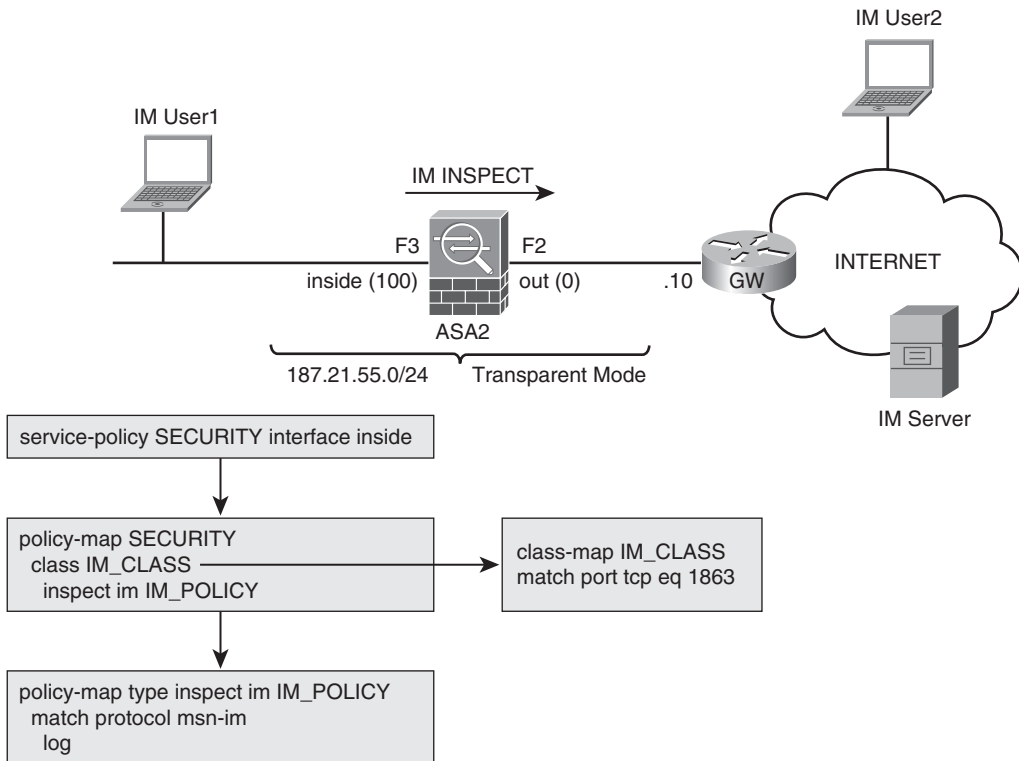


Figure 12-22 Reference Topology for IM Inspection in ASA

Example 12-39 ASA's Visibility of Typical IM Operations

```

! Connecting to the MSN IM Server
%ASA-6-302013: Built outbound TCP connection 9007 for
out:207.46.124.215/1863 (207.46.124.215/1863) to
inside:187.21.55.62/63755 (187.21.55.62/63755)
AIC MSNMSGR: insp_msnmsgr_create_sis_ext()
AIC IM: insp_im_set_result_by_action() result = 3
%ASA-6-726001: Inspected MSN Messenger Session between Client
fredvas2000@hotmail.com and ? Packet flow from
inside:/187.21.55.62/63755 to out:/207.46.124.215/1863 Action:
Received Matched Class 23: MSN
! User 'fredvas2000' sends participants list to the server
  
```

```

AIC MSNMSGGR: CAL peer UserHandle=fredvas2000@hotmail.com len=23 num_of_peers=1
AIC IM: insp_im_set_result_by_action() result = 3
AIC MSNMSGGR: CAL peer UserHandle=jrpvasconcelos@hotmail.com len=26 num_of_peers=2
AIC IM: insp_im_set_result_by_action() result = 3
!
! The remote user enters the chat session (JOI message)

AIC MSNMSGGR: JOI peer UserHandle=jrpvasconcelos@hotmail.com len=23 num_of_peers=2
AIC IM: insp_im_set_result_by_action() result = 3
AIC MSNMSGGR: JOI peer UserHandle=jrpvasconcelos@hotmail.com;{204b3dca-807e-4d7f-af08-b18235c8f920} len=65 num_of_peers=3
AIC IM: insp_im_set_result_by_action() result = 3

! MSN sends a Ring message (RNG) to the remote user

AIC MSNMSGGR: RNG svr ip = 207.46.124.121 port = 1863
AIC IM: insp_im_set_result_by_action() result = 3
AIC MSNMSGGR: insp_msnmgr_create_sis_ext()

```

Example 12-40 Main IM Policy Options in ASA

```

! Match criteria under an IM inspect map
ASA1(config)# policy-map type inspect im IM1
ASA1(config-pmap)# match ?
mpf-policy-map mode commands/options:
  filename          Match filename from IM file transfer service
  ip-address         Match client IP address for IM application or service
  login-name        Match client login-name from IM service
  not                Negate this match result
  peer-ip-address    Match peer (client or server) IP address for IM application or
service
  peer-login-name    Match client peer login name from IM service
  protocol           Match an Instant Messenger Protocol
  service            Match an Instant Messenger Service
  version           Match version from IM file transfer service
!
! Specific services controllable inside an IM policy
ASA1(config-pmap)# match service ?
mpf-policy-map mode commands/options:
  chat              Text Chat
  conference        Conference or Chat Room
  file-transfer     File Transfer
  games             Games
  voice-chat        Voice Chat
  webcam            Web Camera or Video

```

Botnet Traffic Filtering in ASA

The term *botnet* is frequently used to define a collection of computers that run tasks automatically, without intent or consent of their owners. The machines that fall victim of the botnet are remotely controlled and typically used for generating Denial of Service (DoS) attacks. Although DoS is not a new concept, botnets render this class of attack harder to mitigate with traditional firewall features because the infected hosts may be anywhere in the Internet. Clearly, there is need for a dynamic filtering mechanism that can adapt to the changes in the addresses of the *bot* (or *zombie*) computers.

The ASA Botnet Traffic Filtering (BTF) mechanism is an advanced protection feature that leverages DNS inspection, and uses the information obtained from a dynamic database of known bad domain names and IP addresses, to monitor and optionally block connection attempts to malware hosting sites.

This dynamic malware database can be supplemented with statically configured addresses and domain names. Further, if you have extra information that gives you the confidence that a certain address from the dynamic database should not be considered malicious, you can manually move it to a static *whitelist*.

The main tasks associated with the operation of the BTF functionality are represented in Figure 12-23 and described here:

1. The ASA appliance downloads the dynamic malware database and periodically checks for updates.
2. The internal client sends a DNS query to resolve the domain name of the destination host (*mightrymalware.com* in the example).
3. ASA, which is configured for DNS snooping, compares the domain name from the DNS response with its database and, in the event of a match, adds this name and the corresponding IP address to the DNS reverse lookup cache.
4. When the internal client (potentially an infected host) tries to connect to the address of the malware site, the ASA sends a syslog message informing about the suspicious activity and optionally drops the traffic (if this action has been configured).

Example 12-41 registers the **dynamic-filter** commands that govern the use of the dynamic malware database. This example also displays important information about the database contents and the updater client.

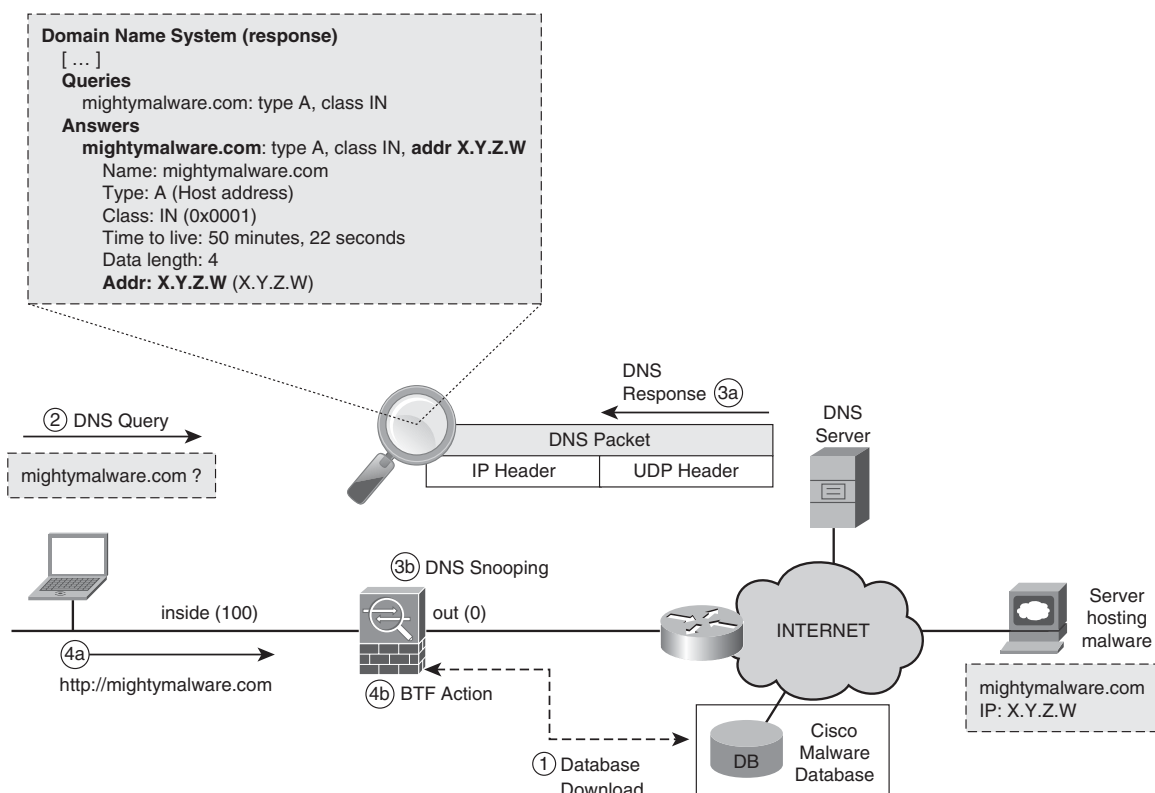


Figure 12-23 Overview of the BotnetTraffic Filtering Feature

Example 12-41 Handling the Malware Database for BTF

```
! Instructing ASA to use the dynamic malware database and enabling updates
dynamic-filter updater-client enable
dynamic-filter use-database
!
! Information about the database updater client

ASA# show dynamic-filter updater-client
Dynamic Filter updater client is enabled
Updater server URL is https://update-manifests.ironport.com
Application name: threatcast, version: 1.0
Encrypted UDI:
0bb93985f42d941e50dc8f022350d1a8ba0ae4629d9db433cd7d3dec465ce33147a06ba4bd25f5a1fa3
25c16f0baf89a
Last update attempted at 02:03:11 UTC Mar 8 2011,
  with result: Downloaded file successfully
Next update is in 00:22:06
```

```
Database file version is '1299543181' fetched at 02:03:11 UTC Mar 8 2011, size:
2097130
```

```
!
```

```
! Information about the content of the malware database
```

```
ASA# show dynamic-filter data
```

```
Dynamic Filter is using downloaded database version '1299543181'
```

```
Fetches at 02:03:11 UTC Mar 8 2011, size: 2097130
```

```
Sample contents from downloaded database:
```

```
warezriley.net livefuss.com lefty.org rm-rf.will.hackyou.info
emomqxfusoqwkol.info xhaito.com seobeat.co.cc buildtraffic.com
```

```
Sample meta data from downloaded database:
```

```
threat-level: very-high, category: Malware,
```

```
description: "These are sources that use various exploits to
deliver adware, spyware and other malware to victim computers. Some
of these are associated with rogue online vendors and distributors of
dialers which deceptively call premium-rate phone numbers."
```

```
threat-level: high, category: Bot and Threat Networks,
```

```
description: "These are rogue systems that control infected
computers. They are either systems hosted on threat networks or
systems that are part of the botnet itself."
```

```
threat-level: moderate, category: Malware,
```

```
description: "These are sources that deliver deceptive or malicious
anti-spyware, anti-malware, registry cleaning, and system cleaning
software."
```

```
threat-level: low, category: Ads,
```

```
description: "These are advertising networks that deliver banner
ads, interstitials, rich media ads, pop-ups, and pop-unders for
websites, spyware and adware. Some of these networks send ad-
oriented HTML emails and email verification services."
```

```
Total entries in Dynamic Filter database:
```

```
Dynamic data: 82226 domain names , 3277 IPv4 addresses
```

```
Local data: 0 domain names , 0 IPv4 addresses
```

```
Active rules in Dynamic Filter asp table:
```

```
Dynamic data: 0 domain names , 3277 IPv4 addresses
```

```
Local data: 0 domain names , 0 IPv4 addresses
```

```
!
```

```
! Searching for specific strings in the dynamic database
```

```
ASA# dynamic-filter database find worm
```

```
scentworm.ru m=44098
```

```
dewworm.ru m=44098
```

```
Found more than 2 matches, enter a more specific string to find an
exact match
```

Note As shown in Example 12-41, the dynamic database also contains IP addresses (and not only domain names). When a host tries to access an IP address that is already in the database, the BTF functionality can monitor and drop traffic without the need of inspecting DNS responses.

Figure 12-24 portrays the reference topology and the corresponding policy structure in an environment that has the BTF feature enabled. Examples 12-42 to 12-44 all relate to this figure, each of them presenting some relevant information about BTF operation.

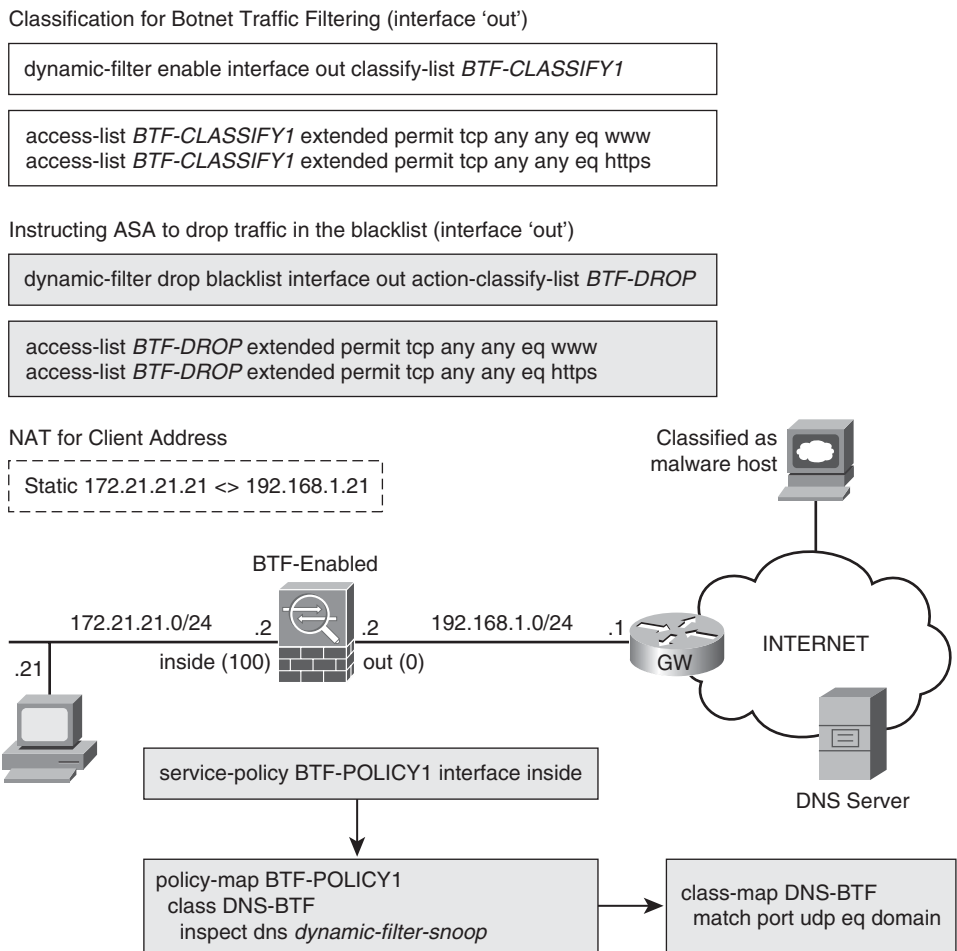


Figure 12-24 Reference Topology for BTF Analysis

The **dynamic-filter enable** command in Example 12-42 establishes, with the aid of the ACL named BTF-CLASSIFY1, that only HTTP and HTTPS requests should be monitored with the BTF resource. Following traffic classification, the **dynamic-filter drop** command defines that HTTP and HTTPS connection attempts to hosts in the blacklist should be blocked.

Example 12-42 also demonstrates a sequence of activities related to the DNS protocol in the BTF environment of Figure 12-24:

- The client 172.21.21.21 queries the DNS server about a host it wants to contact.
- ASA intercepts the DNS response and compares the supplied domain-name with its dynamic database. In this case, the domain name and IP address are graylisted.
- The client establishes the TCP connection to the graylisted host. ASA could have been configured to treat graylisted host as if they were in the blacklist, a task that would require the **dynamic-filter ambiguous-is-black** command.

Example 12-42 *Classifying Traffic for BTF*

```
! BTF classification for DNS responses arriving on interface 'out'
access-list BTF-CLASSIFY1 extended permit tcp any any eq www
access-list BTF-CLASSIFY1 extended permit tcp any any eq https
!
dynamic-filter enable interface out classify-list BTF-CLASSIFY1
!
! BTF dropping for classified traffic

access-list BTF-DROP extended permit tcp any any eq www
access-list BTF-DROP extended permit tcp any any eq https
!
dynamic-filter drop blacklist interface out action-classify-list BTF-DROP
!
! DNS request, interception of the DNS reply and greylisting

%ASA-6-302015: Built outbound UDP connection 2049 for
out:200.221.11.100/53 (200.221.11.100/53) to inside:172.21.21.21/1033
(192.168.1.21/1033)
%ASA-6-338301: Intercepted DNS reply for name r.turn.com from
out:200.221.11.100/53 to inside:172.21.21.21/1033, matched blacklist
%ASA-6-302016: Teardown UDP connection 2049 for out:200.221.11.100/53
to inside:172.21.21.21/1033 duration 0:00:00 bytes 491
%ASA-4-338202: Dynamic Filter monitored greylisted TCP traffic from
inside: 172.21.21.21/1507 (192.168.1.21/1507) to out:69.194.244.11/80
(69.194.244.11/80), destination 69.194.244.11 resolved from dynamic
list: r.turn.com, threat-level: low, category: Ads
```



```
%ASA-6-302013: Built outbound TCP connection 2050 for
out:69.194.244.11/80 (69.194.244.11/80) to inside:172.21.21.21/1507
(192.168.1.21/1507)
!
ASA# show asp table dynamic-filter hits
Context: single_vf
Address 69.194.244.11 mask 255.255.255.255 name: r.turn.com flags:
0x44 hits 2
!
ASA# show dynamic-filter data | begin Active
Active rules in Dynamic Filter asp table:
Dynamic data: 2 domain names , 3278 IPv4 addresses
Local data: 0 domain names , 0 IPv4 addresses

! Information about DNS Snooping

ASA# show dynamic-filter dns-snoop
DNS Reverse Cache Summary Information:
32 addresses, 75 names
Next housekeeping scheduled at 18:38:25 UTC Mar 8 2011,
!
! Searching for a string in the dns-snoop table (DNS reverse cache)

ASA# show dynamic-filter dns-snoop detail | include cisco
[cisco.112.207.net] type=0, ttl=867
[newsroom.cisco.com] type=0, ttl=12766
[ output suppressed ]
[www.cisco.com] type=0, ttl=0
```

Note The default **threat-level range** in the **dynamic-filter drop** command is from *moderate* to *very high*. A description of the categories associated to each of these levels is shown in Example 12-41 (**show dynamic-filter data** command).

Example 12-43 illustrates a situation in which a connection attempt to a host in the blacklist was blocked by the BTF mechanism. As opposed to Example 12-42 (in which the host was only graylisted), the HTTP connection is not allowed through.

Example 12-44 ends this section by showing information related to BTF statistics and reports. In the reports section that, from the BTF standpoint, an access attempt to a blacklisted site is enough to consider the host as infected.

Example 12-43 *Blocking Traffic with BTF*

```

! Sample traffic block using the BTF feature
ASA-6-302015: Built outbound UDP connection 2085 for
out:200.221.11.100/53 (200.221.11.100/53) to inside:172.21.21.21/1033
(192.168.1.21/1033)
%ASA-6-338301: Intercepted DNS reply for name www.lefty.org from
out:200.221.11.100/53 to inside:172.21.21.21/1033, matched blacklist
%ASA-5-338302: Address 64.62.200.84 discovered for domain
www.lefty.org from blacklist. Adding rule
%ASA-6-338301: Intercepted DNS reply for name lefty.org from
out:200.221.11.100/53 to inside:172.21.21.21/1033, matched blacklist
%ASA-6-302016: Teardown UDP connection 2085 for out:200.221.11.100/53 to
inside: 172.21.21.21/1033 duration 0:00:00 bytes 158
%ASA-4-338002: Dynamic Filter monitoredblacklisted TCP traffic from inside:
172.21.21.21/1525 (192.168.1.21/1525) to out:64.62.200.84/80
(64.62.200.84/80), destination 64.62.200.84 resolved from dynamic
list: lefty.org, threat-level: very-high, category:
Malware
%ASA-4-338006: Dynamic Filter dropped blacklisted TCP traffic from
inside: 172.21.21.21/1525 (192.168.1.21/1525) to out:64.62.200.84/80
(64.62.200.84/80), destination 64.62.200.84 resolved from
dynamic list: lefty.org, threat-level: very-high,
category: Malware
!
ASA# show asp table dynamic-filter hits | include lefty
Address 64.62.200.84 mask 255.255.255.255 name: lefty.org flags:
0x41 hits 3

```

Example 12-44 *BTF Statistics and Reports*

```

! Information about the DNS inspection policy
ASA# show service-policy interface inside inspect dns
Interface inside:
  Service-policy: BTF-POLICY1
    Class-map: DNS-BTF
      Inspect: dns _default_dns_map dynamic-filter-snoop, packet 173, drop 0,
reset-drop 0
        dns-guard, count 45
        protocol-enforcement, drop 0
        nat-rewrite, count 0
    !
! BTF Statistics

```

```
ASA# show dynamic-filter statistics
Enabled on interface out using classify-list BTF-CLASSIFY1
Total conns classified 17, ingress 0, egress 17
Total whitelist classified 0, ingress 0, egress 0
Total greylist classified 11, dropped 0, ingress 0, egress 11
Total blacklist classified 6, dropped 3, ingress 0, egress 6
!
! BTF Reports

ASA# show dynamic-filter reports top malware-sites
Malware Sites (since last clear)
Site                               Connections  Logged  Dropped Threat-level Category
-----
64.62.200.84 (lefty.org)           3          3      3 very-high Malware
70.85.196.211 (buildtraffic.com)   2          0      0 low Ads
173.236.31.204 (support.buildtraffic.com) 1          0      0 low Ads
!

ASA# show dynamic-filter reports top malware-ports
Malware Ports (since last clear)
Port                               Connections  Logged
-----
tcp 80                             5
tcp 443                             1
!

ASA# show dynamic-filter reports top infected-hosts
Infected Hosts (since last clear)
Host                               Connections  Logged
-----
172.21.21.21 (inside)              6
```

Note Static entries are created in the BTF database with the **dynamic-filter blacklist** and **dynamic-filter whitelist** commands. Manually defined blacklist entries are always considered to have a high threat-level.

Summary

This chapter analyzed some application-oriented inspection mechanisms available on Cisco network firewalls. This set of upper-layer resources enhance the generic (L4-based) inspection and basically serve three purposes:

- Adapt the appropriate parameters of application protocols that include IP addressing information above Layer 3, by performing NAT inside the IP payload.
- Watch the negotiation of secondary channels so that the dynamic opening of Layer 4 ports, which are necessary for correct protocol transport across stateful firewalls, can be achieved.
- Use the application knowledge to implement more powerful inspection for some of these protocols.

The Modular Policy structure analyzed in the current chapter will be revisited in the next one for the specific demand of inspecting protocols within the Unified Communications context.

Chapter 17, “Firewall Interactions,” includes a study on how to integrate firewalls and Intrusion Prevention Systems (IPS) devices, a classic way of using inspection capabilities to see even further.

Further Reading

TCP/IP Illustrated, Volume 1: The Protocols (W. Richard Stevens)

<http://www.pearsonhighered.com/educator/product/TCPIP-Illustrated-Volume-1-The-Protocols/9780201633467.page>

TCP/IP Illustrated, Volume 2: The Implementation (Gary R. Wright, W. Richard Stevens)

<http://www.pearsonhighered.com/educator/product/TCPIP-Illustrated-Volume-2-The-Implementation/9780201633542.page>

This page intentionally left blank

Inspection of Voice Protocols

This chapter covers the following topics:

- Introduction to Voice Terminology
- Skinny Protocol
- H.323 Framework
- Session Initiation Protocol - SIP
- MGCP Protocol
- Cisco IP Phones and Digital Certificates
- Advanced Voice Inspection with ASA TLS-Proxy
- Advanced Voice Inspection with ASA Phone-Proxy

“I have noticed that nothing I never said ever did me any harm.”—Calvin Coolidge

Chapter 12, “Application Inspection,” devotes a great deal of attention to the study of application-level inspection on both IOS-based firewalls and on the Adaptive Security Appliance (ASA). This chapter applies this knowledge to the specific tasks related to protecting IP Telephony (IPT) signaling protocols.

As a matter of fact, the main objective of the chapter is to create some awareness about the particularities of the classic IPT signaling protocols and how firewalls can successfully deal with them. Some of the motivations behind the idea of dedicating a whole chapter to voice follow:

- Convergent networks that can simultaneously carry data, voice, and video are a fact and not simply a trend anymore. Organizations now understand the power of collaboration for business efficiency and productivity. As a result, deployments are on the rise. Given that these new services are materialized with a specific set of application

protocols, security professionals then need to start becoming acquainted with the characteristics, requirements, and challenges associated with such protocols.

- A great part of the terminology is common to all the signaling protocols. If you are not familiar with the subject, this is a chance to start building a mental picture of the IPT concepts that help during project discussions with the voice professionals. Further, if you are aware of protocol behavior through firewalls, you can confidently propose security measures that prove helpful for real-life implementations.
- You can combine some of the protocols to provide more complete solutions. Confining the topic in a single place makes your work more organized and knowledge acquisition simpler.
- Before starting the actual analysis, it is convenient to notice that the ASA has been chosen as the platform for the topics covered in this chapter. This is a result of the following basic facts:
- ASA provides a rich (and always evolving) feature set for dealing with IPT and Unified Communications (UC) technologies.
- ASA offers great visibility of what occurs behind the scenes for signaling sessions. This is especially relevant when the topic is potentially new to you.
- The chapter is not focused on L7 filtering activities. Rather, it is aimed at providing insight about protocol behavior when crossing stateful firewalls. The ability to create L7 policies for the IPT Protocols naturally derives from the combination of the topics discussed hereafter with the techniques examined in the previous chapter.

Although this book is concerned with firewalls, they are just one of the recommended protection elements for an UC security solution. For instance, switch, router, and IP phone security features should be taken into account for any practical implementation.

Introduction to Voice Terminology

Skinny, SIP, MGCP, H.323, H.225, RTP, RTCP.... People who are not familiar with voice technology and find out that their firewall supports such a myriad of protocols, eventually might end up asking, Do I need all that to implement an IPT solution?

The answer is “no.” First, you need to separate those protocols in categories and understand that some of them might be used to accomplishing similar tasks. For instance, SIP, Skinny, MGCP, and the H.225 are call signaling protocols, whereas RTP and RTCP deal with media sessions (a natural follow-on to the signaling process). Although firewalls do not need to inspect RTP or RTCP, they are supposed to obtain RTP/RTCP information from the analysis of signaling protocol messages.

H.323 is actually a suite of protocols. Its call setup protocol is H.225, whereas H.245 deals with call control. (Many other protocols are in the H.323 Framework.) H.323 offers a direct and a centralized call signaling model (using an element known as a *gatekeeper*).

Skinny, an abridged reference to the Skinny Client Control Protocol (SCCP), is a *line-side* protocol, meaning that it is used by Call Agents (such as the Cisco Unified Communications Manager [CUCM]) to control endpoints such as IP Phones and audio conferencing stations. H.323 protocols are typically used by the Call Agent when Skinny stations need to communicate with terminals outside of its original cluster. This integration can happen by means of H.323 *trunks*.

The Session Initiation Protocol (SIP) is both a line-side and a trunk-side protocol. A Call Agent can control IP terminals using SIP and also integrate with elements external to its cluster by means of SIP.

The Media Gateway Control Protocol (MGCP) is focused on controlling media gateways (elements that translate between media types, like IP to TDM) from a central point (Call Agent or, equivalently, media gateway controller, using MGCP nomenclature). The implementation of the MGCP centralized architecture, in which the call intelligence resides on a single element, is common on service provider environments to control the analog and digital ports on voice gateways.

The term *gateway* is not exclusive to MGCP. Within the context of Voice over IP (VoIP) deployments, a voice gateway establishes the interface between the VoIP network and non-IP systems, such as the Public Switched Telephone Network (PSTN), FAX equipment, or an analog phone. Many Cisco IOS routers support integrated voice gateway functionality.

The Real-Time Transport Protocol (RTP) is in charge of carrying the voice or video streams after the call is set up by a signaling protocol. It was designed to enable receivers to compensate from *jitter* (delay variation) and out-of-order packets that might be introduced by IP networks. The Real Time Control Protocol (RTCP) is frequently used in conjunction with RTP and is concerned with transmitting control packets to participants of a given RTP session.

Cisco Unified Communications Manager (CUCM) supports all the signaling protocols mentioned and, therefore, it can be used as a kind of interworking element for different classes of IP-based voice networks. When communicating with legacy systems, CUCM would still rely on the functionality provided by gateways. A Call Agent like CUCM would sometimes be called an IP-PBX.

It is worth mentioning that Cisco also offers a router-based version of CUCM known as Cisco Communications Manager Express (CCME).

Figure 13-1 is an attempt to create a correspondence between the traditional telephony world and the new IP-based systems that materialize UC.

If this introduction intimidates you, do not give up. *There is a high probability that you are not the only one....* Read each of the following sections and pay close attention to the proposed examples. It will become easier for you to connect the dots.

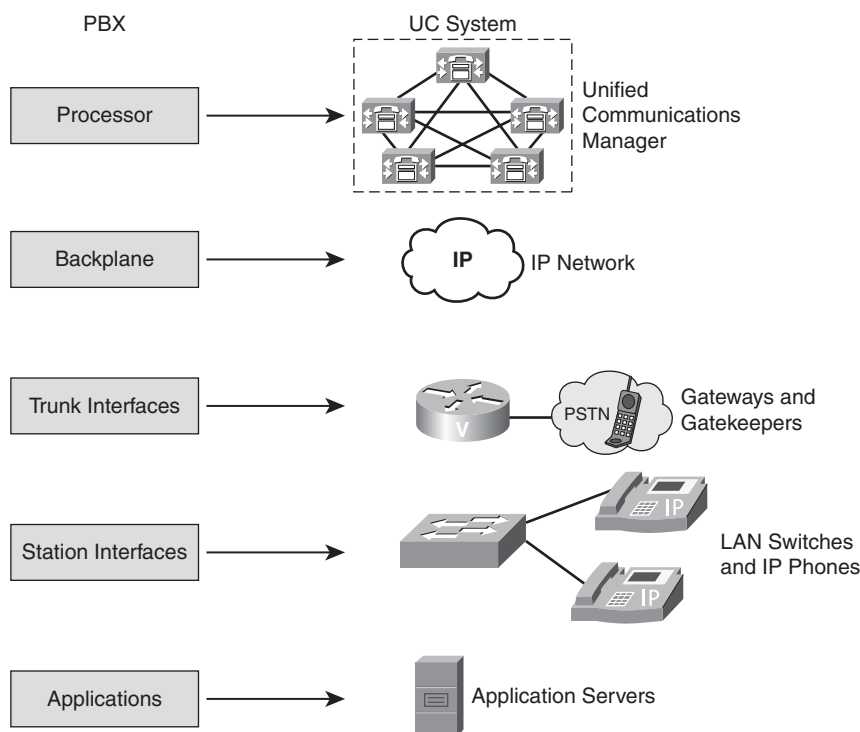


Figure 13-1 Overview of UC

Skinny Protocol

Cisco IP Phones use a lightweight protocol named Skinny Client Control Protocol (SCCP) to communicate with their Call Agent, CUCM. Following registration with CUCM, Skinny-based phones exchange SCCP messages with the CUCM for all call setup activities. Skinny is basically a *line-side* signaling protocol and actual SCCP deployments normally rely on the H.323 framework for establishing trunk connections.

ASA watches SCCP message exchange, which, by default, happens over port TCP/2000, to determine the permissions that should be dynamically created for the negotiated media ports (RTP/RTCP). Skinny inspection is enabled by default on ASA's global policy.

Figure 13-2 depicts the reference scenario for Skinny analysis. Example 13-1 relates to this figure and brings some auxiliary configurations:

- Port TCP/2000 is reserved for Skinny registration and call signaling. Port TCP/2443 is the Skinny over TLS option, which is analyzed later as a provider of secure signaling.
- DHCP is the classic option for IP phone addressing. In the scenario, ASA has been configured as the DHCP Server on interfaces 'phone1' and 'phone3'. DHCP Option 150 corresponds to the IP address of the Trivial File Transfer Protocol (TFTP) server.

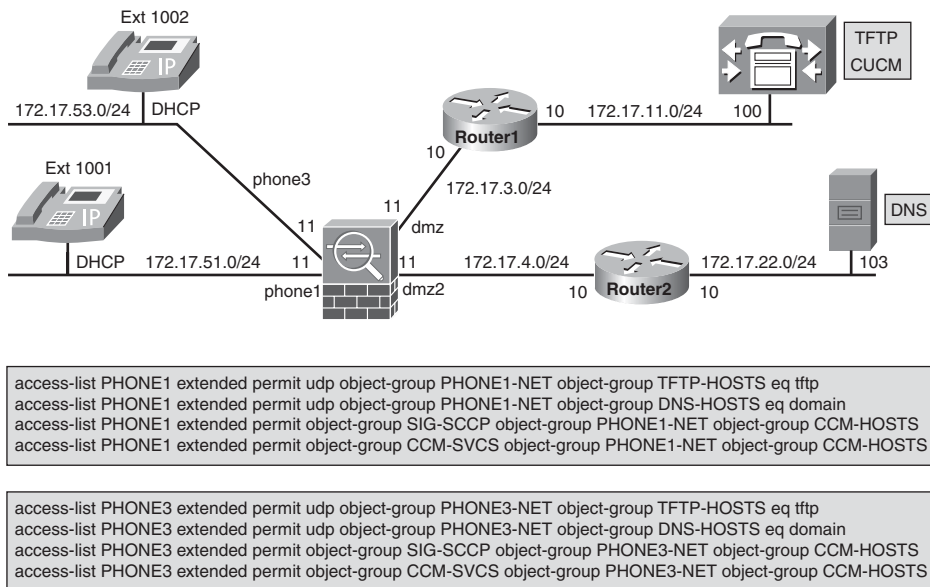


Figure 13-2 *Reference Topology for Skinny Protocol Analysis*

- TFTP is a basic component of CUCM deployments. Its main responsibilities are the delivery of loads (firmware) and configuration files for IP phones. Cisco recommends the TFTP service to coexist with a CUCM server or, alternatively, to dedicate one of the CUCM servers within a cluster to TFTP.
- The DNS protocol is used for resolution of the CUCM name.
- The **object-group** CCM-SVCS is configured to allow access to additional options like extension mobility (HTTP on port 8080) and certificate services on the Certification Authority Proxy Function (CAPF) port (TCP/3804). CAPF is studied later in the chapter.

Figure 13-3 summarizes the Skinny registration process, highlighting the tasks associated with the DHCP, TFTP, DNS, and SCCP services. This process is described in the following list:

1. The IP phone receives a DHCP response containing its own IP address, the address of DNS server, and the address (or name) of the TFTP server.
2. The IP phone gets its configuration from the TFTP server.
3. The IP phone uses DNS to resolve CUCM's name.
4. The Skinny registration process starts on server port TCP/2000. Some important Skinny registration messages are documented in Example 13-2. In this example, the beginning of each Skinny message is indicated by the "Proxy mode with X bytes of data" text, whereas its final line is characterized by "Proxy forward X bytes, total X".

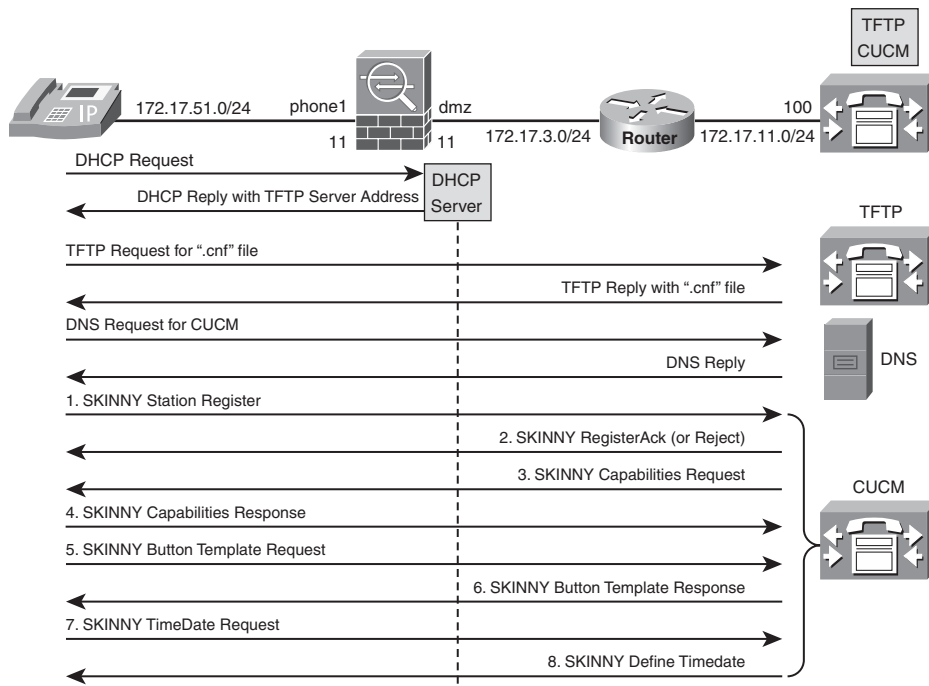


Figure 13-3 Registration Steps for a Skinny-Based IP Phone

Example 13-1 Baseline Configuration for the Analysis of the Skinny Protocol

```
! Service object-groups (used throughout this chapter)
object-group service SIG-SCCP
  service-object tcp eq 2000
  service-object tcp eq 2443
!
object-group service CCM-SVCS
  service-object tcp eq 3804
  service-object tcp eq 8080
!
! Network object-groups (used throughout this chapter)

object-group network PHONE1-NET
  network-object 172.17.51.0 255.255.255.0
!
object-group network PHONE3-NET
  network-object 172.17.53.0 255.255.255.0
!
object-group network DNS-HOSTS
  network-object host 172.17.22.103
```

```

!
object-group network TFTP-HOSTS
  network-object host 172.17.11.100
!
object-group network CCM-HOSTS
  network-object host 172.17.11.100
!
! ASA acting as DHCP server on interface 'phone1'

dhcpd domain uc.lab.bsa
dhcpd address 172.17.51.101-172.17.51.109 phone1
dhcpd dns 172.17.22.103 interface phone1
dhcpd lease 86400 interface phone1
dhcpd option 150 ip 172.17.11.100 interface phone1
dhcpd enable phone1

```

Example 13-2 Registration Process for a Skinny Phone

```

! Registration messages seen with 'debug skinny' enabled
SKINNY:: Proxy mode with 104 bytes of data
SKINNY:: 05:12:10 PM received packet from phone3:172.17.53.103/50176
to dmz:172.17.11.100/2000
SKINNY:: *****StationAlarmMessageID, 104 bytes
SKINNY:: Embedded IP addr 172.17.53.103
SKINNY:: incoming packet
SKINNY:: this conn: l_addr = 172.17.11.100 l_port = 2000 foreign
global IP = 172.17.53.103 foreign global port = 50176
SKINNY:: Proxy forward 104 bytes, total 104

SKINNY:: Proxy mode with 76 bytes of data
SKINNY:: 05:12:10 PM received packet from phone3:172.17.53.103/50176
to dmz:172.17.11.100/2000
SKINNY:: *****StationRegisterMessageID, 76 bytes
SKINNY:: Embedded station IP addr 172.17.53.103
SKINNY:: incoming packet
SKINNY:: this conn: l_addr = 172.17.11.100 l_port = 2000 foreign
global IP = 172.17.53.103 foreign global port = 50176
SKINNY:: Endpoint supports skinny version 11
SKINNY:: Proxy forward 76 bytes, total 76

SKINNY:: Proxy mode with 32 bytes of data
SKINNY:: 05:12:10 PM received packet from dmz:172.17.11.100/2000 to
phone3:172.17.53.103/50176
SKINNY:: StationRegisterAckMessageID, 32 bytes

```

```

SKINNY:: Endpoint supports skinny version 19
SKINNY:: This session is going to use SCCP version 11
SKINNY:: Proxy forward 32 bytes, total 32
[ output suppressed ]

SKINNY:: Proxy mode with 16 bytes of data
SKINNY:: 05:12:10 PM received packet from phone3:172.17.53.103/50176
to dmz:172.17.11.100/2000
SKINNY:: StationRegisterAvailableLinesMessageID, 16 bytes
SKINNY:: Proxy forward 16 bytes, total 16
[ output suppressed ]

SKINNY:: Proxy mode with 48 bytes of data
SKINNY:: 05:12:10 PM received packet from dmz:172.17.11.100/2000 to
phone3:172.17.53.103/50176
SKINNY:: StationDefineTimeDateID, 48 bytes
SKINNY:: Proxy forward 48 bytes, total 48
[ output suppressed ]

SKINNY:: Proxy mode with 12 bytes of data
SKINNY:: 05:12:36 PM received packet from phone3:172.17.53.103/50176
to dmz:172.17.11.100/2000
SKINNY:: StationKeepAliveMessageID, 12 bytes
SKINNY:: Proxy forward 12 bytes, total 12
!
! Statistics about Skinny inspection

ASA1# show service-policy global inspect skinny
Global policy:
  Service-policy: global_policy
    Class-map: inspection_default
      Inspect: skinny , packet 149245, drop 0, reset-drop 0
        tcp-proxy: bytes in buffer 0, bytes dropped 0

```

Figure 13-4 exemplifies some additional Skinny messages and the type of control they provide. For instance, there are messages used to define the meaning of phone buttons (*ButtonTemplateMessage*) and Soft Keys (*SoftKeyTemplateResMessage*) and others in charge of informing about the codecs supported by the IP Phone (*CapabilitiesResMessage*). It is interesting to notice that ASA is aware of these messages (refer to Example 13-2).

After successful registration on CUCM through ASA, IP Phones are ready to place calls. The relevant message exchange for call setup is shown in Figure 13-5.

Internet Protocol, Src: 172.17.11.100 , Dst: 172.17.51.101 Transmission Control Protocol, Src Port: 2000 , Dst Port: 49873 Skinny Client Control Protocol Data Length: 4 Reserved: 0x00000000 Message ID: CapabilitiesReqMessage (0x0000009b)	Internet Protocol, Src: 172.17.11.100 , Dst: 172.17.51.101 Transmission Control Protocol, Src Port: 2000 , Dst Port: 49873 Skinny Client Control Protocol Data Length: 100 Reserved: 0x00000000 Message ID: ButtonTemplateMessage (0x00000097) ButtonOffset: 0 ButtonCount: 6 TotalButtonCount: 6 InstanceNumber: One (0x01) ButtonDefinition: Line (0x09) InstanceNumber: Two (0x02) ButtonDefinition: Line (0x09) InstanceNumber: One (0x01) ButtonDefinition: SpeedDial (0x02) InstanceNumber: Two (0x02) ButtonDefinition: SpeedDial (0x02) InstanceNumber: Three (0x03) ButtonDefinition: SpeedDial (0x02) [...] InstanceNumber: Zero (0x00) ButtonDefinition: Undefined (0xff)
Internet Protocol, Src: 172.17.51.101, Dst: 172.17.11.100 Transmission Control Protocol, Src Port: 49873, Dst Port: 2000 Skinny Client Control Protocol Data Length: 8 Reserved: 0x00000000 Message ID: HeadsetStatusMessage (0x0000002b) Headset Mode: HeadsetOff (2)	
Internet Protocol, Src: 172.17.51.101, Dst: 172.17.11.100 Transmission Control Protocol, Src Port: 49873, Dst Port: 2000 Skinny Client Control Protocol Data Length: 136 Reserved: 0x00000000 Message ID: CapabilitiesResMessage (0x00000010) CapCount: 8 PayloadCapability: Wideband 256k (25) MaxFramesPerPacket: 120 PayloadCapability: G.711 u-law 64k (4) MaxFramesPerPacket: 40 PayloadCapability: G.711 A-law 64k (2) MaxFramesPerPacket: 40 PayloadCapability: G.729 Annex B (15) MaxFramesPerPacket: 60 PayloadCapability: G.729 Annex A+Annex B (16) MaxFramesPerPacket: 60 PayloadCapability: G.729 (11) MaxFramesPerPacket: 60 PayloadCapability: G.729 Annex A (12) MaxFramesPerPacket: 60 PayloadCapability: RFC2833_DynPayload (257) MaxFramesPerPacket: 4	Internet Protocol, Src: 172.17.11.100 , Dst: 172.17.51.101 Transmission Control Protocol, Src Port: 2000 , Dst Port: 49873 Skinny Client Control Protocol Data Length: 656 Reserved: 0x00000000 Message ID: SoftKeyTemplateResMessage (0x00000108) SoftKeyOffset: 0 SoftKeyCount: 20 TotalSoftKeyCount: 20 SoftKeyLabel: \200\001 SoftKeyEvent: Redial (1) SoftKeyLabel: \200\002 SoftKeyEvent: NewCall (2) SoftKeyLabel: \200\003 SoftKeyEvent: Hold (3) SoftKeyLabel: \200\004 [...]

Figure 13-4 Some Additional Skinny Messages

Example 13-3 reveals ASA's perspective on a Skinny call between two IP Phones located on distinct subnets:

- The **show skinny** command provides insight about the initial situation, in which only the control channels are established, and later reveals the audio connections just built.
- Syslog information associated with the calling phone (172.17.51.101) shows the SCCP messages inspected by ASA to determine the media ports that need to be opened.
- The **show conn detail** command displays the control connections (at the bottom), which have 1 hour as the default timeout value. Given that Skinny employs *keepalive* messages, this control channel can remain established for a long time (as long as events such as **clear conn** or phone resets do not take place). This avoids the need for rebuilding the signaling channel for each call.
- The **show conn detail** command unveils the existence of a connection with flag 'k', corresponding to Skinny media. It is worth comparing the UDP port numbers in this entry with those on the output of the **show skinny** command. The various entries with flags 'ki' are associated with temporary Skinny media (sometimes referred to as *early media*).

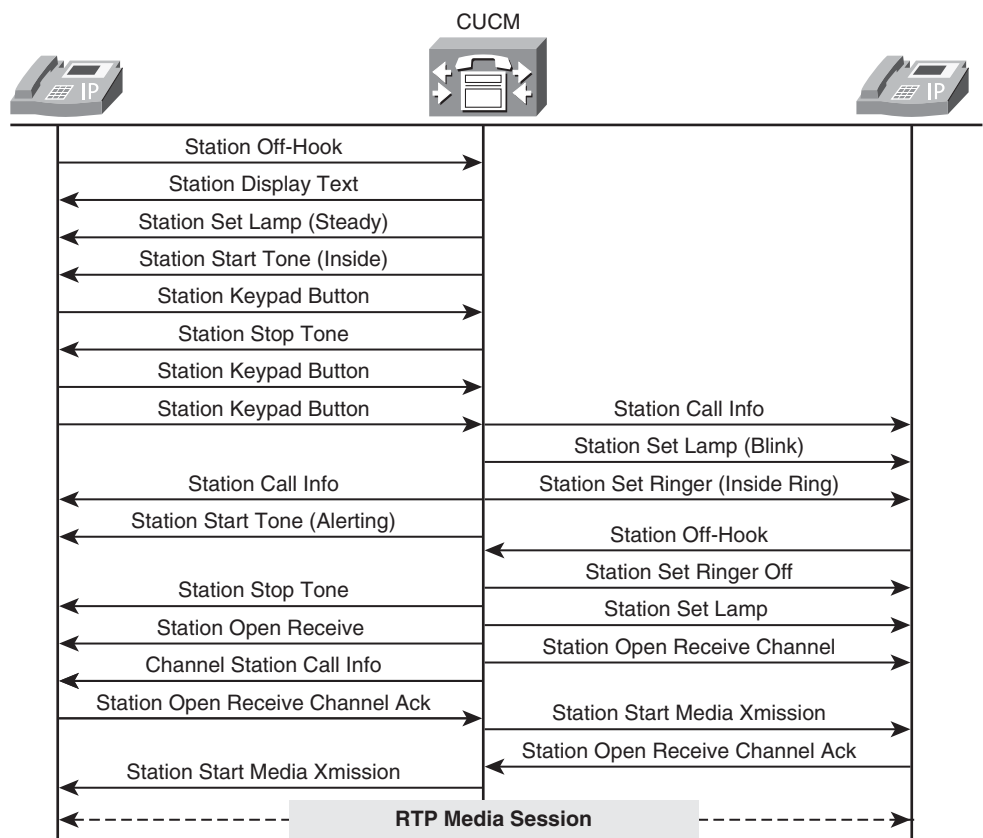


Figure 13-5 Message Exchange for a Skinny Call Involving Two IP Phones

It is convenient to mention that if the phones are both on the same subnet, there is no entry with flag ‘k’ in the connection table. This relates to the fact that, in this specific scenario, there is no media session (between IP Phones) traversing ASA. Nonetheless, the multiple ‘ki’ entries would still be there.

Example 13-4 provides a different perspective on the call initially analyzed in Example 13-3. In the second case, `debug skinny` is turned on to provide more details about SCCP call setup tasks.

Example 13-3 ASA’s Perspective on a Skinny Call (1)

```
! Initial situation (control connections with CUCM established)
ASA1# show skinny
LOCAL                                FOREIGN                                STATE
```

```

-----
1      172.17.11.100/2000      172.17.53.103/50178      0
2      172.17.11.100/2000      172.17.51.101/50138      0
!
! Channels for RTP (media) and RTCP (media control) are created

%ASA-6-608001:Pre-allocate Skinny RTP secondary channel for
phone1:172.17.51.101/21646 to dmz:172.17.11.100 from
*****StationOpenReceiveChannelAckID message
%ASA-6-608001:Pre-allocate Skinny RTCP secondary channel for
phone1:172.17.51.101/21647 to dmz:172.17.11.100 from
*****StationOpenReceiveChannelAckID message
%ASA-6-608001:Pre-allocate Skinny RTP secondary channel for
phone1:172.17.51.101/21646 to phone3:0.0.0.0 from
*****StationStartMediaTransmissionID message
!
! Details about the connection table
ASA1# show conn detail
14 in use, 211 most used
Flags: A - awaiting inside ACK to SYN, a - awaiting outside ACK to
SYN,
      B - initial SYN from outside, b - TCP state-bypass or nailed,
C - CTIQBE media,
      D - DNS, d - dump, E - outside back connection, F - outside
FIN, f - inside FIN,
      G - group, g - MGCP, H - H.323, h - H.225.0, I - inbound data,
      i - incomplete, J - GTP, j - GTP data, K - GTP t3-response
      k - Skinny media, M - SMTP data, m - SIP media, n - GUP
      O - outbound data, P - inside back connection, p - Phone-proxy
TFTP connection,
      q - SQL*Net data, R - outside acknowledged FIN,
      R - UDP SUNRPC, r - inside acknowledged FIN, S - awaiting
inside SYN,
      s - awaiting outside SYN, T - SIP, t - SIP transient, U - up,
      V - VPN orphan, W - WAAS,
      X - inspected by service module
UDP phone3:172.17.53.103/32695 dmz:172.17.11.100/0,
      flags ki, idle 49s, uptime 49s, timeout 5m0s, bytes 0
UDP phone3:172.17.53.103/32694 dmz:172.17.11.100/0,
      flags ki, idle 49s, uptime 49s, timeout 5m0s, bytes 0
UDP phone3:172.17.53.103/32695 phone1:172.17.51.101/0,
      flags ki, idle 49s, uptime 49s, timeout 5m0s, bytes 0
UDP phone3:172.17.53.103/32694 phone1:172.17.51.101/21646,
      flags k, idle 0s, uptime 49s, timeout 5m0s, bytes 857420

```



```
UDP phone3:172.17.53.103/0 phone1:172.17.51.101/21647,
  flags ki, idle 49s, uptime 49s, timeout 5m0s, bytes 0
UDP phone3:172.17.53.103/0 phone1:172.17.51.101/21646,
  flags ki, idle 49s, uptime 49s, timeout 5m0s, bytes 0
UDP phone1:172.17.51.101/21647 dmz:172.17.11.100/0,
  flags ki, idle 49s, uptime 49s, timeout 5m0s, bytes 0
UDP phone1:172.17.51.101/21646 dmz:172.17.11.100/0,
  flags ki, idle 49s, uptime 49s, timeout 5m0s, bytes 0
TCP phone3:172.17.53.103/50178 dmz:172.17.11.100/2000,
  flags UIOB, idle 27s, uptime 43m40s, timeout 1h0m, bytes 10704
TCP phone1:172.17.51.101/50138 dmz:172.17.11.100/2000,
  flags UIOB, idle 13s, uptime 1h39m, timeout 1h0m, bytes 15800
!
ASA1# show skinny
      LOCAL                                FOREIGN                                STATE
-----
1      172.17.11.100/2000                  172.17.53.103/50178                    2
  AUDIO 172.17.51.101/21646              172.17.53.103/32694
2      172.17.11.100/2000                  172.17.51.101/50138                    2
  AUDIO 172.17.53.103/32694              172.17.51.101/21646
```

Example 13-4 *ASA's Perspective on a Skinny call (2)*

```
! Media connection creation as revealed by 'debug skinny'
SKINNY:: Proxy mode with 32 bytes of data
SKINNY:: 06:28:40 PM received packet from phone1:172.17.51.101/50138
to dmz:172.17.11.100/2000
SKINNY:: *****StationOpenReceiveChannelAckID, 32 bytes
SKINNY::Embedded IP addr 172.17.51.101 & port 21646 for this RTP
endpoint
Created media session for PTPID:16777261 and CR:0
SKINNY:: trying to allocate conn for PTPID: 16777261 CR: 0 media
channels: side =
sidOUTSIDE
    local IP    = 0.0.0.0    local RTP = 0    local RTCP = 0
    foreign IP  = 172.17.51.101    foreign RTP = 21646    foreign RTCP =
21647
SKINNY:: The foreign endpoint IP or both endpoint IPs for the media
channels are
unknown
-> wait for info of the other endpoint, so return now
SKINNY:: RTP ports of both endpoints are unknown
-> so return now
SKINNY:: trying to allocate conn for PTPID: 16777261 CR: 0 media
```

```

channels: side =
sidOUTSIDE
    local IP    = 172.17.11.100  local RTP = 0   local RTCP = 0
    foreign IP  = 172.17.51.101  foreign RTP = 21646  foreign RTCP =
21647
SKINNY:: both endpoint IPs are known
-> let's open the 2 media channels
SKINNY:: Proxy forward 32 bytes, total 32

SKINNY:: Proxy mode with 120 bytes of data
SKINNY:: 06:28:40 PM received packet from dmz:172.17.11.100/2000 to
phone3:172.17.53.103/50178
SKINNY:: *****StationStartMediaTransmissionID, 120 bytes
SKINNY::Embedded remote RTP endpoint's IP addr 172.17.51.101 & port
21646
Created media session for PTPID:16777262 and CR:0
SKINNY:: RTP ports of both endpoints are unknown
-> so return now
SKINNY:: trying to allocate conn for PTPID: 16777262 CR: 0 media
channels: side =
sidINSIDE
    local IP    = 172.17.51.101  local RTP = 21646  local RTCP = 21647
    foreign IP  = 0.0.0.0        foreign RTP = 0     foreign RTCP = 0
SKINNY:: Proxy forward 120 bytes, total 120

SKINNY:: Proxy mode with 32 bytes of data
SKINNY:: 06:28:40 PM received packet from phone3:172.17.53.103/50178
to dmz:172.17.11.100/2000
SKINNY:: *****StationOpenReceiveChannelAckID, 32 bytes
SKINNY::Embedded IP addr 172.17.53.103 & port 32694 for this RTP
endpoint
Found media session for PTPID:16777262 and CR:0
SKINNY:: trying to allocate conn for PTPID: 16777262 CR: 0 media
channels:          side =    sidOUTSIDE
    local IP    = 172.17.51.101  local RTP = 0     local RTCP = 0
    foreign IP = 172.17.53.103  foreign RTP = 32694  foreign RTCP =
32695
SKINNY:: both endpoint IPs are known
-> let's open the 2 media channels
SKINNY:: trying to allocate conn for PTPID: 16777262 CR: 0 media
channels:          side =    sidOUTSIDE
    local IP    = 172.17.51.101  local RTP = 21646  local RTCP = 21647
    foreign IP  = 172.17.53.103  foreign RTP = 0     foreign RTCP = 0
SKINNY:: both endpoint IPs are known

```

```

-> let's open the 2 media channels
SKINNY:: both endpoint IPs are known
-> but RTP & RTCP channels are NOT open
-> t_addr (local IP) = 172.17.51.101 t_port (local RTP) = 21646
-> f_addr (foreign IP) = 172.17.53.103 f_port (foreign RTP) = 0
SKINNY:: trying to allocate conn for PTPID: 16777262 CR: 0 media
channels:          side = sidOUTSIDE
    local IP      = 172.17.11.100 local RTP = 0 local RTCP = 0
    foreign IP    = 172.17.53.103 foreign RTP = 32694 foreign RTCP =
32695
SKINNY:: both endpoint IPs are known
-> let's open the 2 media channels
SKINNY:: Proxy forward 32 bytes, total 32

SKINNY:: Proxy mode with 120 bytes of data
SKINNY:: 06:28:40 PM received packet from dmz:172.17.11.100/2000 to
phone1:172.17.51.101/50138
SKINNY:: *****StationStartMediaTransmissionID, 120 bytes
SKINNY::Embedded remote RTP endpoint's IP addr 172.17.53.103 & port
32694
Found media session for PTPID:16777261 and CR:0

```

H.323 Framework

The H.323 standard is a set recommendations developed by the Telecommunication Standardization Sector of the International Telecommunication Union (ITU-T) to enable IP-based networks to concurrently transport audio, video, and data traffic. The H.323 definitions deal with both point-to-point and multipoint communications and include topics such as call control, bandwidth management, multimedia management, and integration of the IP domain with other types of networks. Examples of these non-H.323 networks are Integrated Services Digital Network (ISDN) and the PSTN.

The suite of protocols that constitute the H.323 framework are represented, for the sake of reference, in Figure 13-6. New standards may be added to this, and explaining each of the elements in this set is beyond the scope of this book.

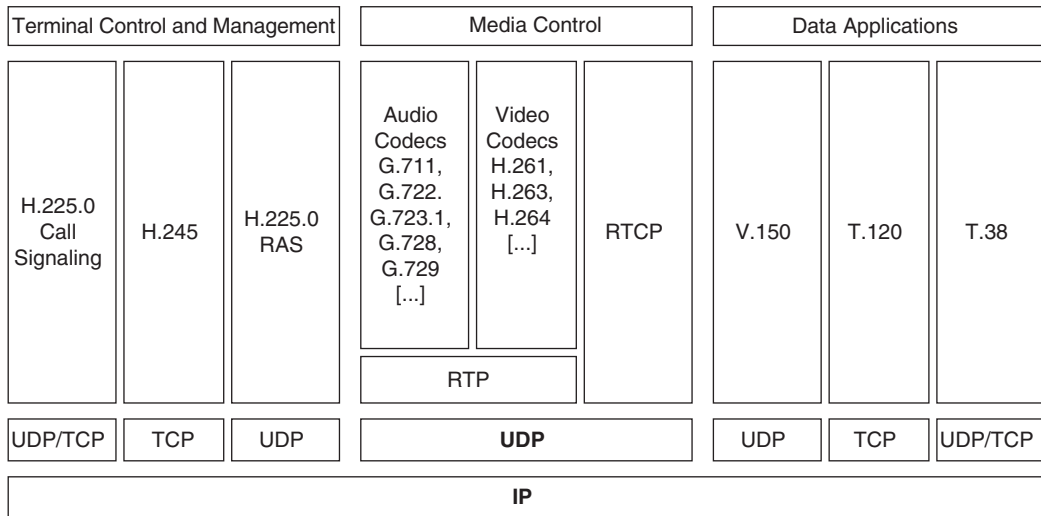


Figure 13-6 *H.323 Protocol Suite*

The H.323 defines four basic components on a multimedia network:

- **Terminals:** User endpoints that communicate with each other. Two simple examples are Cisco IP Phones and video conferencing units.
- **Gateways:** Used to communicate with terminals residing on other types of networks such as the PSTN (which is circuit-switched).
- **Gatekeepers:** Used to implement centralized dial-plan. Can offer services such as Call Admission Control, Call Routing, and some kinds of security policies.
- **Multipoint Control Units (MCUs):** Used to set up conferences that involve multiple participants. Among the typical tasks of an MCU, some deserve special mention:
 - H.245 capabilities negotiation
 - Controlling conference resources
 - Mixing and splitting media streams (audio and video)
 - Performing transcoding when there is a mismatch of codecs or sampling rates for two given media streams

Among the protocols comprising the H.323 suite, some are of particular importance for VoIP/IPT networks:

- **H.225:** Used for call setup and termination on H.323 environments. Inspection of H.225 is enabled by default on ASA global policy over TCP port 1720 (**inspect h323 h225**).

- **H.245:** Controls traffic flow, capabilities negotiation, and allocation of media channels, among other tasks. The TCP port used for H.245 during a call is obtained from initial H.225 inspection.
- **H.225 RAS:** Registration, Admission, and Status (RAS) messages are used for H.323 scenarios that involve gatekeepers. RAS inspection is turned on by default on ASA global policy for UDP port 1719 (`inspect h323 ras`). The RAS inspection also takes care of Gatekeeper Discovery messages sent over UDP Port 1718. These messages are transmitted in the form of multicast to the reserved group 224.0.1.41 and are used when the H.323 endpoints do not have a statically assigned gatekeeper.

H.323 supports a distributed architecture, enabling (but not requiring) voice gateways to maintain local call routing information. Two classes of dial plans are exemplified:

1. CUCM is in charge of call control for extensions 10XX and points directly to an IOS Voice gateway when it needs to reach an extension 11XX (and conversely). This is depicted in Figure 13-7. Such an approach is not the ideal choice for large networks because every gateway would need specific **dial-peer** information locally configured to reach destination-patterns on the other gateways.

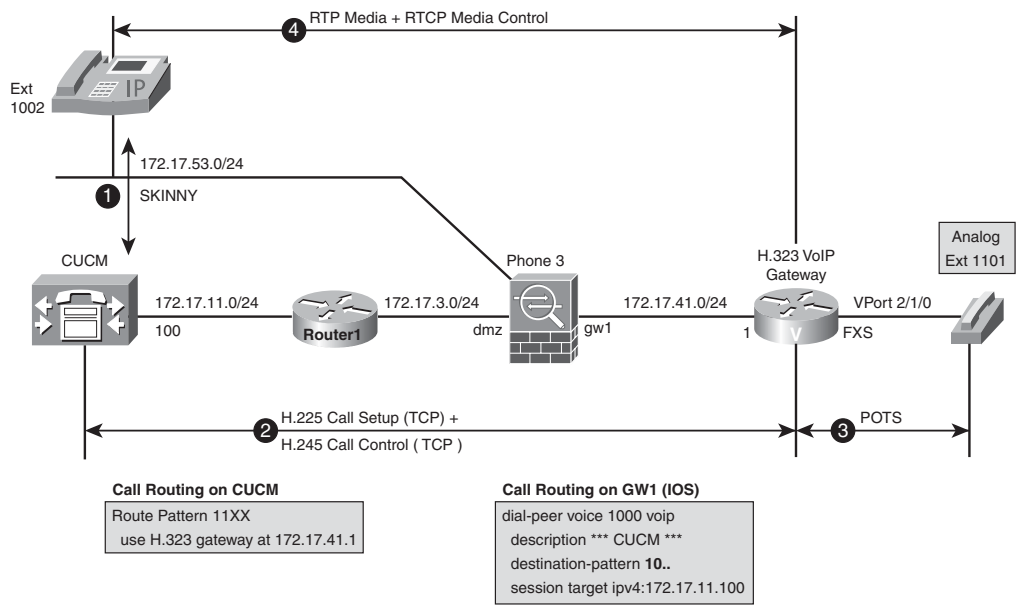


Figure 13-7 *Reference Topology for the Analysis of an H.323 Direct Call*

2. CUCM, CCME1, and GW1 are responsible for call control for some local extensions each and rely on an H.225 RAS gatekeeper when they need to reach remote prefixes. A sample scenario for this centralized dial plan is shown in Figure 13-8.

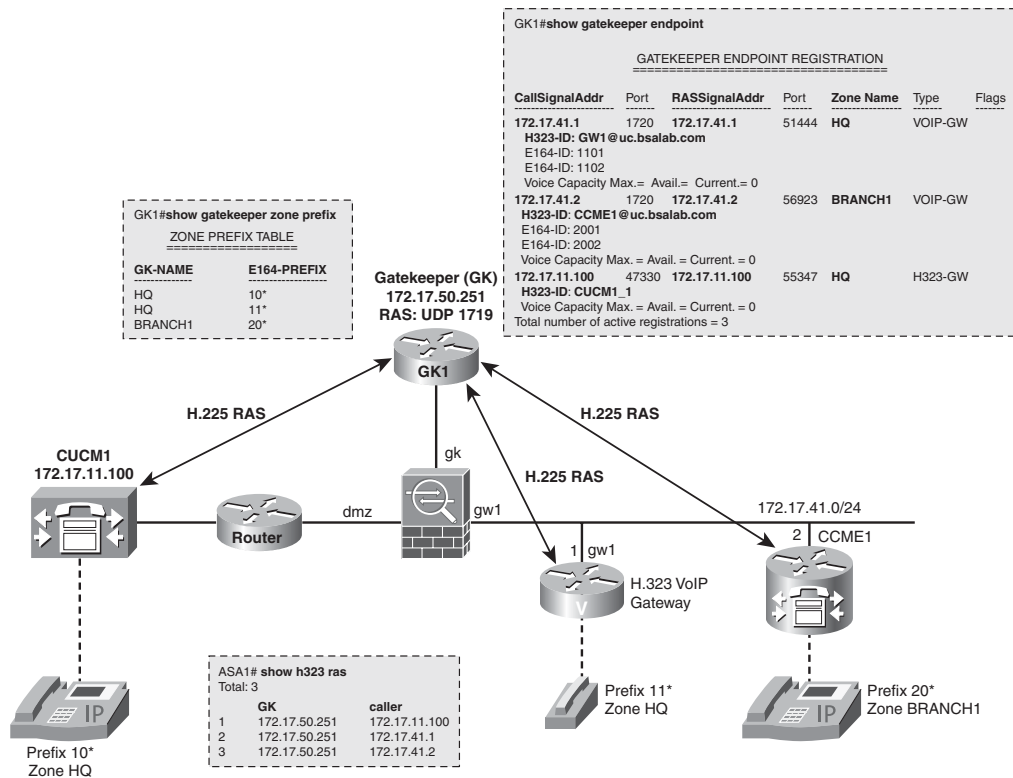


Figure 13-8 Reference Topology for a Gatekeeper-Controlled H.323 Scenario

The two examples just mentioned respectively map to the H.323 call routing models known as Direct Call Signaling and Gatekeeper-routed Call Signaling. These models are discussed in the following sections.

H.323 Direct Calls

Figure 13-7 proposes a scenario for the analysis of H.323 direct calls. The main operations involved in this approach are summarized below for a call initiated by the IP Phone 1002 and directed to the analog phone 1101 (behind gateway 172.17.41.1):

1. The IP Phone talks to the CUCM over its pre-established Skinny control channel.
2. CUCM connects to port TCP/1720 (H.225 Call Setup) on the remote gateway. ASA inspects the H.225 channel and establishes the appropriate H.245 connection used for Call Control. This second TCP connection is unveiled with the **show h245** command on ASA and with the **show tcp brief** command on the gateway side, as documented on Example 13-5.

- 3. The Voice gateway sets up a Plain Old Telephone Service (POTS) call to the analog phone.
- 4. The UDP connections for RTP media and RTCP media control are created by H.245. It is important to notice that the UDP connections are between the IP phone (172.17.53.103) and the voice gateway (172.17.41.1).

Example 13-6 employs **debug** commands to provide additional information about the call analyzed in Example 13-5. The establishment of the H.245 call control channel and the subsequent allocation of media and media control ports are readily visible in the output.

Example 13-5 *ASA's Perspective on a Direct H.323 Call (1)*

```
! 1002 (IP Phone) calls analog phone 1101 (H.225 and H.245
connections are created)
%ASA-6-302013: Built inbound TCP connection 196701 for
dmz:172.17.11.100/50265 (172.17.11.100/50265) to gw1:172.17.41.1/1720
(172.17.41.1/1720)
%ASA-6-302003: Built H245 connection for faddr 172.17.11.100/0 laddr
172.17.41.1
%ASA-6-302013: Built inbound TCP connection 196702 for
dmz:172.17.11.100/50266 (172.17.11.100/50266) to
gw1:172.17.41.1/12981 (172.17.41.1/12981)
%ASA-6-302015: Built inbound UDP connection 196707 for
phone3:172.17.53.103/24993 (172.17.53.103/24993) to
gw1:172.17.41.1/19157 (172.17.41.1/19157)
%ASA-6-302003: Built H245 connection for faddr 172.17.11.100/0 laddr
172.17.41.1
!
ASA1# show h225
Total: 1
0          CRV: 4   Local: 172.17.41.1/1720 58          Foreign: 172.17.11.100/50265
225
1 Concurrent Call(s) for
    Local:   172.17.41.1/1720   Foreign: 172.17.11.100/50265
!
ASA1# show h245
Total: 1
          LOCAL                TPKT    FOREIGN                TPKT
0          172.17.41.1/12981      0      172.17.11.100/50266      0
!
! What the IOS voice gateway sees (RTP between IP phone and Voice GW)

GW1# show voip rtp connections detail
VoIP RTP active connections :
No. CallId      dstCallId LocalRTP RmtRTP LocalIP      RemoteIP
```

```

1    21          22          19156    24992  172.17.41.1
172.17.53.103
  callId 21 (dir=1):  called=1101 calling=1002 redirect=
    dest callId 22: called=1101 calling=1002 redirect=
      1 context 66F70E90 xmitFunc 61769C04
Found 1 active RTP connections
!
! What the IOS voice gateway sees (H.225 and H.245 with CUCM)

GW1# show tcp brief
TCB          Local Address          Foreign Address      (state)
68050F1C     172.17.41.1.1720             172.17.11.100.50265  ESTAB
6804F5F8     172.17.41.1.12981           172.17.11.100.50266  ESTAB

```

Example 13-6 ASA's Perspective on a Direct H.323 Call (2)

```

Proxy-mode with 225 bytes of data
H225::find_h225 hash -844033107
H225::Created global h225 for faddr 172.17.11.100/50265 laddr
172.17.41.1/1720, Total 0
H225::with hash number -844033107
H225::msg received from 172.17.11.100/50265 to 172.17.41.1/1720
      curr_tpkt (0) data_len (225) curr_state (0) last_msg (0) seq
(3642687449)
H225::Local TPKT 0 Foreign TPKT 225 LEN 225
H225::find_h225 hash -844033111
H225::Created h225 CRV 4 for faddr 172.17.11.100/50265 laddr
172.17.41.1/1720, Total 1, per call 1
H225::with hash number -844033111
H225::Decode Q931 message
Q931::Undefine value 0x4
Q931::Calling party number detected
Q931::Party number 1002
Q931::Called party number detected
Q931::Party number 1101
H225::Decoding without error
H225::SETUP received from 172.17.11.100/50265 to 172.17.41.1/1720
H225::Version 5 message detected
H225::TransportAddress embedded ip 172.17.11.100 port 1720
H225::TransportAddress embedded ip 172.17.11.100 port 1720
H225::Update sourceCallSigAddr 172.17.11.100 into H225_CONN_T
H225::curr_msg->tpkt_len 225,msg_count 0 total_offset 225

Proxy-mode with 45 bytes of data

```



```

H225::msg received from 172.17.41.1/1720 to 172.17.11.100/50265
      curr_tpkt (0) data_len (45) curr_state (0) last_msg (2) seq
(1160795076)
H225::Local TPKT 45 Foreign TPKT 225 LEN 45
H225::Decode Q931 message
H225::No need to inspect message Unknown(d), data_len 45

Proxy-mode with 76 bytes of data
H225::msg received from 172.17.41.1/1720 to 172.17.11.100/50265
      curr_tpkt (0) data_len (76) curr_state (0) last_msg (3) seq
(1160795121)
H225::Local TPKT 76 Foreign TPKT 225 LEN 76
H225::Decode Q931 message
H225::Decoding without error
H225::CALL_PROCEEDING received from 172.17.41.1/1720 to 172.17.11.100/50265
H225::Version 4 message detected
H225::TransportAddress embedded ip 172.17.41.1 port 12981
H225::TransportAddress embedded ip 172.17.41.1 port 12981
**CALL PROCEEDING H245 Addrss: 172.17.41.1/46386
H225::curr_msg->tpkt_len 76,msg_count 0 total_offset 76

Proxy-mode with 122 bytes of data
H245::Created_h245(): laddr 172.17.41.1/12981 faddr
172.17.11.100/50266, total 1
H245::Foreign TPKT 0, Local TPKT 0, data_len 122 curr tpkt = 0 dp 3
[ output suppressed ]

Proxy-mode with 24 bytes of data
H245::Foreign TPKT 0, Local TPKT 0, data_len 24 curr tpkt = 0 dp 3
H245::Decoding without error
H245::**OLC** received from 172.17.41.1/12981 to 172.17.11.100/50266
H245::Audio chosen in this olc
H245::Parsing H2250LCPs
H245::UnicastAddr: embedded ip 172.17.41.1, port 19157
H245::New IP 172.17.41.1/19157
H245::Look for matching SID 1
H245::Media with Session ID 1 not found
H245::Create a new media.
H245::set media is going to set Audio Media 0x1
FastStart: FALSE SID: 1 LCN: 1 Media(0xda170808)
      Foreign Media 0.0.0.0 media 0 mediaCtrl 0
      Local Media 0.0.0.0 media 0 mediaCtrl 0
      Inside OLCack receive FALSE, Outside OLCack receive FALSE
H245::Audio Media type 0x1 set for this session

```

```

H245::fix_H2250LCPS::No H323 media allocated
[ output suppressed ]
Proxy-mode with 27 bytes of data
H245::Foreign TPKT 0, Local TPKT 0, data_len 27 curr tpkt = 0 dp 3
H245::Decoding without error
H245::**OLCAck** received from 172.17.11.100/50266 to 172.17.41.1/12981
H245::Parsing H2250LCAPs
H245::UnicastAddr: embedded ip 172.17.53.103, port 24992
H245::New IP 172.17.53.103/24992
H245::OLCAck. Search for media by sessionID(1)
H245::Look for matching SID 1
H245::matching SID 1 found at index 0.
H245::set media is going to set Audio Media 0x1
FastStart: FALSE SID: 1 LCN: 1 Media(0xda170808)
    Foreign Media 172.17.53.103 media 24992 mediaCtrl 24993
    Local Media 172.17.41.1 media 19156 mediaCtrl 19157
    Inside OLCAck receive TRUE, Outside OLCAck receive TRUE
H245::Audio Media type 0x1 set for this session
H245::checking policy allow_data_type
H245::Attempt to create media channel
H245::media laddr 172.17.41.1 19156-19157, faddr 172.17.53.103 24992-
24993

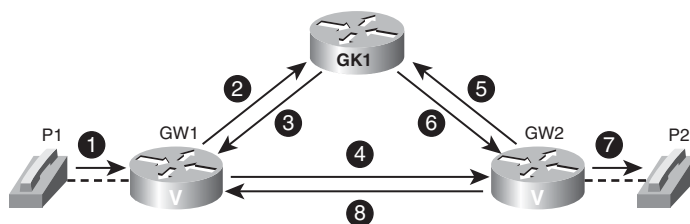
```

H.323 Calls Through a Gatekeeper

Figure 13-8 portrays an H.323 environment in which an IOS-based gatekeeper (GK1) has been deployed solely with the objective of providing centralized dial plan configuration. In this scenario, three types of call control elements (CUCM, CCME, and an IOS gateway) use H.225 RAS signaling to interact with GK1.

It is important to emphasize that CUCM, CCME1 and GW1 are only allowed to request gatekeeper services after successfully registering with GK1 (on port UDP 1719). Some other relevant facts documented in this figure follow:

- CUCM and GW1 are registered on Zone 'HQ'. As such, a call from 10* to 11* is deemed an *intrazone*. The detailed message exchange for this type of call is presented in Figure 13-9.
- A call from an IP phone registered on CCME1 (prefix 20*, belonging to zone 'BRANCH1') to either CUCM or GW1 will be considered *interzone*.
- GK1 has visibility of the registered Call Control elements and their associated zone prefixes.
- ASA knowledge of RAS sessions is characterized with the **show h323 ras** command.



1. P1 dials the Phone Number of P2.
2. GW1 sends an AdmissionRequest (ARQ) to GK1, requesting permission to call P2.
3. If GK1 finds P2 registered, it sends back an AdmissionConfirm (ACF) with the IP Address of GW2.
4. GW1 sends an H.225 Call Setup to GW2 with the Phone Number of P2.
5. GW2 sends an ARQ to GK1 requesting permission to answer call from GW1.
6. GK1 returns an ACF with the IP Address of GW1.
7. GW2 sets up a call to P2 (POTS or IP, depending on the type of Phone).
8. When P2 answers, GW2 sends an H.225 Connect message to GW1.

Figure 13-9 *Intrazone Call Setup in a Gatekeeper-Controlled H.323 Scenario*

Figure 13-10 represents the relevant connections (sometimes referred to as *call legs*) in a sample Gatekeeper-routed scenario. The typical sequence of operations for a call from extension 1001 (registered on CUCM) to 2002 (CCME-controlled Skinny phone) is described as follows:

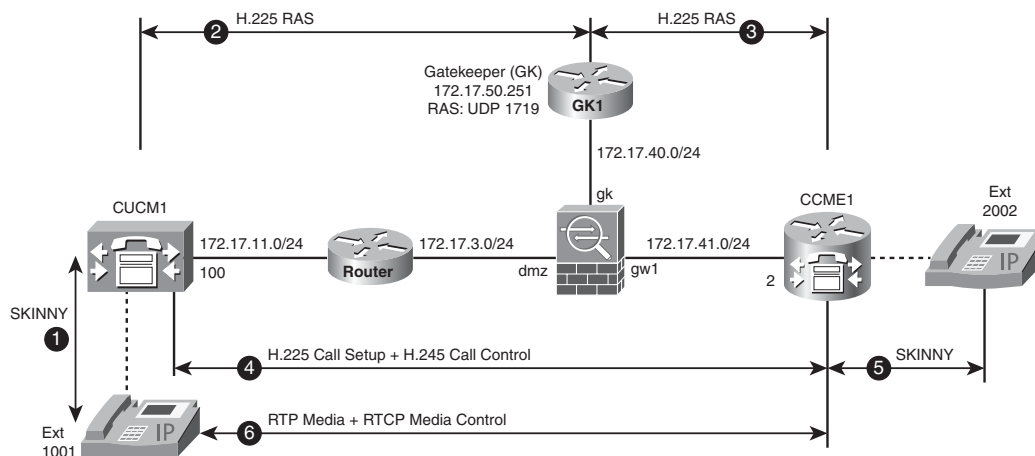


Figure 13-10 *Reference Topology for Call Between CUCM and CCME Through GK*

1. IP phone 1001 (172.17.51.101) uses SCCP to communicate with CUCM1 and requests a call setup to 2002 (remote Skinny phone registered on CCME1).

2. CUCM1 uses a RAS connection with GK1 to signal its intent of connecting to the remote phone 2002.
3. CCME1 also maintains a RAS connection with GK1.
4. After resolving the IP of CCME1 (with the aid of GK1), CUCM1 can open an H.225 connection (TCP/1720) to CCME1. The H.245 ports are dynamically derived from H.225 inspection.
5. CCME1 uses Skinny as the call control protocol for its local IP Phones.
6. The RTP and RTCP media connections are established between the originating IP Phone (172.17.51.101) and CCME1's address (172.17.41.2).

Example 13-7 assembles important information about the GK-routed call illustrated in Figure 13-10:

- There is an H.225 call signaling session from CUCM (172.17.11.100) to CCME1 (172.17.41.2) on TCP port 1720, followed by H.245 connection establishment. This is similar to what was presented in Example 13-5 but only made possible after consulting the Gatekeeper. It is worth noting that the H.225 RAS session includes *keepalive* messages so that it does not need to be rebuilt for every call. The **show conn detail** command, in this particular scenario, reveals that CUCM connection to GK1 (UDP/1719) has been up for more than three days.
- The UDP media (RTP/RTCP) connections are created between 172.17.51.101 and the remote gateway (172.17.41.2).
- The **show conn detail** displays the H.225 Call Setup and H.245 sessions, both having 5 minutes for their default timeout values.
- GK1 highlights that the calling number 1001 used 172.17.11.100 (CUCM) as its call signaling entity. This call was directed to 2002, which used 172.17.41.2 (CCME1) as the signaling element.
- The example also shows CCME1's perspective for RTP connections.

Example 13-7 ASA's Perspective on a H.323 Call Through a Gatekeeper

```
! 1001 (CUCM-controlled) calls 2002 (CCME-controlled) - H.225
signalling
%ASA-6-302012: Pre-allocate H225 Call Signalling Connection for faddr
172.17.11.100/0 to laddr 172.17.41.2
%ASA-6-302013: Built inbound TCP connection 48064 for
dmz:172.17.11.100/52095 (172.17.11.100/52095) to gw1:172.17.41.2/1720
(172.17.41.2/1720)
%ASA-6-302003: Built H245 connection for faddr 172.17.11.100/0 laddr
172.17.41.2
%ASA-6-302013: Built inbound TCP connection 48065 for
```

```

dmz:172.17.11.100/52096 (172.17.11.100/52096) to
gw1:172.17.41.2/49243 (172.17.41.2/49243)

! Pre-allocation of media channels
%ASA-6-608001:Pre-allocate Skinny RTP secondary channel for
gw1:172.17.41.2/18718 to phone1:0.0.0.0 from
****StationStartMediaTransmissionID message
%ASA-6-608001:Pre-allocate Skinny RTCP secondary channel for
gw1:172.17.41.2/18719 to phone1:0.0.0.0 from
****StationStartMediaTransmissionID message
%ASA-6-608001:Pre-allocate Skinny RTP secondary channel for
phone1:172.17.51.101/23838 to gw1:172.17.41.2 from
****StationOpenReceiveChannelAckID message
%ASA-6-608001:Pre-allocate Skinny RTCP secondary channel for
phone1:172.17.51.101/23839 to
gw1:172.17.41.2 from
****StationOpenReceiveChannelAckID message
%ASA-6-608001: Pre-allocate Skinny RTP secondary channel for
phone1:172.17.51.101/23838 to dmz:172.17.11.100 from
****StationOpenReceiveChannelAckID message
%ASA-6-608001:Pre-allocate Skinny RTCP secondary channel for
phone1:172.17.51.101/23839 to dmz:172.17.11.100 from
****StationOpenReceiveChannelAckID message

! Media channels are setup between IP phone 1001 (172.17.51.101) and
CCME1
%ASA-6-302015: Built inbound UDP connection 48068 for
phone1:172.17.51.101/23838 (172.17.51.101/23838) to
gw1:172.17.41.2/18718 (172.17.41.2/18718)
%ASA-6-302015: Built inbound UDP connection 48069 for
phone1:172.17.51.101/23839 (172.17.51.101/23839) to
gw1:172.17.41.2/18719 (172.17.41.2/18719)
!
ASA1# show conn detail
[ output suppressed ]
    G - group, g - MGCP, H - H.323, h - H.225.0, I - inbound data,
    i - incomplete, J - GTP, j - GTP data, K - GTP t3-response
    k - Skinny media, M - SMTP data, m - SIP media, n - GUP
[ output suppressed ]
UDP phone1:172.17.51.101/0 gw1:172.17.41.2/18719,
    flags ki, idle 12s, uptime 12s, timeout 5m0s, bytes 0
UDP phone1:172.17.51.101/0 gw1:172.17.41.2/18718,
    flags ki, idle 12s, uptime 12s, timeout 5m0s, bytes 0
TCP dmz:172.17.11.100/52096 gw1:172.17.41.2/49243,

```

```

    flags UfIOB, idle 12s, uptime 12s, timeout 5m0s, bytes 353
TCP dmz:172.17.11.100/52095 gw1:172.17.41.2/1720,
    flags UIOhB, idle 12s, uptime 18s, timeout 1h0m, bytes 587
UDP gw1:172.17.41.2/56923 gk:172.17.50.251/1719,
    flags H, idle 12s, uptime 3D18h, timeout 5m0s, bytes 1154055
UDP dmz:172.17.11.100/55347 gk:172.17.50.251/1719,
    flags H, idle 12s, uptime 3D18h, timeout 5m0s, bytes 1121779
UDP phone1:172.17.51.101/23839 dmz:172.17.11.100/0,
    flags ki, idle 12s, uptime 12s, timeout 5m0s, bytes 0
UDP phone1:172.17.51.101/23838 dmz:172.17.11.100/0,
    flags ki, idle 12s, uptime 12s, timeout 5m0s, bytes 0
UDP phone1:172.17.51.101/23839 gw1:172.17.41.2/18719,
    flags k, idle 1s, uptime 11s, timeout 5m0s, bytes 448
UDP phone1:172.17.51.101/23838 gw1:172.17.41.2/18718,
    flags k, idle 0s, uptime 12s, timeout 5m0s, bytes 40000
TCP phone1:172.17.51.101/49871 dmz:172.17.11.100/2000,
    flags UIOB, idle 5s, uptime 6D2h, timeout 1h0m, bytes 463988
!
! Gatekeeper's perspective on a call from 1001 (CUCM-bound) to 2002 (CCME1-controlled)
GK1# show gatekeeper calls
Total number of active calls = 1.

                        GATEKEEPER CALL INFO
                        =====
LocalCallID                Age(secs)    BW
10-199                      24           16(Kbps)
Endpt(s): Alias            E.164Addr
src EP: CUCM1_1            1001
      CallSignalAddr Port RASSignalAddr Port
      172.17.11.100 47330 172.17.11.100 55347
Endpt(s): Alias            E.164Addr
dst EP: CCME1              2002
      CallSignalAddr Port RASSignalAddr Port
      172.17.41.2 1720 172.17.41.2 56923
!
! What CCME1 sees: RTP between IP Phone and voice GW (CCME1)

CCME1# show voip rtp connections detail
VoIP RTP active connections :
No. CallId    dstCallId LocalRTP RmtRTP LocalIP RemoteIP
1 33          34          18718 23838 172.17.41.2 172.17.51.101
callId 33 (dir=1): called=2002 calling=1001 redirect=
dest callId 34: called=2002 calling=1001 redirect=
1 context 4A0A81D0 xmitFunc 40CEB054
Found 1 active RTP connections

```

This section presents as a reference for better understanding of RAS operations a set of sample messages exchanged between Call Controllers and their assigned gatekeeper. This can be seen in Figures 13-11 and 13-12:

- As the names imply, *registrationRequest* and *registrationConfirm* messages are used for registering with the gatekeeper. After successful registration, keepalives help on connection maintenance, and calls can then be signaled.
- *AdmissionRequest* and *AdmissionConfirm* messages are used for gateway address resolution and followed by an H.225.0 CS transaction for actual call setup. This GK-routed signaling process was previously demonstrated in Figure 13-9.

Sample Registration Request: CUCM to GK1

```
Internet Protocol, Src: 172.17.11.100, Dst: 172.17.50.251
User Datagram Protocol, Src Port: 55347, Dst Port: 1719
H.225.0 RAS
  RasMessage: registrationRequest (3)
    registrationRequest
      requestSeqNum: 8428
      protocolIdentifier: 0.0.8.2250.0.5 (Version 5)
      0..... discoveryComplete: False
      callSignalAddress: 1 item
        Item 0
          TransportAddress: ipAddress (0)
            ipAddress
              ip: 172.17.11.100 (172.17.11.100)
              port: 47330
      rasAddress: 1 item
        Item 0
          TransportAddress: ipAddress (0)
            ipAddress
              ip: 172.17.11.100 (172.17.11.100)
              port: 55347
      terminalType
      gatekeeperIdentifier: HQ
      endpointVendor
        vendor
          H.221 Manufacturer: Cisco (0xb5000012)
          productId: CiscoCallManager
          versionId: 1
      timeToLive: 60
      1..... keepAlive: True
      endpointIdentifier: 68BE6D7000000001
      0..... willSupplyUIEs: False
```

Sample Registration Confirm: GK1 to CCME1

```
Internet Protocol, Src: 172.17.50.251, Dst: 172.17.41.2
User Datagram Protocol, Src Port: 1719, Dst Port: 56923
H.225.0 RAS
  RasMessage: registrationConfirm (4)
    registrationConfirm
      requestSeqNum: 9440
      protocolIdentifier: 0.0.8.2250.0.4 (Version 4)
      callSignalAddress: 0 items
      gatekeeperIdentifier: BRANCH1
      endpointIdentifier: 68BE660800000003
      timeToLive: 60
      0. .... willRespondToIRR: False
      1. .... maintainConnection: True
```

Sample Admission Confirm: GK1 to CCME1

```
Internet Protocol, Src: 172.17.50.251, Dst: 172.17.41.2
User Datagram Protocol, Src Port: 1719, Dst Port: 56923
H.225.0 RAS
  RasMessage: admissionConfirm (10)
    admissionConfirm
      requestSeqNum: 9445
      bandwidth: 1280
      callModel: direct (0)
      direct: NULL
      destCallSignalAddress: ipAddress (0)
        ipAddress
          ip: 172.17.41.2 (172.17.41.2)
          port: 1720
      irrFrequency: 240
      0. .... willRespondToIRR: False
      uiesRequested
      usageSpec: 1 item
```

Figure 13-11 Sample H.225 RAS Messages

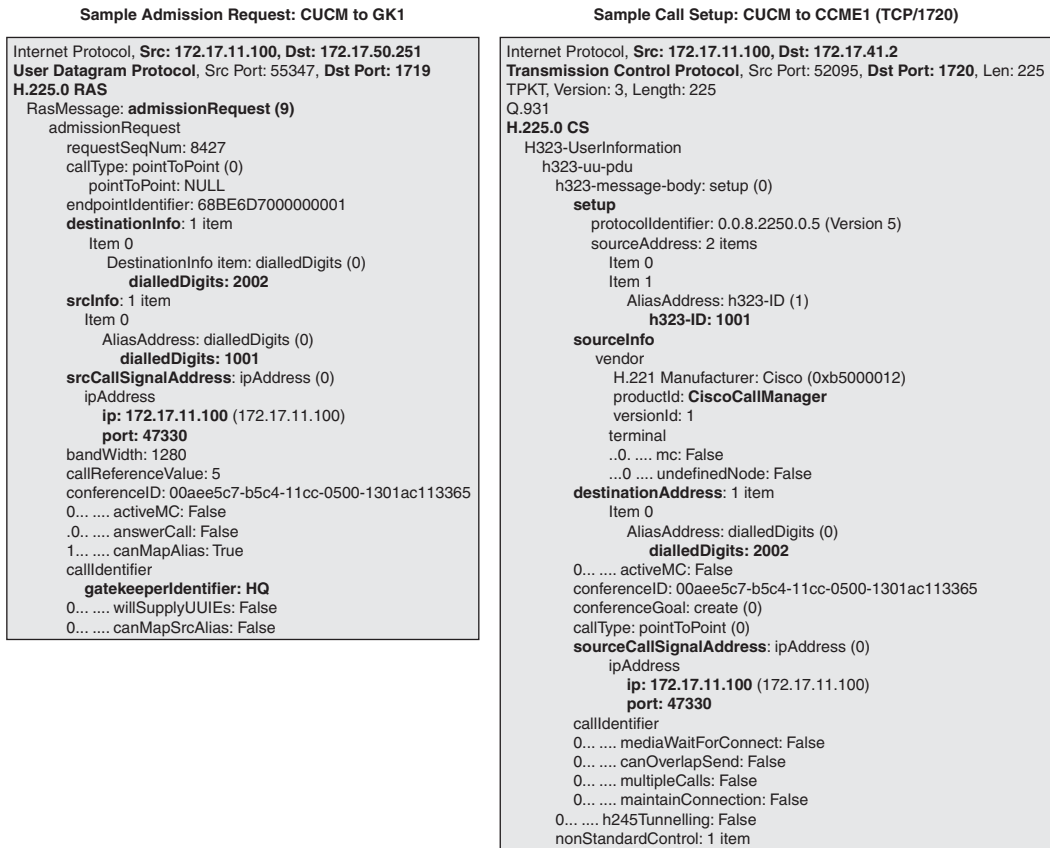


Figure 13-12 Sample H.225 Messages Involved on a Gatekeeper-Routed Call Setup

Session Initiation Protocol (SIP)

SIP is an application layer signaling protocol that can establish, modify, and terminate multimedia sessions (conferences) such as IP telephony calls and Internet video sessions. SIP offers the possibility of inviting participants to existing sessions, such as multicast conferences.

The H.323 suite has been developed by the ITU-T, whereas SIP is authored by the IETF and brings in its DNA some similarities with classic Internet protocols such as HTTP.

There are several (more than 50) IETF RFCs defining SIP and its possible interactions with other protocols to provide additional services. The following RFCs can be deemed the core SIP-related documents:

- RFC 3261: SIP: Session Initiation Protocol
- RFC 3262: Reliability of Provisional Responses in the Session Initiation Protocol

- RFC 3263: Session Initiation Protocol (SIP): Locating SIP Servers
- RFC 3264: An Offer/Answer Model with the Session Description Protocol (SDP)
- RFC 3265: Session Initiation Protocol (SIP) - Specific Event Notification
- RFC 2327: SDP: Session Description Protocol

SIP was conceived with client flexibility and integration of media types as some of its baseline requirements. For instance, SIP networks make it easier to combine services such as Instant Messaging (IM), IP Telephony, Video Telephony, and Video Conferencing. SIP also cares a lot about the concept of *presence* (on-hook/off-hook information for a phone, user login status for a chat application, user interest on being contacted by other elements of the SIP network and so on).

In a similar fashion to H.323, SIP enables distributed call control. A SIP-enabled Call Agent (such as CUCM) might be in charge of call control for a set of IP phones whereas an IOS-based SIP gateway is in use for connection to the PSTN. This SIP-based gateway would have locally defined dial-peers to contact outside destinations (those that are not directly controlled by the gateway, such as a remote IP phone or an analog phone reachable through a PSTN interface). Figure 13-13 depicts some of the main logical components of a SIP-based network:

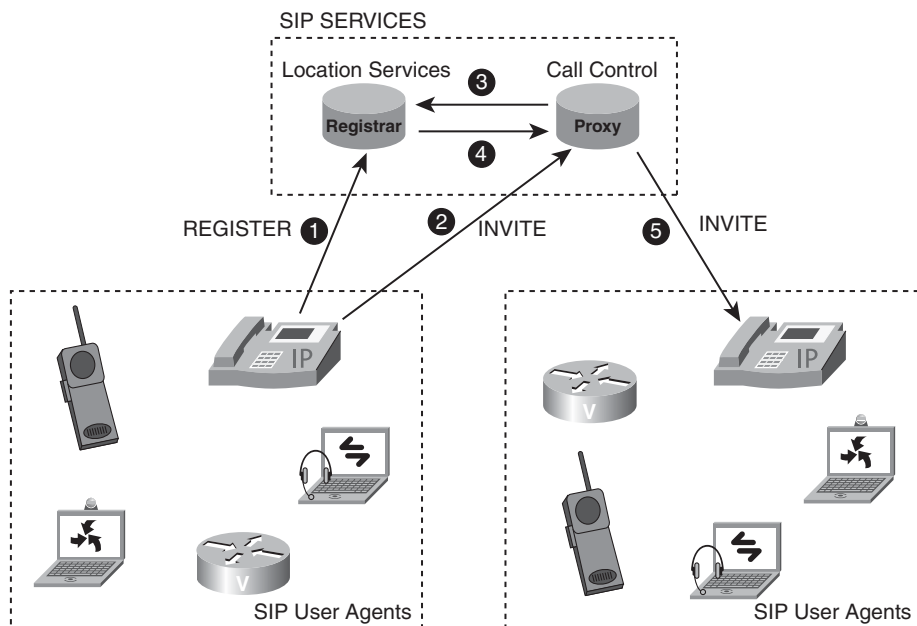


Figure 13-13 *Logical Components of a SIP-Based Network*

- **User Agents:** SIP endpoints are referred to as User Agents (UA). UAs can be IP phones, voice gateways, a SIP-based application on a PC, and so on. Each UA can act as a client (UAC), when it is initiating a call, or a server, in case it's receiving the call.
- **Registrar Server:** UAs register their location with a Registrar Server, which places this information in a location database (that can be local or reside on a dedicated service).
- **Proxy Server:** This type of server can perform services like call routing, address resolution, or authentication of UAs. In Figure 13-13, a UA sends its call setup messages through a proxy server. The proxy consults the registrar server to obtain the location of the destination endpoint (in this case, acting as an UAS).

Other possible logical elements that deserve specific mention follow:

- **Back-to-back User Agent (B2BUA):** This server behaves simultaneously as a UAC and a UAS. It terminates the call signaling from the calling party (UAC) and reinitiates signaling to the UAS (called entity).
- **Redirect Server:** Typically used for locating mobile users. The redirect server informs UAs and proxy servers that a given user has moved.
- **Presence Server:** Collects subscription and presence information and sends status notifications.

It is common to have a SIP component running more than one logical service. As an example, CUCM 8.x can act as a registrar server and as a B2BUA. Cisco routers can act as either redirect servers or as registrar servers.

Figure 13-13 also includes two important SIP messages: *REGISTER* (the main message for the registration process) and *INVITE* (the core call setup message).

Figure 13-14 represents the baseline topology for SIP phone registration on CUCM and subsequent SIP-based calls. Two basic activities relate to this scenario:

- Each SIP-based phone registers with CUCM, which, by default, happens on port UDP/5060. ASA's perspective on the SIP registration process is shown in Example 13-8 via the pertinent Syslog messages and the `debug sip` output. The default timeout for the SIP control connection is 30 minutes.
- Extension 1501 (172.17.51.102) calls 1502 (172.17.53.104). ASA's awareness of SIP messages (INVITE, 200, ACK) enables it to correctly allocate the UDP media channels, as illustrated in Example 13-9. The main media session through ASA carries the 'm' flag, whereas *early media* channels are flagged 'mi'.

Example 13-10 brings a complete SIP INVITE message, which corresponds to the main call setup component on a SIP-based environment. This output reveals, for instance, that the media port was obtained from SDP, the Session Description Protocol. The idea is to use this as a reference for later examples that present only simplified messages.

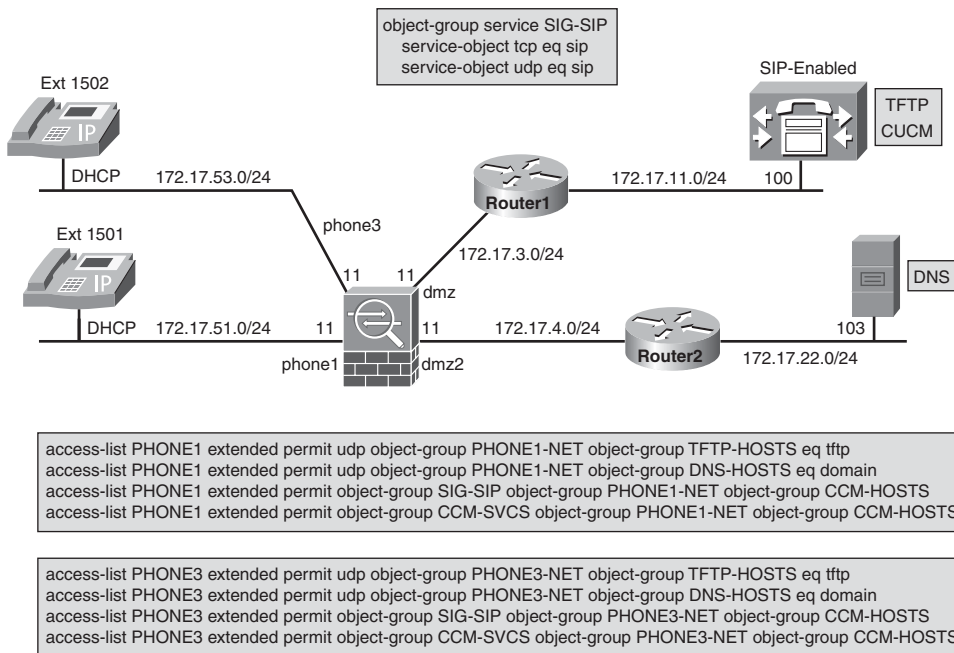


Figure 13-14 Reference Topology for the Analysis of Simple SIP Calls

Example 13-8 Sample SIP REGISTER Message as Seen by ASA

```
%ASA-6-607001: Pre-allocate SIP Via UDP secondary channel for
phone1:172.17.51.102/5060 to dmz:172.17.11.100 from REGISTER message
%ASA-6-302016: Teardown UDP connection 216826 for
phone1:172.17.51.102/5060 to dmz:172.17.11.100/0 duration 0:00:00
bytes 0
%ASA-6-607001: Pre-allocate SIP NOTIFY UDP secondary channel for
phone1:172.17.51.102/5060 to dmz:172.17.11.100 from 200 message
%ASA-6-302016: Teardown UDP connection 216822 for
phone1:172.17.51.102/5060 to dmz:172.17.11.100/0 duration 0:02:03
bytes 0

ASA1# show sip
Total: 2
call-id 00146948-196c0002-642a1413-66795e1e@172.17.51.102
CSeq: REGISTER
From: sip:1501@CUCM-803;00146948196c01e770532333-27b13ecb
To: sip:1501@CUCM-803;168937079
state Call init, timeout 0:03:00 idle 0:00:10
Transaction State Timeout
Idle
```

```

Cseq 586 REGISTER                                Transaction Proceeding0:03:00
0:00:10
!
! Details obtained from 'debug sip'

SIP::REGISTER received from phone1:172.17.51.102/50582 to
dmz:172.17.11.100/5060
    Found port 5060
Via Port 5060
SIP::Found User-Agent
    Found port 5060
SIP::Found Expires, 3600 seconds
SIP::regex engine has reached end of packet
SIP::Found CSeq 560 REGISTER
SIP::Found URI in request line "sip:CUCM-803" (12)
SIP::Found valid SIP URI: sip:1501@CUCM-803
SIP::Found From addr "sip:1501@CUCM-803" (17)
SIP::Found From addr tag "00146948196c01cd413e29da-456184bd" (33)
SIP::Found valid SIP URI: sip:1501@CUCM-803
SIP::Found To addr "sip:1501@CUCM-803" (17)
SIP::Found Via branch "z9hG4bK5a837305" (15)
SIP::Found Via addr "SIP/2.0/UDP
172.17.51.102:5060;branch=z9hG4bK5a837305" (53)
SIP::Found Max-Forwards 70
SIP::Found Call-ID 00146948-196c0002-642a1413-66795e1e@172.17.51.102
(49)
SIP::Found Expires, 3600 seconds
SIP::Found valid SIP URI: sip:15015060
SIP::Found Contact sip:15015060
SIP::Found Content-length 0
Created SIP session for phone1:172.17.51.102/50582 to
dmz:172.17.11.100/5060, 4 total
    From: sip:1501@CUCM-803 (17);tag=00146948196c01cd413e29da-
456184bd (33)
    To: sip:1501@CUCM-803 (17)
    Call-ID: 00146948-196c0002-642a1413-66795e1e@172.17.51.102
(49)
Created SIP Transaction for phone1:172.17.51.102/50582 to
dmz:172.17.11.100/5060
    Call-ID: 00146948-196c0002-642a1413-66795e1e@172.17.51.102
(49)
CSeq: 560 REGISTER
Branch: z9hG4bK5a837305

```

```
SIP::Updating xlate timeout for 172.17.51.102/5060 to 1:00:00
>>>> SIP::Payload not modified
SIP:: Forward 592 bytes, total 592
```

Example 13-9 Sample SIP Call Between Two IP Phones

```
! Extension 1501 (172.17.51.102) calls Extension 1502 (172.17.53.104)
%ASA-6-607001: Pre-allocate SIP SIGNALLING UDP secondary channel for
phone1:172.17.51.102/5060 to dmz:172.17.11.100 from INVITE message
%ASA-6-607001: Pre-allocate SIP Via UDP secondary channel for
phone1:172.17.51.102/5060 to dmz:172.17.11.100 from INVITE message
%ASA-6-607001: Pre-allocate SIP RTP secondary channel for
phone1:172.17.51.102/24364 to dmz:172.17.11.100 from INVITE message

%ASA-6-607001: Pre-allocate SIP SIGNALLING UDP secondary channel for
phone3:172.17.53.104/5060 to dmz:172.17.11.100 from 200 message
%ASA-6-302016: Teardown UDP connection 217593 for
phone3:172.17.53.104/0 to dmz:172.17.11.100/5060 duration 0:00:02
bytes 0
%ASA-6-607001: Pre-allocate SIP RTP secondary channel for
phone3:172.17.53.104/24046 to phone1:172.17.51.102 from ACK message
%ASA-6-607001: Pre-allocate SIP Via UDP secondary channel for
phone1:172.17.51.102/5060 to dmz:172.17.11.100 from ACK message
!
! SIP Media connections

ASA1# show conn
22 in use, 211 most used
UDP phone3 172.17.53.104:24047 phone1 172.17.51.102:0, idle 0:00:13,
bytes 0, flags mi
UDP phone3 172.17.53.104:24046 phone1 172.17.51.102:24364, idle
0:00:00, bytes 222052, flags m
UDP phone3 172.17.53.104:0 phone1 172.17.51.102:24365, idle 0:00:13,
bytes 0, flags mi
UDP phone3 172.17.53.104:0 phone1 172.17.51.102:24364, idle 0:00:13,
bytes 0, flags mi
UDP phone1 172.17.51.102:24365 dmz 172.17.11.100:0, idle 0:00:15,
bytes 0, flags mi
UDP phone1 172.17.51.102:24364 dmz 172.17.11.100:0, idle 0:00:15,
bytes 0, flags mi
```

Example 13-10 Sample SIP INVITE Message as Seen by ASA

```
SIP::INVITE received from phone1:172.17.51.102/50582 to
```

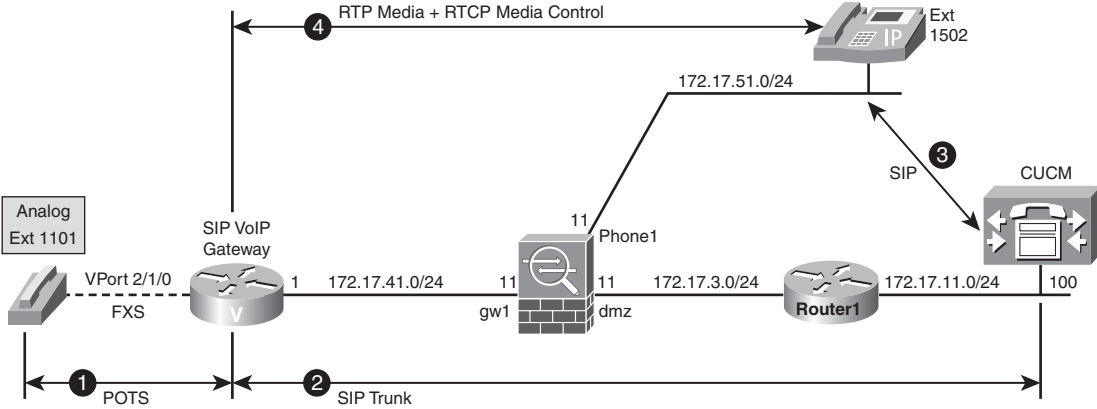
```

dmz:172.17.11.100/5060
    Found port 5060
Via Port 5060
SIP::Found User-Agent
    Found port 5060
SIP::Found Expires, 180 seconds
SIP: Media port 24364
SIP::media level connection addr 172.17.51.102, media port 24364
SIP::Embedded media port 24364 found in SDP with session IP
172.17.51.102
SIP::Non-session level connection addr 172.17.51.102, media port
24364
SIP::regex engine has reached end of packet
SIP::Found CSeq 101 INVITE
SIP::Found URI in request line "sip:1502@CUCM-803" (17)
SIP::Found valid SIP URI: sip:1501@CUCM-803
SIP::Found From addr "sip:1501@CUCM-803" (17)
SIP::Found From addr tag "00146948196c0253128a568a-73a1ffac" (33)
SIP::Found valid SIP URI: sip:1502@CUCM-803
SIP::Found To addr "sip:1502@CUCM-803" (17)
SIP::Found Via branch "z9hG4bK4915b96d" (15)
SIP::Found Via addr "SIP/2.0/UDP
172.17.51.102:5060;branch=z9hG4bK4915b96d" (53)
SIP::Found Max-Forwards 70
SIP::Found Call-ID 00146948-196c0003-43c56cc9-4f02040d@172.17.51.102
(49)
SIP::Found Expires, 180 seconds
SIP::Not updating database for Contact 172.17.51.102/5060, registry
database total 1
SIP::Found valid SIP URI: sip:15015060
SIP::Found Contact sip:15015060
SIP::Found Content-type application/sdp
SIP::Found Content-length 277
Created SIP session for phone1:172.17.51.102/50582 to
dmz:172.17.11.100/5060, 5 total
    From: sip:1501@CUCM-803 (17);tag=00146948196c0253128a568a-
73a1ffac (33)
    To: sip:1502@CUCM-803 (17)
    Call-ID: 00146948-196c0003-43c56cc9-4f02040d@172.17.51.102
(49)
Created SIP Transaction for phone1:172.17.51.102/50582 to
dmz:172.17.11.100/5060
    Call-ID: 00146948-196c0003-43c56cc9-4f02040d@172.17.51.102

```

```
(49)
      CSeq: 101 INVITE
      Branch: z9hG4bK4915b96d
>>>> SIP::Payload not modified
SIP:: Forward 958 bytes, total 958
```

Figure 13-15 proposes a reference topology for the analysis of SIP calls between an IP phone and an analog phone connected to a Foreign Exchange Station (FXS) port of a Cisco IOS voice gateway. It also shows the gateway’s perspective on this call:



```
GW1#show sip calls
SIP UAC CALL INFO
Call 1
SIP Call ID: 30D5951D-59A111D6-8058D9FC-94FF3FCE@172.17.41.1
State of the call : STATE_ACTIVE (7)
Substate of the call : SUBSTATE_NONE (0)
Calling Number : 1101
Called Number : 1502
Bit Flags : 0xC04018 0x100 0x80
CC Call ID : 41
Source IP Address (Sig) : 172.17.41.1
Destn SIP Req Addr:Port : 172.17.11.100:5060
Destn SIP Resp Addr:Port : 172.17.11.100:5060
Destination Name : 172.17.11.100
Number of Media Streams : 1
Number of Active Streams : 1
RTP Fork Object : 0x0
[ continues ]

[ continued ]
Media Mode : flow-through
Media Stream 1
State of the stream : STREAM_ACTIVE
Stream Call ID : 41
Stream Type : voice-only (0)
Negotiated Codec : g729r8 (20 bytes)
Codec Payload Type : 18
Negotiated Dtmf-relay : inband-voice
Dtmf-relay Payload Type : 0
Media Source IP Addr:Port : 172.17.41.1:17862
Media Dest IP Addr:Port : 172.17.51.103:19968
Orig Media Dest IP Addr:Port : 0.0.0.0:0
Options-Ping ENABLED:NO ACTIVE:NO
Number of SIP User Agent Client(UAC) calls : 1
SIP UAS CALL INFO
Number of SIP User Agent Server(UAS) calls : 0
```

Figure 13-15 Reference Topology for a SIP Call Between IP and Analog Phones

- Detailed UAC call information is presented because the call has been initiated by a phone connected to the gateway.
- There is no UAS call information at this moment. This would be reversed in case the call had been initiated by the IP phone and destined to the analog phone.

Example 13-11 supplies the basic information about the call from 1101 to the SIP-based IP phone 1502 in the scenario in Figure 13-15. In a completely analogous way to what was studied in Example 13-9, the inspection of the INVITE, 200, and ACK messages are the key elements for the opening of SIP media channels.

Example 13-11 *Sample SIP Call Between One Analog Phone and an IP Phone (1)*

```
! Extension 1101 (controlled by IOS-GW) calls extension 1502 (IP
Phone)
%ASA-6-302015: Built outbound UDP connection 223706 for
dmz:172.17.11.100/5060 (172.17.11.100/5060) to gw1:172.17.41.1/55195
(172.17.41.1/55195)

%ASA-6-607001: Pre-allocate SIP SIGNALLING UDP secondary channel for
gw1:172.17.41.1/5060 to dmz:172.17.11.100 from INVITE message
%ASA-6-607001: Pre-allocate SIP Via UDP secondary channel for
gw1:172.17.41.1/5060 to dmz:172.17.11.100 from INVITE message

%ASA-6-607001: Pre-allocate SIP SIGNALLING UDP secondary channel for
phone1:172.17.51.103/5060 to dmz:172.17.11.100 from 200 message
%ASA-6-302016: Teardown UDP connection 223711 for
phone1:172.17.51.103/0 to dmz:172.17.11.100/5060 duration 0:00:02
bytes 0

%ASA-6-607001: Pre-allocate SIP RTP secondary channel for
phone1:172.17.51.103/19968 to gw1:172.17.41.1 from ACK message
%ASA-6-302015: Built inbound UDP connection 223714 for
phone1:172.17.51.103/19969 (172.17.51.103/19969) to
gw1:172.17.41.1/17863 (172.17.41.1/17863)
!
! SIP Media connections seen on ASA (other connections not shown)

ASA1# show conn
UDP phone1 172.17.51.103:0 gw1 172.17.41.1:17863, idle 0:00:09, bytes
0, flags mi
UDP phone1 172.17.51.103:0 gw1 172.17.41.1:17862, idle 0:00:09, bytes
0, flags mi
UDP dmz 172.17.11.100:0 gw1 172.17.41.1:17863, idle 0:00:12, bytes 0,
flags mi
UDP dmz 172.17.11.100:0 gw1 172.17.41.1:17862, idle 0:00:12, bytes 0,
flags mi
UDP phone1 172.17.51.103:19969 gw1 172.17.41.1:17863, idle 0:00:00,
bytes 396,flags m
UDP phone1 172.17.51.103:19968 gw1 172.17.41.1:17862, idle 0:00:00,
bytes 30944,flags m
```



```

!
! What the IOS voice gateway (GW1) sees

GW1# show voip rtp connections detail
VoIP RTP active connections :
No. CallId      dstCallId  LocalRTP RmtRTP LocalIP      RemoteIP
1   41         40        17862    19968  172.17.41.1  172.17.51.103
  callId 41 (dir=2):  called=1502 calling=1101 redirect=
    dest callId 40: called= calling=1101 redirect=
      1 context 66F70E90 xmitFunc 61769C04
Found 1 active RTP connections

```

Example 13-12 registers the complete message exchange for establishing the SIP call presented in Example 13-11 (the messages, obtained from **debug sip**, were highly simplified to display only the most relevant information and save space).

Example 13-12 Sample SIP Call Between One Analog Phone and an IP Phone (2)

```

! SIP Message Exchange: IOS-GW to CUCM and CUCM to IP Phone
SIP::INVITE received from gw1:172.17.41.1/55195 to
dmz:172.17.11.100/5060
SIP::Found From addr "sip:1101@172.17.41.1" (20)
SIP::Found To addr "sip:1502@172.17.11.100" (22)
SIP::Found Call-ID 30D5951D-59A111D6-8058D9FC-94FF3FCE@172.17.41.1
(47)
SIP::Embedded media port 17862 found in SDP with session IP 172.17.41.1
SIP::Non-session level connection addr 172.17.41.1, media port 17862
!
SIP::100 received from dmz:172.17.11.100/5060 to gw1:172.17.41.1/5060
SIP::Found From addr "sip:1101@172.17.41.1" (20)
SIP::Found To addr "sip:1502@172.17.11.100" (22)
SIP::Found Call-ID 30D5951D-59A111D6-8058D9FC-94FF3FCE@172.17.41.1
(47)
!
SIP::INVITE received from dmz:172.17.11.100/5060 to
phone1:172.17.51.103/5060
SIP::Found From addr "sip:1101@172.17.11.100" (22)
SIP::Found To addr "sip:1502@CUCM-803" (17)
SIP::Found Call-ID 95e32400-ce3106d1-18-640b11ac@172.17.11.100 (43)
!
SIP::100 received from phone1:172.17.51.103/50804 to
dmz:172.17.11.100/5060
SIP::Found Call-ID 95e32400-ce3106d1-18-640b11ac@172.17.11.100 (43)
!

```

```

SIP::180 received from phone1:172.17.51.103/50804 to
dmz:172.17.11.100/5060
SIP::Found Call-ID 95e32400-ce3106d1-18-640b11ac@172.17.11.100 (43)
!
SIP::180 received from dmz:172.17.11.100/5060 to gw1:172.17.41.1/5060
SIP::Found Call-ID 30D5951D-59A111D6-8058D9FC-94FF3FCE@172.17.41.1
(47)
!
SIP::200 received from phone1:172.17.51.103/50804 to
dmz:172.17.11.100/5060
SIP::Embedded media port 19968 found in SDP with session IP
172.17.51.103
SIP::Non-session level connection addr 172.17.51.103, media port
19968
SIP::Found Call-ID 95e32400-ce3106d1-18-640b11ac@172.17.11.100 (43)

SIP::ACK received from dmz:172.17.11.100/5060 to
phone1:172.17.51.103/5060
SIP::Found Call-ID 95e32400-ce3106d1-18-640b11ac@172.17.11.100 (43)
SIP::Embedded media port 17862 found in SDP with session IP
172.17.41.1

SIP::183 received from dmz:172.17.11.100/5060 to gw1:172.17.41.1/5060
SIP::Embedded media port 19968 found in SDP with session IP
172.17.51.103
SIP::Found Call-ID 30D5951D-59A111D6-8058D9FC-94FF3FCE@172.17.41.1
(47)

SIP::200 received from dmz:172.17.11.100/5060 to gw1:172.17.41.1/5060
SIP::Embedded media port 19968 found in SDP with session IP
172.17.51.103
SIP::Found Call-ID 30D5951D-59A111D6-8058D9FC-94FF3FCE@172.17.41.1
(47)

SIP::ACK received from gw1:172.17.41.1/55195 to
dmz:172.17.11.100/5060
SIP::Found Call-ID 30D5951D-59A111D6-8058D9FC-94FF3FCE@172.17.41.1
(47)

```

Figure 13-16 complements the message flow documented in Example 13-12, highlighting that call setup between IOS-GW and CUCM is virtually identical to call setup for communication between two IP phones registered on the same CUCM (refer to Example 13-9).

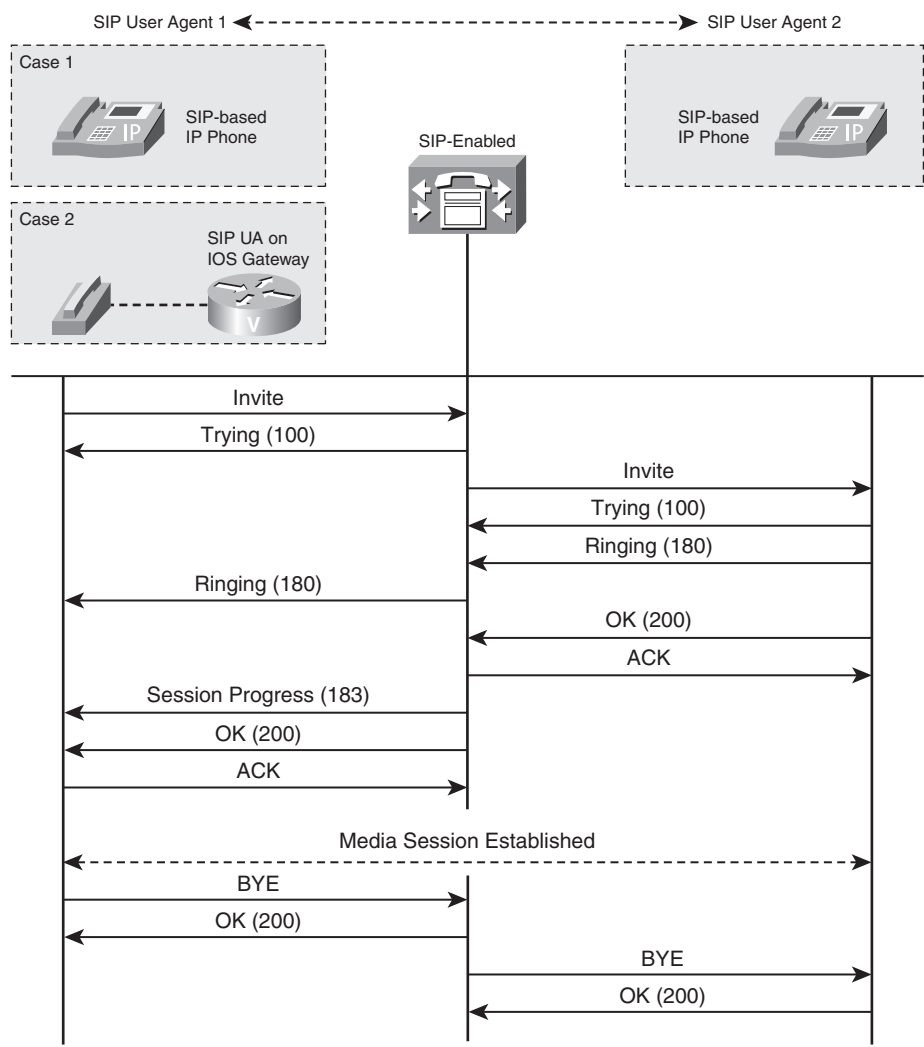


Figure 13-16 SIP Call Setup

MGCP Protocol

MGCP is a call signaling protocol that follows a different approach from H.323 and SIP. Instead of using a distributed model for call control, it defines a master/slave relationship between a Media Gateway Controller (Call Agent) and media gateways (which handle media translation between IP-based and legacy systems).

CUCM supports the Call Agent function (therefore concentrating the call control intelligence) and can control media gateways (such as Cisco IOS voice gateways) via MGCP. The MGCP specification makes it possible to remotely control individual voice ports used by gateways for connecting to the PSTN or POTS phones, for instance.

Two appealing features of MGCP follow:

- **Easier configuration:** Almost all the gateway configuration is downloaded from CUCM. Example 13-13 emphasizes this by showing the two commands needed to instruct the gateway to obtain its configuration from CUCM (via TFTP).
- **Simpler administration:** Call intelligence is confined to the Call Agent, meaning that the whole dial plan may be defined on a single place (CUCM). This almost eliminates the requirement of creating dial-peers on the voice gateways.

Before it can start being controlled by the Call Agent, a media gateway must successfully complete the registration process, which, by default, happens over UDP port 2427. The critical parameters to be provided on CUCM side (before a registration attempt) follow:

- **Gateway model:** This is simply the Cisco IOS router model.
- **Gateway hostname:** Can be the hostname alone or a combination of hostname and domain name, in case DNS is enabled. The gateway identifies itself to the CUCM as *Gateway_ID@domain_name* and its IP address gets to be known by CUCM upon registration.
- **Type of Voice Module:** Corresponds to the physical model of voice module being controlled and the numbered gateway slot where it resides.
- **Subslot information:** Defines the type of Voice Interface Card (VIC) installed (FXO, FXS, etc) on the voice module. Actually, you need only to tell CUCM about the voice ports it should control, meaning that there may be unmanaged voice ports on the gateway (from the Call Agent's standpoint).

Example 13-13 shows summary information for a registration session from a gateway named GW1 that wants to download its configuration from the Call Agent (**ccm-manager**) at 172.17.11.100, as represented in Figure 13-17. The available MGCP endpoints (source/destination of media streams) on gateway GW1 are the following Analog Access Line Endpoints (AALN):

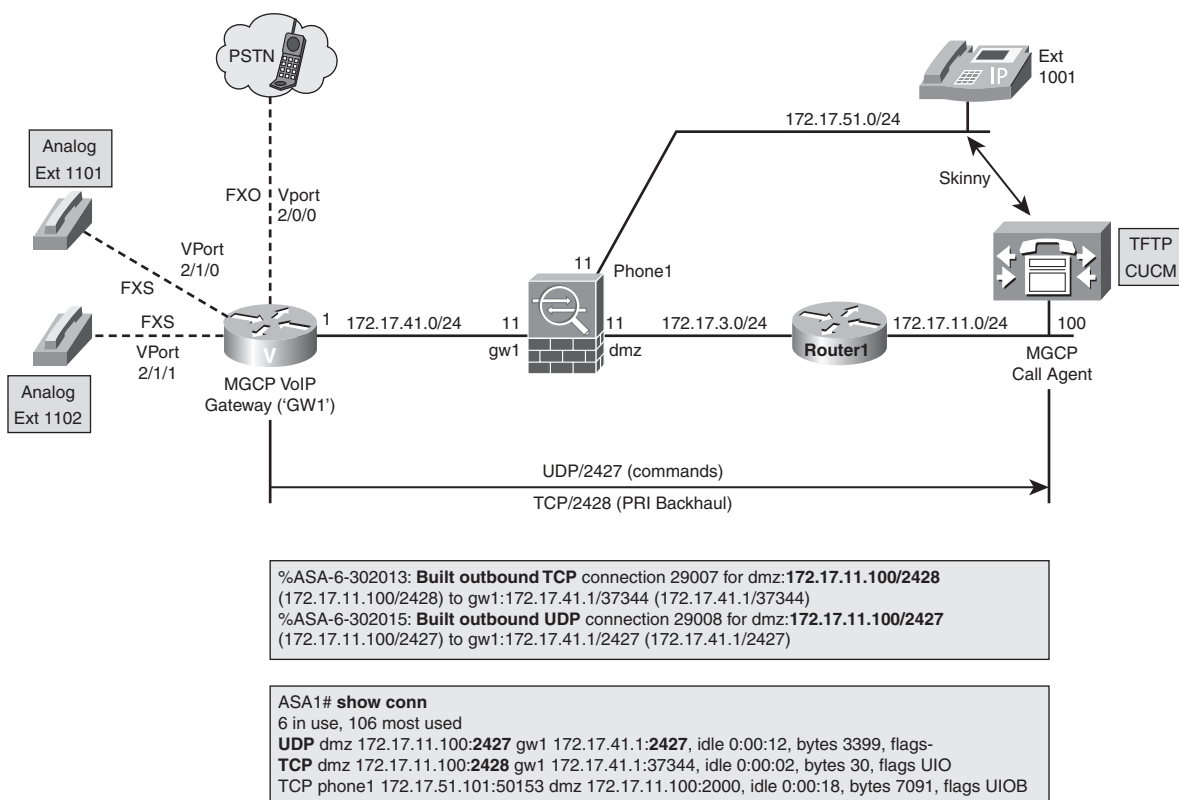


Figure 13-17 Reference Topology for MGCP Analysis

- **AALN/S2/SU0/0** An FXO port on Slot 2, Subslot 0 (Subunit), and port 0. This MGCP notation corresponds to voice-port 2/0/0 on gateway GW1.
- **AALN/S2/SU1/0** An FXS port on Slot 2, Subslot 1, and port 0 (port 2/1/0).
- **AALN/S2/SU1/1** An FXS port on Slot 2, Subslot 1, and port 1 (port 2/1/1).

The IOS **debug** commands used in Example 13-13 reveal that CUCM is configuring a series of **mgcp** global commands on the gateway and associating a **dial-peer** number with each AALN endpoint.

Example 13-14 teaches how to verify registration information on the voice gateway.

Example 13-13 MGCP Gateway Registration

```
! Instructing IOS gateway to get configuration from Call Agent (CUCM)
GW1(config)# ccm-manager config server 172.17.11.100
GW1(config)# ccm-manager config

cmapp_xml_tftp_download_file line 90: File
```

```

(tftp://172.17.11.100/GW1.cnf.xml) read 0 bytes
xmlmem = 6626B57C malloc, F:cmapp_xml_tftp_download_file L:94,
S:12666
[ output suppressed ]
cmapp_xml_update_startup_config:
cmapp_xml_open_config_file:
cmapp_xml_configure_router
cmapp_xml_reorder_interfaces:
Building configuration...
[ output suppressed ]
  cmapp_xml_cfg_router: configuring [mgcp call-agent CUCM-8032427
service mgcp version 0.1]
  cmapp_xml_cfg_router: configuring [mgcp dtmf-relay voip codec all mode out-of-
band]
  cmapp_xml_cfg_router: configuring [mgcp package-capability rtp-package]
  cmapp_xml_cfg_router: configuring [no mgcp default-package mt-package]
  cmapp_xml_cfg_router: configuring [no mgcp package-capability res-package]
  cmapp_xml_cfg_router: configuring [mgcp package-capability sst-package]
  cmapp_xml_cfg_router: configuring [mgcp package-capability pre-package]
  cmapp_xml_cfg_router: configuring [no mgcp package-capability fxr]
  cmapp_xml_cfg_router: configuring [mgcp modem passthrough voip mode nse]
  cmapp_xml_cfg_router: configuring [ ]
  cmapp_xml_cfg_router: configuring [mgcp rtp payload-type g726r16 static]
  cmapp_xml_cfg_router: configuring [mgcp rtp unreachable timeout 1000 action
notify]
  cmapp_xml_cfg_router: configuring [no mgcp timer receive-rtcp]
  cmapp_xml_cfg_router: configuring [mgcp sdp simple]
[ output suppressed ]
cmapp_xml_get_interface_info: slot = 2 vic = 0 port = 0
Interface_name: AALN/S2/SU0/0
controller:
addr: 2/0/0
interface:
voice_port: voice-port 2/0/0
dial_peer: dial-peer voice 999200
dial_peer_port: 2/0/0
[ output suppressed ]
cmapp_xml_get_interface_info: slot = 2 vic = 1 port = 0
Interface_name: AALN/S2/SU1/0
controller:
addr: 2/1/0
interface:
voice_port: voice-port 2/1/0
dial_peer: dial-peer voice 999210
dial_peer_port: 2/1/0

```

[output suppressed]

```
cmapp_xml_get_interface_info: slot = 2 vic = 1 port = 1
Interface_name: AALN/S2/SU1/1
controller:
addr: 2/1/1
interface:
voice_port: voice-port 2/1/1
dial_peer: dial-peer voice 999211
dial_peer_port: 2/1/1
```

Example 13-14 Verifying MGCP Gateway Registration

```
GW1# show ccm-manager
MGCP Domain Name: GW1
Priority          Status          Host
=====
Primary          Registered          CUCM-803 (172.17.11.100)
First Backup     None
Second Backup    None
Current active Call Manager: 172.17.11.100
Backhaul/Redundant link port: 2428
Failover Interval: 30 seconds
Keepalive Interval: 15 seconds
Last keepalive sent: 17:43:49 BRT Nov 27 2010 (elapsed time: 00:00:06)
Last MGCP traffic time: 17:43:49 BRT Nov 27 2010 (elapsed time: 00:00:06)
[ output suppressed ]
Configuration Auto-Download Information
=====
Current version-id: 1290717733-a9dbb23e-80a8-4b21-a474-bf437ab3882b
Last config-downloaded:00:00:00
Current state: Waiting for commands
Configuration Download Statistics:
Download Attempted          : 1
Download Successful         : 1
Download Failed              : 0
Configuration Attempted     : 1
Configuration Successful     : 1
Configuration Failed(Parsing): 0
Configuration Failed(config) : 0
Last config download command: New Registration
FAX mode: cisco
Configuration Error History:
!
GW1# show mgcp
```

```
MGCP Admin State ACTIVE, Oper State ACTIVE - Cause Code NONE
MGCP call-agent: CUCM-803 2427 Initial protocol service is MGCP 0.1
MGCP validate call-agent source-ipaddr DISABLED
MGCP validate domain name DISABLED
[ output suppressed ]
```

Example 13-15 refers to the network of Figure 13-17 and illustrates ASA's perspective on a call between two MGCP endpoints (FXS ports, in this case) on gateway GW1. At this point, it is important to note that MGCP inspection is not enabled by default on ASA. To accomplish that, you can, for instance, issue the **inspect mgcp** command under the class *inspection_default* in the **policy-map** called *global_policy*. This procedure was studied in Chapter 12.

Example 13-15 MGCP Sample Call (1)

```
! 1101 (voice-port 2/1/0) calls 1102 (voice-port 2/1/1)
ASA1# show mgcp sessions detail
2 in use, 2 most used
Session active 0:00:55
    Gateway IP      172.17.41.1
    Call ID         A0000000018c8776000000F5
    Connection ID   C
    Endpoint name   AALN/S2/SU1/1
    Media lcl port  17312
    Media rmt IP    172.17.41.1
    Media rmt port  17662
Session active 0:00:55
    Gateway IP      172.17.41.1
    Call ID         A0000000018c8775000000F5
    Connection ID   B
    Endpoint name   AALN/S2/SU1/0
    Media lcl port  17662
    Media rmt IP    172.17.41.1
    Media rmt port  17312
!
! Gateway perspective

GW1# show voice call status
CallID      CID  ccVdb      Port      DSP/Ch  Called #  Codec      Dial-peers
0x15       120D 0x671FC158 2/1/0     2/3:1   *         g711ulaw   999210/0
0x17       11E4 0x67AA1D0C 2/1/1     2/4:1   *         g711ulaw   0/0
2 active calls found
!
GW1# show voip rtp connections detail
```



```
VoIP RTP active connections :
No. CallId      dstCallId LocalRTP RmtRTP LocalIP RemoteIP
1 22           21           17662 17312 172.17.41.1 172.17.41.1
  callId 22 (dir=2): called= calling= redirect=
    dest callId 21: called= calling= redirect=
      1 context 66338030 xmitFunc 61769C04
2 24           23           17312 17662 172.17.41.1 172.17.41.1
  callId 24 (dir=2): called= calling= redirect=
    dest callId 23: called= calling= redirect=
      1 context 6807B4B4 xmitFunc 61769C04
Found 2 active RTP connections
```

Example 13-16 documents ASA's visibility of a call from a Skinny-based IP Phone (1001) to an MGCP endpoint (FXS port) on GW1. The example highlights the derivation of RTP channels from MGCP *ModifyConnection* (MDCX) commands. It also shows that ASA understands the *CreateConnection* commands. It is important to highlight the fact the CUCM controls the gateway via MGCP and the IP Phone via SCCP.

Example 13-16 MGCP Sample Call (2)

```
! 1001 (172.17.51.101) calls 1101 (analog phone on MGCP Gateway
172.17.41.1)
%ASA-6-608001:Pre-allocate Skinny RTP secondary channel for
gw1:172.17.41.1/18298 to phone1:0.0.0.0 from
*****StationStartMediaTransmissionID message
%ASA-6-608001:Pre-allocate Skinny RTCP secondary channel for
gw1:172.17.41.1/18299 to phone1:0.0.0.0 from
*****StationStartMediaTransmissionID message
%ASA-6-608001:Pre-allocate Skinny RTP secondary channel for
phone1:172.17.51.101/24574 to gw1:172.17.41.1 from
*****StationOpenReceiveChannelAckID message
%ASA-6-608001:Pre-allocate Skinny RTCP secondary channel for
phone1:172.17.51.101/24575 to gw1:172.17.41.1 from
*****StationOpenReceiveChannelAckID message
%ASA-6-608001:Pre-allocate Skinny RTP secondary channel for
phone1:172.17.51.101/24574 to dmz:172.17.11.100 from
*****StationOpenReceiveChannelAckID message
%ASA-6-608001:Pre-allocate Skinny RTCP secondary channel for
phone1:172.17.51.101/24575 to dmz:172.17.11.100 from
*****StationOpenReceiveChannelAckID message

%ASA-6-616001:Pre-allocate MGCP RTP connection for
phone1:172.17.51.101 to gw1:172.17.41.1/18298 from MDCX message
%ASA-6-616001:Pre-allocate MGCP RTCP connection for
```

```

phone1:172.17.51.101 to gw1:172.17.41.1/18299 from MDCX message
%ASA-6-616001:Pre-allocate MGCP RTP connection for gw1:172.17.41.1 to
phone1:172.17.51.101/24574 from MDCX message
%ASA-6-616001:Pre-allocate MGCP RTCP connection for gw1:172.17.41.1 to
phone1:172.17.51.101/24575 from MDCX message

%ASA-6-302015: Built outbound UDP connection 29134 for
phone1:172.17.51.101/24574 (172.17.51.101/24574) to
gw1:172.17.41.1/18298 (172.17.41.1/18298)
%ASA-6-302015: Built outbound UDP connection 29135 for
phone1:172.17.51.101/24575 (172.17.51.101/24575) to gw1:172.17.41.1/18299
(172.17.41.1/18299)
!
ASA1# show mgcp commands
2 in use, 25 most used
MDCX, gateway IP: 172.17.41.1, transaction ID: 461, idle: 0:00:11
CRCX, gateway IP: 172.17.41.1, transaction ID: 460, idle: 0:00:14
!
ASA1# show mgcp sessions detail
1 in use, 1 most used
Session active 0:00:23
    Gateway IP      172.17.41.1
    Call ID         A00000000018c87720000000F5
    Connection ID   A
    Endpoint name   AALN/S2/SU1/0
    Media lcl port  18298
    Media rmt IP    172.17.51.101
    Media rmt port  24574
!
ASA1# show skinny

```

	LOCAL	FOREIGN	STATE
1	172.17.11.100/2000	172.17.51.101/50153	2
AUDIO	172.17.41.1/18298	172.17.51.101/24574	

MGCP construction is based on a set of text-based *commands* and *responses* that flow over UDP Port 2427 between the Call Agent and the media gateway. The main types of MGCP commands are briefly described here:

- **AuditEndpoint (AUEP):** Sent by the Call Agent to determine the status or capabilities of an endpoint.
- **AuditConnection (AUCX):** Issued by the Call Agent to obtain the parameters of a given connection.

- **RestartInProgress (RSIP):** Used by the gateway to inform that an endpoint (or group of endpoints) have changed status (*in-service* or *out-of-service*).
- **EndpointConfiguration (EPCF):** Informs the gateway about configuration that should be applied to an endpoint.
- **NotificationRequest (RQNT):** Instructs the Gateway to watch for events such as on-hook/off-hook or to provide signals like *busy tone* and *dial tone* to endpoints.
- **Notify (NTFY):** The gateway informs the Call Agent about requested events.
- **CreateConnection (CRCX):** Creates a connection between two endpoints and typically include information such as QoS settings, selected codec, and call bandwidth.
- **ModifyConnection (MDCX):** Alters parameters related to an established connection.
- **DeleteConnection (DLCX):** Sent by the Call Agent to terminate an existent connection. When issued by the Gateway, it means that a connection can be no longer maintained.

This section ends by registering a typical MGCP call flow, which includes some of the commands just described, in Figure 13-18.

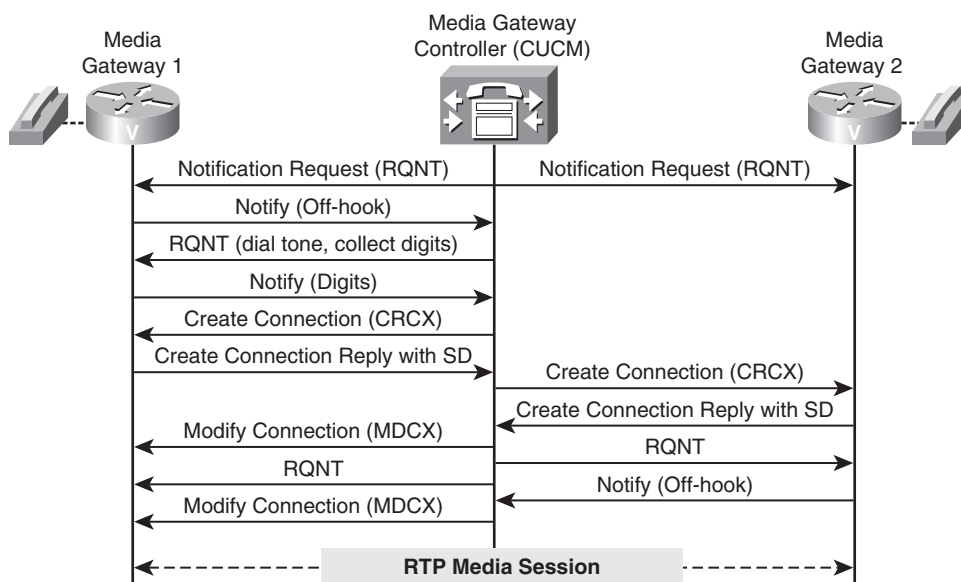


Figure 13-18 Typical MGCP Call flow

Cisco IP Phones and Digital Certificates

Cisco IP phones need to be provisioned with digital certificates before they can use any encryption feature (either for secure call signaling or media stream confidentiality).

This section quickly reviews some Public Key Infrastructure (PKI) concepts associated with CUCM and its IP Phones, which will be later required on topics related to ASA advanced voice inspection resources (like the TLS-Proxy and the Phone-Proxy).

Figure 13-19 summarizes the main Certification Authorities (CA) acting as PKI roots on a typical CUCM 8.x deployment:

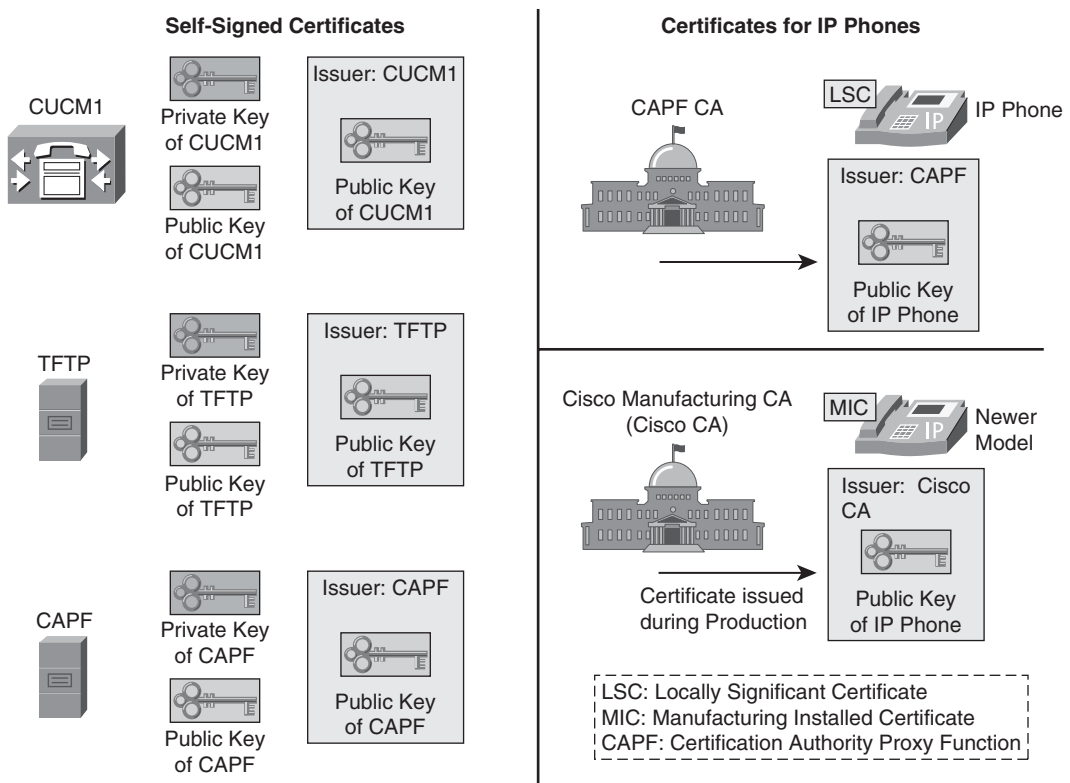


Figure 13-19 Overview of Certification Authorities Used on CUCM Deployments

- CUCM and the TFTP service have a self-signed certificate each. CUCM's is basically needed for secure call signaling whereas TFTP's is used for signing configuration files.
- The Certificate Authority Proxy Function (CAPF) is an internal service on CUCM that issues certificates for IP Phones (*Locally Significant Certificates*, [LSC]) or acts as a proxy to obtain an LSC from another CA. The CAPF service has its own self-

signed certificate. The CAPF (or an external CA to which CAPF enrolls) plays the role of root CA for all phone LSCs.

- Some newer Cisco IP Phone models (such as 7975 and 7965) support Manufactured Installed Certificates (MIC), which are issued by *Cisco Manufacturing CA* during production time. This Cisco CA acts as the root for all MICs. Cisco phone models like the 7960 or 7940 do not support MICs and must be provided with a LSC.

The presence of MICs on modern phones facilitates PKI deployments for Cisco IP Phones, because CUCM inherently trusts *Cisco Manufacturing CA*. In the event that an IP phone is simultaneously provisioned with a MIC and a LSC, the latter takes precedence.

Figure 13-20 illustrates the creation of the Certificate Trust List (CTL) file by the Cisco CTL Client application. The CTL file is downloaded by the phone when it first connects to the TFTP server during the boot process. This file contains the trusted certificate issuers (PKI roots) on a CUCM environment along with their certificates (and respective public keys). Some additional information about the CTL Client deserves specific mention:

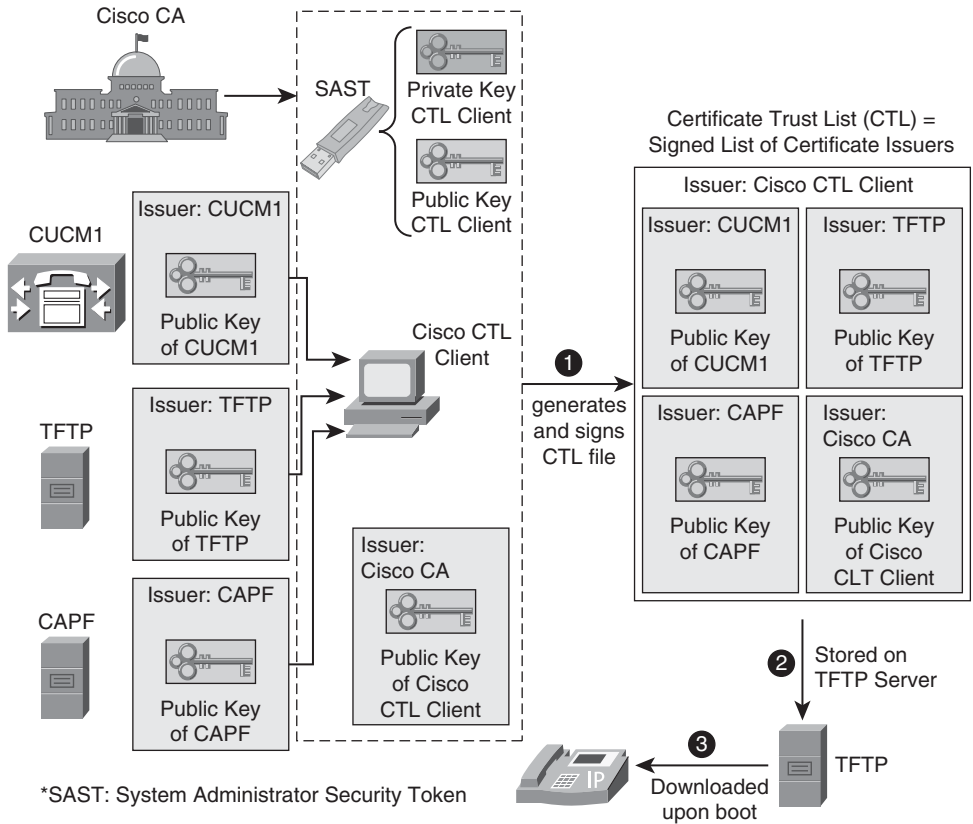


Figure 13-20 Cisco CTL Client and the CTL File

- The CTL file is signed by Cisco CTL Client using the private key from one of the SAST (*System Administrator Security Tokens*), which are all signed by the Cisco CA (Cisco Manufacturing CA). The public and private key pair of the CTL Client is physically stored on the SAST tokens.
- The CTL file also plays the role of an authorization list that specifies the function of each server certificate (CUCM, TFTP, and so on).
- Every time an IP Phone receives a new CTL file, it is verified. A new CTL must be signed by one of the SAST tokens that are listed in the IP phone's current CTL file. For the special situations of initial deployment or after erasing the CTL from the IP phone, (so that no CTL is available on the phone), the new CTL is not verified.

Once downloaded to IP Phones, the CTL file plays a critical role on establishing many trust relationships:

- IP Phones use the public-key from the CUCM self-signed certificate to authenticate the CUCM on *encrypted signaling* scenarios.
- IP Phones use the public-key within the TFTP self-signed certificate to validate *signed configuration files*. (These files are signed with TFTP's private key.)
- IP Phones employ the public-key of the CAPF server (issuer of LSCs) to validate the CAPF during LSC enrollment process (which is protected by TLS). Notice that the CTL file must be on the phone before an LSC can be obtained.

Figure 13-21 portrays some tasks related to TLS exchanges between IP phones and the CUCM to which they are registered:

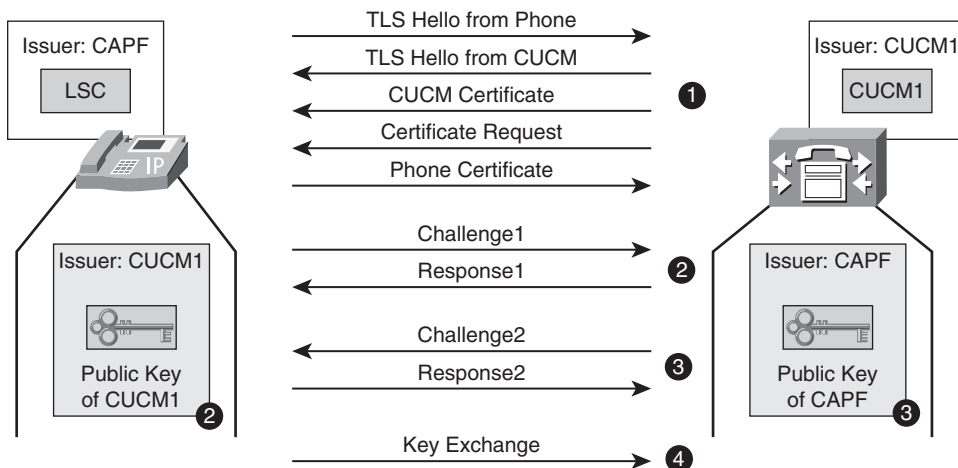


Figure 13-21 Certificate Validation Between IP Phones and CUCM

1. TLS Hellos are employed to negotiate TLS session attributes (one-way or two-way certificate exchange, encryption and HMAC algorithms, key lengths, and so on). After that, the IP phone and the CUCM exchange their certificates.
2. The IP phone authenticates the CUCM by requesting it to sign a challenge (random data) with its private-key. This server-side validation depends on phone's possession of CUCM's public-key (obtained from CUCM's certificate contained in the CTL file).
3. CUCM authenticates the IP phone by asking it to sign a challenge with its private-key.
4. The IP phone generates session keys for AES and SHA-1 and encrypts them using CUCM's public key. CUCM employs its private-key to decrypt the session keys. Now the phone and the CUCM share secret keys.

Note In Figure 13-21 the number '2' on the IP Phone side indicates that the public-key from CUCM1's certificate is used by the phone to authenticate CUCM1 on Step 2.

Note When an IP Phone registers, its certificate is stored on a *dynamic trust list* on CUCM.

Advanced Voice Inspection with ASA TLS-Proxy

Many Cisco IP Phone models have supported secure signaling and encrypted media streams for quite a while. This, on one hand, covers an important aspect of voice security but, on the other hand, render firewalls unaware of the call signaling process.

The study conducted so far, characterized the process of opening only the necessary media channels, after inspecting the signaling protocol messages. (This was possible for all the protocols covered within this chapter.) If encrypted signaling is employed, the protocol messages become meaningless to firewall inspection engines because they will see only a static port reserved for signaling (such as TCP/2443, the SCCP over TLS port) and treat it as a generic, single-channel, TCP-based protocol. This ends up meaning that all the RTP and RTCP media ports would need to be open in advance. Well, but maintaining a large range of UDP ports open would virtually make your firewall useless. *Wouldn't it?*

However, if from the end user's standpoint, the most relevant feature is voice confidentiality, a *simple solution* would be using unencrypted signaling so that the connections involving Secure RTP (SRTP) ports could be dynamically created across firewall interfaces. *Well, not that simple actually....* You should never lose sight of the fact that the keys used for media encryption are generated during the signaling process, and as a result, in the cases where signaling is not encrypted, these session keys would be exposed.

After this quick reflection about the impact of enabling encryption of voice traffic on the work of firewalls, a basic question naturally arises: Is it feasible at all to enable stateful inspection and encryption simultaneously on IP Telephony environments? (*Yes, indeed....*)

This is made possible by the use of advanced features such as the ASA TLS-Proxy and Phone-Proxy, which will be analyzed going forward.

Example 13-17 provides information about the scenario portrayed in Figure 13-22. Phones have been configured on CUCM to use *encrypted mode*, causing them to register using the Skinny over TLS port (TCP/2443). The permissions for this type of communication were provisioned in Example 13-1 and in its companion topology of Figure 13-2. Even though ASA sees a session between each IP phone and CUCM on port 2443, this port has no special meaning. It might be instructive to compare the current output of `show skinny` with those of previous examples.

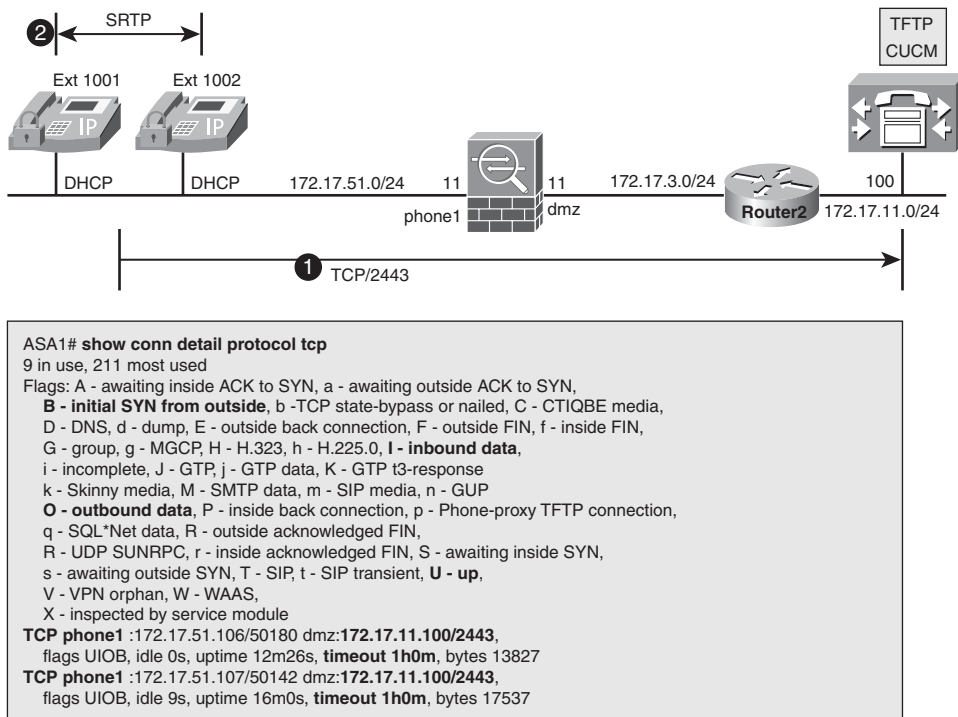


Figure 13-22 IP Phones in Secure Mode - Scenario 1

In the particular network of Figure 13-22 the problem was minimized because the phones were on the same subnet (on the same ASA interface). This enabled them to receive the SRTP ports from CUCM and establish the correspondent secure connection (without crossing ASA). This is not the case for the scenario depicted in Figure 13-23, in which the phones reside on distinct ASA interfaces. They still receive CUCM instructions for the SRTP ports to be open but cannot establish the connection through ASA (which, at least in a firewall book, would not be wide open...).

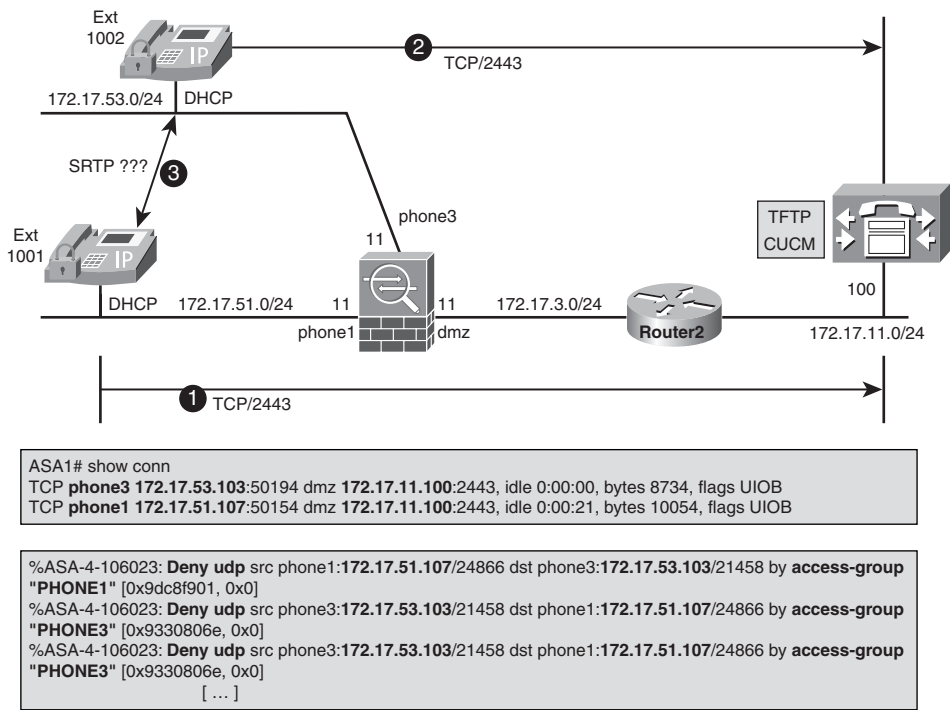


Figure 13-23 IP Phones in Secure Mode - Scenario 2

Example 13-17 ASA Is Not Skinny-Aware for Phones in Secure Mode

```
! IP Phones connect to CUCM on port TCP/2443 (reserved for Skinny
over TLS)
ASA1# show conn detail protocol tcp
[ output suppressed ]
TCP phone1 :172.17.51.106/50180 dmz:172.17.11.100/2443,
    flags UIOB, idle 0s, uptime 12m26s, timeout 1h0m, bytes 13827
TCP phone1 :172.17.51.107/50142 dmz:172.17.11.100/2443,
    flags UIOB, idle 9s, uptime 16m0s, timeout 1h0m, bytes 17537
!
! TCP 2443 has no special meaning for ASA.
ASA1# show skinny
LOCAL FOREIGN STATE
-----
ASA1#
```

Figure 13-24 organizes the configuration elements for the ASA TLS-Proxy solution, an ingenious way of inserting ASA appliances in encrypted signaling scenarios and recovering protocol visibility. This renewed awareness is further explored in Example 13-18.

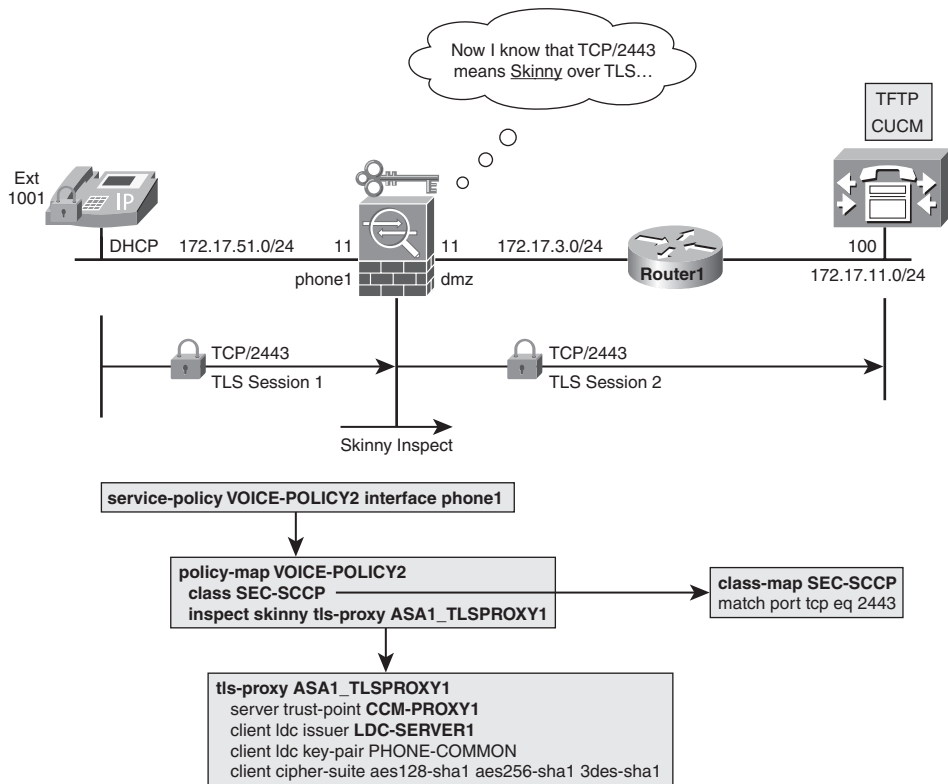


Figure 13-24 Recovering Skinny-Awareness with ASA TLS-Proxy

Note A completely analogous policy (using the same TLS-Proxy instance) has been applied to interface 'phone3' (172.17.53.0/24), in which another IP phone resides (as shown in Figure 13-23). For the sake of simplicity, this was not represented in Figure 13-24.

Example 13-18 Recovering Skinny-Awareness with ASA TLS-Proxy

```

! ASA now knows that TCP/2443 corresponds to Skinny
ASA1# show skinny
LOCAL                                FOREIGN                                STATE
-----
1      172.17.11.100/2443             172.17.53.101/51131                 0
!
ASA1# show tls-proxy session
1 in use (1 established), 5 most used
phone3 172.17.53.101:51131 dmz 172.17.11.100:2443
P:0xd95fa698(ASA1_TLSPROXY1) S:0xd960d8f8 byte 8580

```

```

!
! Details about the TLS-Proxy session

ASA1# show tls-proxy session detail
1 in use (1 established), 5 most used
phone3 172.17.53.101:51131 dmz 172.17.11.100:2443
P:0xd95fa698(ASA1_TLSPROXY1) S:0xd960d8f8 byte 9636
  Client: State SSLOK Cipher AES128-SHA Ch 0xd5a93cc0 TxQSize 0
LastTxLeft 0 Flags 0x31
  Server: State SSLOK Cipher AES128-SHA Ch 0xd5a93ca0 TxQSize 0
LastTxLeft 0 Flags 0x9
Local Dynamic Certificate
  Status: Available
  Certificate Serial Number: 2b
  Certificate Usage: General Purpose
  Public Key Type: RSA (1024 bits)
  Issuer Name:
    hostname=ldc-asa1.uc.lab.bsa
    cn=TLSPROXY-ASA1
  Subject Name:
    cn=SEP000750D54236
    ou=cisco
    o=cisco
    c=US
  Validity Date:
    start date: 08:07:06 BRT Nov 3 2010
    end   date: 08:07:06 BRT Nov 3 2011
  Associated Trustpoints:

```

Example 13-19 concentrates on certificate validation activities for the ASA TLS-Proxy scenario of Figure 13-24. ASA validates the phone (client side) and CUCM (server side), acting as proxy between these two TLS-enabled entities. It is instructive to compare these Syslog messages with the original analysis of Figure 13-21, always keeping in mind that ASA logs are simply highlighting the authentication of the entities that are talking to ASA (either the client or the server). The logs do not show ASA being authenticated.

Example 13-19 Certificate Validation on TLS-Proxy Environment

```

! First client-side validation (ASA validates IP Phone certificate,
the UC client)
%ASA-6-725001: Starting SSL handshake with client
phone3:172.17.53.101/50190 for TLSv1 session.
%ASA-7-725010: Device supports the following 4 cipher(s).
%ASA-7-725011: Cipher[1] : RC4-SHA

```

```

%ASA-7-725011: Cipher[2] : AES128-SHA
%ASA-7-725011: Cipher[3] : AES256-SHA
%ASA-7-725011: Cipher[4] : DES-CBC3-SHA
%ASA-7-725008: SSL client phone3:172.17.53.101/50190 proposes the
following 2
cipher(s).
%ASA-7-725011: Cipher[1] : AES256-SHA
%ASA-7-725011: Cipher[2] : AES128-SHA
%ASA-7-725012: Device chooses cipher : AES128-SHA for the SSL session
with client phone3:172.17.53.101/50190
%ASA-7-717025: Validating certificate chain containing 1
certificate(s).
%ASA-7-717029: Identified client certificate within certificate
chain. serial number:
03, subject name: cn=SEP000750D54236,ou=cisco,o=cisco,c=US.
%ASA-7-717030: Found a suitable trustpoint CAPF_Trustpoint to
validate certificate.
%ASA-6-717022: Certificate was successfully validated. serial number:
03, subject name: cn=SEP000750D54236,ou=cisco,o=cisco,c=US.
%ASA-6-717028: Certificate chain was successfully validated with
warning, revocation status was not checked.
%ASA-6-725002: Device completed SSL handshake with client
phone3:172.17.53.101/50190

! Second client-side validation (ASA validates CUCM certificate, the UC server)

%ASA-6-725001: Starting SSL handshake with server
dmz:172.17.11.100/2443 for TLSv1 session.
%ASA-7-725009: Device proposes the following 3 cipher(s) to server
dmz:172.17.11.100/2443
%ASA-7-725011: Cipher[1] : AES128-SHA
%ASA-7-725011: Cipher[2] : AES256-SHA
%ASA-7-725011: Cipher[3] : DES-CBC3-SHA
%ASA-7-725013: SSL Server dmz:172.17.11.100/2443 choose cipher :
AES128-SHA
%ASA-7-717025: Validating certificate chain containing 1 certificate(s).
%ASA-7-717029: Identified client certificate within certificate
chain. serial number: 1A9ADE4B6B746725, subject name:
c=US,st=cisco,l=cisco,o=cisco,ou=cisco, cn=CUCM-
803.uc.lab.bsa.
%ASA-6-717022: Certificate was successfully validated. Certificate is
resident and trusted, serial number: 1A9ADE4B6B746725, subject name:
c=US,st=cisco,l=cisco,o=cisco,ou=cisco,cn=CUCM-803.uc.lab.bsa.
%ASA-6-717028: Certificate chain was successfully validated with

```

```
revocation status check.
```

```
%ASA-6-725002: Device completed SSL handshake with server
```

```
dmz:172.17.11.100/2443
```

This section ends by examining the necessary trust relationships in the TLS-Proxy model. The sequence of operations, associated with the TLS authentications in this environment, is shown in Figure 13-25 and described as follows:

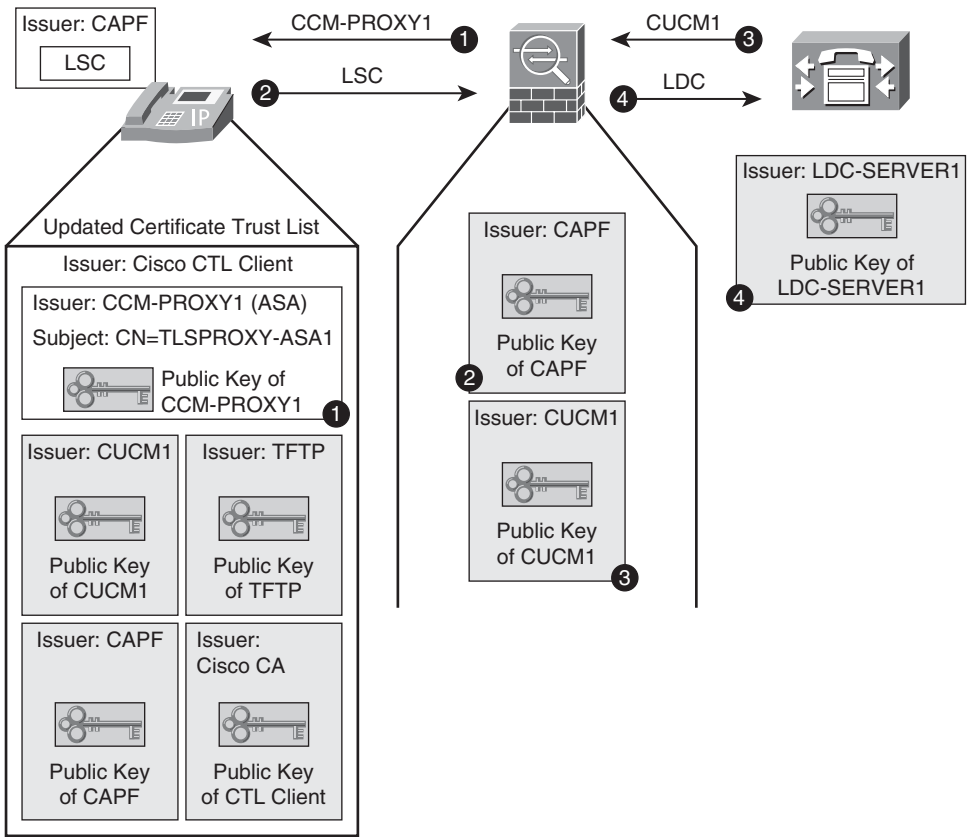


Figure 13-25 Trust Relationships in the ASA TLS-Proxy Environment

1. ASA appliance presents a certificate to the IP phone on behalf of CUCM. This digital certificate is generated by ASA and is sent to the phone within an updated CTL file (The creation and distribution of the original CTL file were discussed on the section "Cisco IP Phones and Digital Certificates.") The appropriate public-key for validating this ASA certificate is obtained from the new CTL file.
2. Assuming the use of LSCs, ASA needs to trust the CAPF Service (the PKI root for all LSCs), so that it can verify the authenticity of the LSC presented by the phone.

(ASA's administrator therefore needs to create a CAPF **trustpoint**.) Just to emphasize this point, notice that in the first client-side validation of Example 13-19, the CAPF **trustpoint** was used by ASA to validate the phone certificate (LSC).

3. Another **trustpoint** is created on ASA so that it can authenticate CUCM1. (This is achieved by installing CUCM1's certificate on ASA.)
4. A special Certificate Authority is created on ASA to generate the phone certificates known as *Local Dynamic Certificates (LDC)*. ASA appliance passes a unique LDC to CUCM on behalf of each phone. Because these LDCs are presented to CUCM for validation, CUCM needs to trust the CA that signed them. The certificate of the internal ASA CA that is the PKI root for LDCs (*LDC-SERVER1*) is then uploaded to CUCM's certificate store. A sample LDC is unveiled by the **show tls-proxy session detail** command, as illustrated in Example 13-18.

This final discussion is summarized on Figure 13-25. In this figure, each sign containing a number 'X' (X =1, 2, 3, 4) has two meanings:

- Order of TLS handshaking operations.
- Public-key used by authenticator to validate the response of the challenged entity: In Step 3, for example, CUCM1 signs the challenge sent by ASA using the private-key of CUCM1 PKI root. ASA uses the public-key from the CUCM1 certificate to verify CUCM1's response.

Note Correlating the information in Figures 13-24 and 13-25 with Examples 13-18 and 13-19 might be an instructive exercise to consolidate the PKI concepts discussed so far.

Advanced Voice Inspection with ASA Phone-Proxy

The previous section analyzed the ASA TLS-Proxy solution, a mechanism whose purpose is providing stateful inspection in conjunction with encrypted voice calls. It is critical to keep in mind that this feature applies only to voice calls in which both parties make use of encryption.

Another interesting solution is the ASA Phone-Proxy, a resource that takes advantage of the native encryption capability in Cisco IP Phones to provide secure calls from *external* phones. The term "external" here means a phone that is requesting access from the outside of the Organization (even from the Internet). This usage scenario is represented in Figure 13-26.

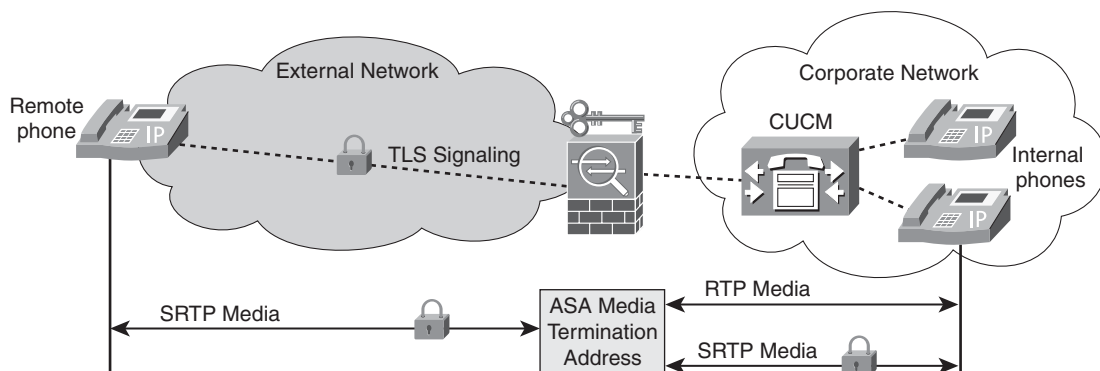


Figure 13-26 *Securely Integrating Remote Phones with ASA Phone-Proxy*

ASA Phone-Proxy can also lend itself to solve the *Secure VLAN Traversal* challenge, on corporate environments that contain softphones as part of an UC deployment. This is portrayed in Figure 13-27 and discussed as follows:

- Considering that the softphones reside on PCs belonging to a Data VLAN, and need to establish RTP (or SRTP) connections with IP phones on a Voice VLAN, how can traffic be filtered between these two domains? After all, any access control rule created between these two domains could be exploited by other applications on the PC where the softphone is installed to compromise the Voice VLAN.
- ASA Phone-Proxy offers a way of encrypting calls from Cisco IP Communicator (CIPC) 7.x softphones or, at least authenticating earlier releases of CIPC, before allowing them to traverse the firewall. This is done for the CIPC application and not for the IP address of the hosting PC. The authentication provided by Phone-Proxy is also useful for protecting CUCM from rogue softphones.

Example 13-20 and Figure 13-28 are used in tandem to summarize ASA's Phone-Proxy configuration. Some relevant aspects of this setup are listed here:

- The TFTP, CUCM, and DNS server addresses must be translated by ASA using static NAT. In the scenario of Figure 13-28, the address of CUCM/TFTP is seen on the outside (interface 'out1') as 172.17.131.100, whereas the DNS server is reached using the 172.17.131.103 address. In the example, the address of the TFTP server is informed to the phone by means of the DHCP Option #150.
- Assuming the use of an ASA pre-8.3 release, TFTP and DNS must be permitted to the appropriate translated addresses, as shown by ACL named OUT1. The interactions between NAT and ACLs is different if you use 8.3 or later code, as explained in Appendix A, "NAT and ACL Changes in ASA 8.3."
- You must ensure that all TFTP traffic between the remote IP phone and the TFTP server flows through ASA. As characterized in the examples, TFTP inspection is critical for the solution to work correctly.

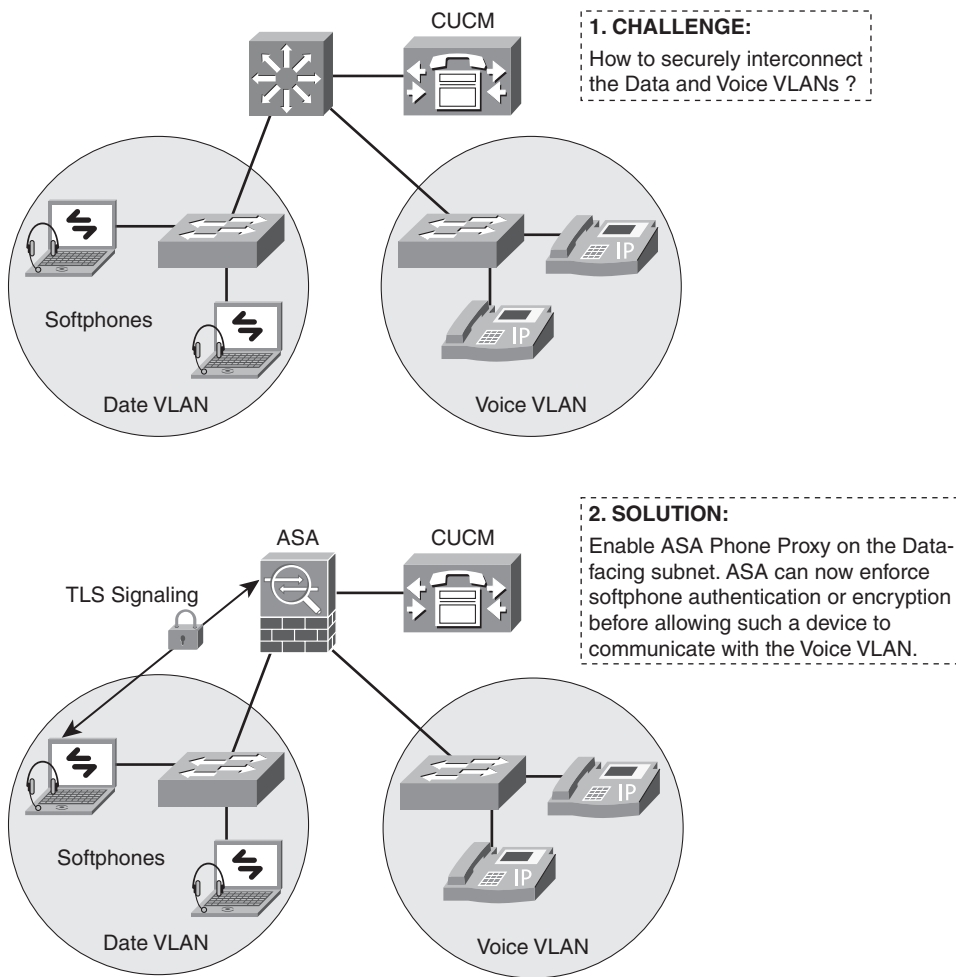


Figure 13-27 *Secure VLAN Traversal Challenge*

- DNS resolution must guarantee that the reply sent to the external phone contains the translated address of the server. In the scenario under analysis, this was achieved by enabling *DNS Doctoring*, a technique studied in Chapter 12.
- A new CTL file containing entries for the CAPF and CUCM+TFTP entities should be created and delivered to phones. As discussed in the section “Cisco IP Phones and Digital Certificates,” the original CTL file needs to be deleted before this new one can be installed. More details about the certificates included in this updated CTL file are assembled in Example 13-21.
- For **mixed mode** CUCM clusters (those that enable a mix of phones operating in *nonsecure* and *secure* modes), there might be IP Phones already configured as

encrypted. Such a scenario would require ASA to act as TLS-Proxy. This possibility is taken care of by using the **tls-proxy** command inside the **phone-proxy** instance.

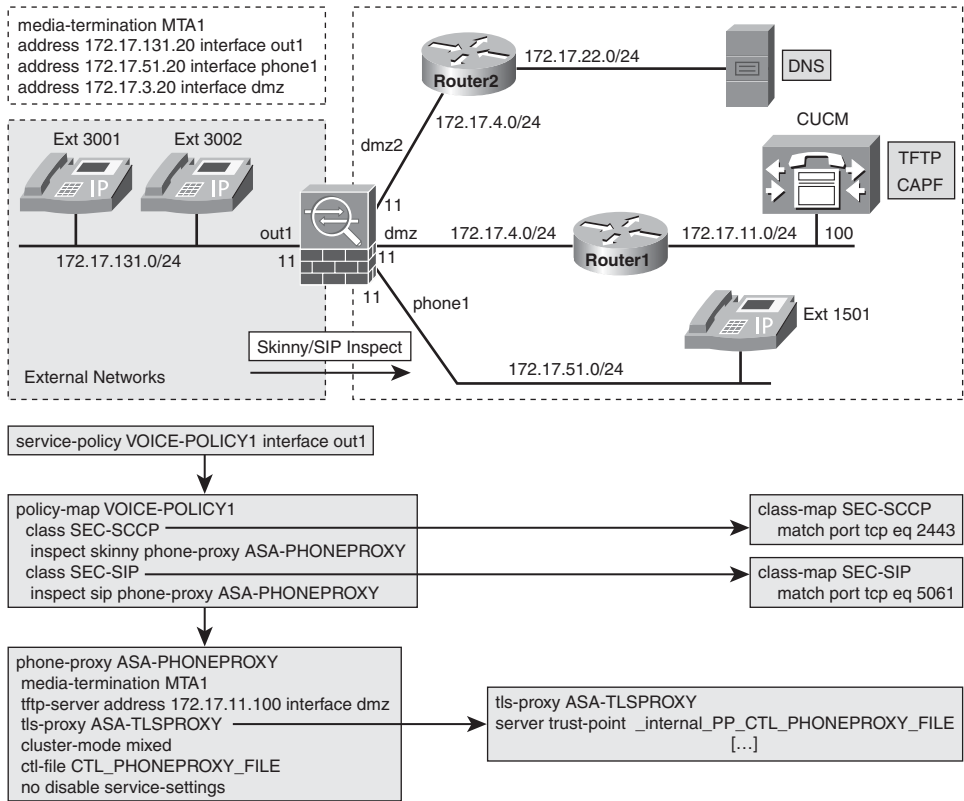


Figure 13-28 Reference Topology for ASA Phone-Proxy Analysis

- Phone-Proxy uses a Media Termination Address (MTA) for RTP/SRTP sessions. This address can be configured either globally or on a per-interface basis. (This was the choice on the network of Figure 13-28.) Some guidelines for selecting an interface-specific MTA follow:
 - Each MTA address must be routable and unique.
 - Each MTA address must belong to the network range assigned to that interface.
 - The MTA address must be different from ASA's interface address.
- The real address of the TFTP server must be specified in the **phone-proxy** instance.

Example 13-20 Auxiliary Configs for ASA Phone-Proxy Scenario

```
! ASA acts as DHCP server on external interface 'out1'
```

```

dhcpd domain uc.lab.bsa
dhcpd address 172.17.131.201-172.17.131.210 out1
dhcpd dns 172.17.131.103 interface out1
dhcpd lease 86400 interface out1
dhcpd option 150 ip 172.17.131.100 interface out1
dhcpd enable out1
!
! Translated address of the DNS server is 172.17.131.103
static (dmz2,out1) 172.17.131.103 172.17.22.103 netmask
255.255.255.255
!
! Translated address of CUCM (which also hosts TFTP and CAPF
services)
static (dmz,out1) 172.17.131.100 172.17.11.100 netmask
255.255.255.255 dns
!
! Service and network object-groups and ACL to be applied to interface 'out1'

object-group service OUT1-SVCS1
  service-object udp eq tftp
!
object-group network PHONEPROXY-NETS
  network-object 172.17.131.0 255.255.255.0
!
object-group network TFTP1
  network-object host 172.17.131.100
!
object-group network DNS1
  network-object host 172.17.131.103
!
access-list OUT1 extended permit object-group OUT1-SVCS1 object-group
PHONEPROXY-NETS object-group TFTP1
access-list OUT1 extended permit udp object-group PHONEPROXY-NETS
object-group DNS1 eq domain
!
! The CTLfile generated by ASA

ctl-file CTL_PHONEPROXY_FILE
  record-entry capf trustpoint CAPF_Trustpoint address 172.17.131.100
  record-entry cucm-tftp trustpoint PHONEPROXY-TRUSTPOINT address
172.17.131.100
  no shutdown

```

Example 13-21 *More Information About the ASA CTL-File for Phone-Proxy*

```

ASA1# show ctl-file CTL_PHONEPROXY_FILE
Total Number of Records: 5
CTL Record Number 1
Subject Name:
  c=US,st=cisco,l=cisco,o=cisco,ou=cisco,cn=CAPF-2f2377a0
Issuer Name:
  c=US,st=cisco,l=cisco,o=cisco,ou=cisco,cn=CAPF-2f2377a0
Function:
  capf
IP Address:
  172.17.131.100
Associated Trustpoint:
  CAPF_Trustpoint

CTL Record Number 2
Subject Name:
  hostname=ASA1.uc.lab.bsa
Issuer Name:
  hostname=ASA1.uc.lab.bsa
Function:
  cucm-tftp
IP Address:
  172.17.131.100
Associated Trustpoint:
  PHONEPROXY-TRUSTPOINT

CTL Record Number 3
Subject Name:
  cn=_internal_CTL_PHONEPROXY_FILE_SAST_0,ou=STG,o=Cisco Inc
Issuer Name:
  cn=_internal_CTL_PHONEPROXY_FILE_SAST_0,ou=STG,o=Cisco Inc
Function:
  sast
IP Address:
  0.0.0.0
Associated Trustpoint:
  _internal_CTL_PHONEPROXY_FILE_SAST_0

CTL Record Number 4
Subject Name:
  cn=_internal_CTL_PHONEPROXY_FILE_SAST_1,ou=STG,o=Cisco Inc
Issuer Name:
  cn=_internal_CTL_PHONEPROXY_FILE_SAST_1,ou=STG,o=Cisco Inc

```

```

Function:
  sast
IP Address:
  0.0.0.0
Associated Trustpoint:
  _internal_CTL_PHONEPROXY_FILE_SAST_1

CTL Record Number 5
Subject Name:
  cn=_internal_PP_CTL_PHONEPROXY_FILE,ou=STG,o=Cisco Inc
Issuer Name:
  cn=_internal_PP_CTL_PHONEPROXY_FILE,ou=STG,o=Cisco Inc
Function:
  cucm-tftp
IP Address:
  172.17.130.11
Associated Trustpoint:
  _internal_PP_CTL_PHONEPROXY_FILE
ASA1#

```

Example 13-22 documents some important information about phone registration on an ASA Phone-Proxy deployment. The main steps pertaining to this registration process are described here:

1. ASA sends the local CTL file to the remote phone.
2. After receiving the complete configuration file from the TFTP server, ASA starts the file modification process:
 - It rewrites the address of the CUCM to the translated address.
 - It creates an ACL for permitting the phone to talk with the secure port on CUCM (TCP/2443) and with the CAPF Service (TCP/3804).
 - It applies an *Application Redirect Rule* if the phone is configured in non-secure mode.
 - It sets the phone mode to *Encrypted*.
3. Upon receipt of the modified configuration file, the phone attempts a connection to CUCM's external address on the secure signaling port (2443 for SCCP over TLS and 5061 for SIP over TLS).
4. ASA keeps acting as a TFTP-proxy for other files that need to be downloaded to the phone.
5. After registration, the phone is ready to place calls. A successful registration may be verified with **show phone-proxy secure-phones**.

Example 13-22 *Phone Registration on CUCM with ASA Phone-Proxy in Action*

```

PP: Remote phone at 172.17.131.201/52554 requesting
CTLSEP9CAFAFE28C9.tlv from 172.17.11.100/69
PP: Created entry for secure device out1:172.17.131.201, MAC
9caf.cafe.28c9, current count 2, list count 1
PP: ASA sending local CTL file on disk CTL_PHONEPROXY_FILE to remote
phone, opened 0x1d942a
PP: Data Block 1 sent from ASA at 172.17.131.100/5641 to remote phone
at 172.17.131.201/52554, ingress ifc
PP: Received ACK for Block 1 from remote phone
out1:172.17.131.201/52554 to ASA at dmz:172.17.11.100
[ output suppressed ]
PP: Received ACK for Block 9 from remote phone
out1:172.17.131.201/52554 to ASA at dmz:172.17.11.100
PP: CTL file TFTP session complete, CTL file has been successfully
sent to the remote phone.

PP: Remote phone at 172.17.131.201/52104 requesting
ITLSEP9CAFAFE28C9.tlv from 172.17.11.100/69
PP: ASA sent request for ITLSEP9CAFAFE28C9.tlv sourced from
out1:172.17.131.201 to dmz:172.17.11.100, opened 0x1eb636
PP: Remote phone at 172.17.131.201/50073 requesting
SEP9CAFAFE28C9.cnf.xml.sgn from 172.17.11.100/69
[ output suppressed ]
PP: Intercepted Data Block 16 from dmz:172.17.11.100/59897 to remote
phone out1:172.17.131.201/50073
Intercepted Block 16
PP: ASA sending TFTP Ack for Block #16 from 172.17.131.201/50073 to
TFTP server 172.17.11.100/59897
[ output suppressed ]

PP: Complete configuration file received from Call Manager TFTP
server, beginning config file modification process.
PP: Performing DNS lookup on hostname CUCM-803
PP: Hostname resolves to 172.17.11.100
PP: Added ACL for 172.17.131.201 to out1:172.17.131.100/2443
PP: Applied Application Redirect rule for 172.17.131.201/2000 to
172.17.131.100, secure port 2443
PP: Added ACL for 172.17.131.201 to out1:172.17.131.100/3804
PP: Config Modify : detected Call Manager 172.17.11.100, rewriting to
172.17.131.100
PP: Config Modify : deviceSecurityMode set to 1 (unencrypted), modifying
deviceSecurityMode to 3 (encrypted).
PP: Config Modify : Modifying to TLS as the transport layer protocol.
PP: Encrypting and signing modified config file using trustpoint
<_internal_PP_CTL_PHONEPROXY_FILE>.

```

```

PP: Config file modification complete, forwarding modified config
file to remote phone.
PP: Data Block 1 sent from ASA at 172.17.131.100/59897 to remote
phone at 172.17.131.201/50073
[ output suppressed ]
PP: Received ACK for Block 16 from remote phone
out1:172.17.131.201/50073 to ASA at dmz:172.17.11.100
PP: Config file TFTP session complete, config file has been
successfully sent to the remote phone.

PP: Remote phone at 172.17.131.201/50894 requesting tzupdater.jar.sgn
from 172.17.11.100/69
PP: The remote phone is requesting file tzupdater.jar.sgn.
PP: ASA is requesting file tzupdater.jar.sgn from Call Manager TFTP
server.
PP: ASA sent request for tzupdater.jar.sgn sourced from
out1:172.17.131.201 to dmz:172.17.11.100, opened 0x2345a2
PP: Intercepted Data Block 1 from dmz:172.17.11.100/59897 to remote
phone out1:172.17.131.201/50894
[ output suppressed ]
PP: ASA sending TFTP Ack for Block #456 from 172.17.131.201/50894 to
TFTP server 172.17.11.100/59897
!
ASA1# show phone-proxy secure-phones
ASA-PHONEPROXY: 1 in use, 1 most used

```

	Interface	IP Address	Port	MAC	Timeout	Idle
	out1	172.17.131.201	50903	9caf.cafe.28c9	0:05:00	0:00:16

Tip the output of `show phone-proxy secure-phones` command displays a value of 'zero' for the *Port* parameter, it means that there is something wrong with the IP Phone's registration on CUCM.

Example 13-23 illustrates a call between two remote phones on the ASA Phone-Proxy scenario of Figure 13-28. Some noteworthy aspects of this deployment follow:

- Although the external phones are always in *secure-mode*, ASA is Skinny-aware. This is a result of acting as a proxy for TLS Signaling.
- The sessions with the Media Termination Addresses are shown in the `debug` and `show` commands.

Example 13-24 refers to Figure 13-28 and documents a call between an external phone and a *nonsecure* internal phone. The `show phone-proxy media sessions` reveals a RTP media session between the internal phone and one of the MTA addresses.

Example 13-23 *Call Between External Phones on a Phone-Proxy Environment*

```

! Initial situation ( external phones already registered)
ASA1# show phone-proxy secure-phones
ASA-PHONEPROXY: 2 in use, 2 most used

```

Interface	IP Address	Port	MAC	Timeout	Idle
out1	172.17.131.202	52816	9caf.cafe.12d7	0:05:00	0:00:09
out1	172.17.131.201	49264	9caf.cafe.28c9	0:05:00	0:00:14

```

!
! Extension 3001 (172.17.131.201) calls 3002 (172.17.131.202)

%ASA-6-608001: Pre-allocate Skinny RTP secondary channel for
out1:172.17.131.201/20910 to dmz:172.17.11.100 from
*****StationOpenReceiveChannelAckID message
%ASA-6-608001: Pre-allocate Skinny RTCP secondary channel for
out1:172.17.131.201/20911 to dmz:172.17.11.100 from
*****StationOpenReceiveChannelAckID message
%ASA-6-608001: Pre-allocate Skinny RTP secondary channel for NP
Identity Ifc:172.17.3.20/20910 to out1:0.0.0.0 from
*****StationStartMediaTransmissionID message
!
! What the debugs reveal
Inspect Skinny: generating SRTP key
PP_SIG:: Setting rmt SRTP params in pp_cl d796c488
PP_SIG::master_key and salt cbe722deec89a0e2817a971c6e487541
cbe722deec89a0e2817a971c6e487541
[ output suppressed ]
Creating new media session in SNP 172.17.3.20/20910
PP_SIG:: Established an SRTP media session between phone at 172.17.131.201/19020
and
local media termination address 172.17.3.20/20910.
PP_SIG::Updating lcl PP media info 172.17.3.20/20910
Setting remote_side SRTP params in SNP 172.17.3.20/20910
[ output suppressed ]
Creating new media session in SNP 172.17.3.20/20226
PP_SIG:: Established an SRTP media session between phone at 172.17.131.202/25746
and
local media termination address 172.17.3.20/20226.
PP_SIG::Updating lcl PP media info 172.17.3.20/20226
[ output suppressed ]
SNP_SRTP::Setting client_port to sport 25746
PP_SIG:: Secure call between phone at 172.17.131.202/25746 and phone at
172.17.131.201/19020 is now active.
PP_SIG::updating timestamp for Phone 172.17.131.201/49264
PP_SIG::updating timestamp for Phone 172.17.131.202/52816
!

```

```
! Information about Phone-Proxy sessions
```

```
ASA1# show phone-proxy signaling-sessions
```

```
out1 172.17.131.202:52816 dmz 172.17.11.100:2000
```

```
Local Media (audio) conn: 172.17.131.202/25746 to 172.17.131.20/20910
```

```
Local SRTP key set : Remote SRTP key set
```

```
Remote Media (audio) conn: 172.17.3.20/20910 to 172.17.3.20/20226
```

```
out1 172.17.131.201:49264 dmz 172.17.11.100:2000
```

```
Local Media (audio) conn: 172.17.131.201/19020 to 172.17.131.20/20226
```

```
Local SRTP key set : Remote SRTP key set
```

```
Remote Media (audio) conn: 172.17.3.20/20226 to 172.17.3.20/20910
```

```
!
```

```
ASA1# show phone-proxy media-sessions
```

```
2 in use, 2 most used
```

```
Media-session: 172.17.131.20/20226 :: client ip 172.17.131.202/25746
```

```
Lcl SRTP conn 172.17.131.20/20226 to 172.17.131.201/19020 tx_pkts 7063 rx_pkts 7065
```

```
Media-session: 172.17.131.20/20910 :: client ip 172.17.131.201/19020
```

```
Lcl SRTP conn 172.17.131.20/20910 to 172.17.131.202/25746 tx_pkts 7063 rx_pkts 7063
```

Example 13-24 Call to Internal Phone on a Phone-Proxy Scenario

```
! External phone at 172.17.131.210 calls internal phone at 172.17.51.101
```

```
ASA1# show phone-proxy signaling-sessions
```

```
out1 172.17.131.210:52369 dmz 172.17.11.100:2000
```

```
Local Media (audio) conn: 172.17.131.210/30360 to 172.17.131.20/24748
```

```
Local SRTP key set : Remote SRTP key set
```

```
Remote Media (audio) conn: 172.17.51.101/19098 to 172.17.131.20/26446
```

```
!
```

```
ASA1# show phone-proxy media-sessions
```

```
2 in use, 3 most used
```

```
Media-session: 172.17.131.20/24748 :: client ip 172.17.51.101/19098
```

```
Lcl SRTP conn 172.17.131.20/24748 to 172.17.131.210/30360 tx_pkts 3484 rx_pkts 3485
```

```
Media-session: 172.17.131.20/26446 :: client ip 172.17.131.210/30360
```

```
Lcl RTP conn 172.17.131.20/26446 to 172.17.51.101/19098 tx_pkts 3485 rx_pkts 3485
```

```
!
```

```
ASA1# show skinny
```

	LOCAL	FOREIGN	STATE
1	172.17.11.100/2000	172.17.131.210/52369	2
	AUDIO 172.17.51.101/24748	172.17.131.210/26446	
2	172.17.11.100/2000	172.17.51.101/50138	2
	AUDIO 172.17.131.20/26446	172.17.51.101/19098	

Examples 13-25 and 13-26 use the topology of Figure 13-28 to explore VLAN Traversal concepts for Cisco IP Communicator (CIPC) softphones:

- Example 13-25 covers the case in which a CIPC 7.x is set to Encrypted Mode (no change to the configuration of the **phone-proxy** instance is necessary) and the CIPC behavior is virtually identical to that of the IP phone.
- Example 13-26 covers the case of a CIPC version that does not support encryption. The relevant changes on ASA configuration are shown and CIPC establishes an RTP media session with one of the MTA addresses. (It is interesting to contrast the output of **show phone-proxy media sessions** with that in Example 13-25).

Example 13-25 ASA Phone-Proxy and Softphones (1)

```
! Encrypted Mode: Cisco IP Communicator (CIPC) behaves exactly as an IP Phone
PP: Added ACL for 172.17.131.203 to out1:172.17.131.100/2443
PP: Applied Application Redirect rule for 172.17.131.203/2000 to 172.17.131.100,
secure port 2443
PP: Added ACL for 172.17.131.203 to out1:172.17.131.100/3804
PP: Config Modify : detected Call Manager 172.17.11.100, rewriting to
172.17.131.100
PP: Config Modify : deviceSecurityMode set to 1 (unencrypted), modifying
deviceSecurityMode to 3 (encrypted).
PP: Config Modify : Modifying to TLS as the transport layer protocol.
%ASA-4-106023: Deny tcp src out1:172.17.131.203/2820 dst dmz:172.17.131.100/6970
by access-group "OUT1" [0x0, 0x0]
PP: Encrypting and signing modified config file using trustpoint
<_internal_PP_CTL_PHONEPROXY_FILE>.
PP: Config file modification complete, forwarding modified config file to remote
phone.
!
ASA1# show phone-proxy secure-phones
ASA-PHONEPROXY: 2 in use, 2 most used

      Interface      IP Address  Port MAC           Timeout Idle
      out1  172.17.131.203  2822 001c.2520.4107 0:05:00 0:00:08
      out1  172.17.131.204  51228 9caf.cafe.28c9 0:05:00 0:00:23
!
ASA1# show phone-proxy media-sessions
2 in use, 2 most used
Media-session: 172.17.131.20/28020 :: client ip 172.17.131.204/25258
Lc1 SRTP conn 172.17.131.20/28020 to 172.17.131.203/24578 tx_pkts 13386
rx_pkts 13387
Media-session: 172.17.131.20/19936 :: client ip 172.17.131.203/24578
Lc1 SRTP conn 172.17.131.20/19936 to 172.17.131.204/25258 tx_pkts 13387
rx_pkts 13387
!
ASA1# show skinny
```

	LOCAL	FOREIGN	STATE
1	172.17.11.100/2000	172.17.131.203/2822	2
AUDIO	172.17.3.20/28020	172.17.131.203/19936	
2	172.17.11.100/2000	172.17.131.204/51228	2
AUDIO	172.17.3.20/19936	172.17.131.204/28020	

Example 13-26 ASA Phone-Proxy and Softphones (2)

```

! Authenticated Mode: adding the 'null-sha1' option for SSL Encryption
ssl encryption rc4-sha1 aes128-sha1 aes256-sha1 3des-sha1 null-sha1
!
! Instructing ASA Phone-Proxy to require only authentication from CIPC softphones
phone-proxy ASA-PHONEPROXY
  cipc security-mode authenticated
!
! Registration Process (CIPC Security Mode is changed to 'authenticated')

PP: Added ACL for 172.17.131.203 to out1:172.17.131.100/2443
PP: Applied Application Redirect rule for 172.17.131.203/2000 to 172.17.131.100,
secure port 2443
PP: Added ACL for 172.17.131.203 to out1:172.17.131.100/3804
PP: Config Modify : detected Call Manager 172.17.11.100, rewriting to
172.17.131.100
PP: Config Modify : deviceSecurityMode set to 1 (unencrypted), modifying
deviceSecurityMode to 2 (authenticated).
PP: Config Modify : Modifying to TLS as the transport layer protocol.
PP: Encrypting and signing modified config file using trustpoint
<_internal_PP_CTL_PHONEPROXY_FILE>.
!
ASA1# show phone-proxy secure-phones
ASA-PHONEPROXY: 2 in use, 2 most used

      Interface      IP Address  Port MAC           Timeout Idle
      out1    172.17.131.203  2853 001c.2520.4107 0:05:00 0:00:21
      out1    172.17.131.204  51228 9caf.cafe.28c9 0:05:00 0:00:19

! CIPC (172.17.131.203) calls IP phone on 172.17.131.204

ASA1# show phone-proxy media-sessions
2 in use, 2 most used
Media-session: 172.17.131.20/19720 :: client ip 172.17.131.203/24576
  Lcl SRTP conn 172.17.131.20/19720 to 172.17.131.204/29480 tx_pkts 1299 rx_pkts
1300
Media-session: 172.17.131.20/29168 :: client ip 172.17.131.204/29480
  Lcl RTP conn 172.17.131.20/29168 to 172.17.131.203/24576 tx_pkts 1300 rx_pkts
1299

```

Note At the time this book was written, only the Cisco IP Communicator softphone was supported by the ASA Phone-Proxy solution. Other softphone products such as the Cisco Unified Personal Communicator (CUPC) were on the roadmap. If you need these products on your UC project, it is advisable that you contact your Cisco sales representatives to confirm availability.

Note The CIPC softphones used in Examples 13-25 and 13-26 were connected to interface 'out1' in the topology of Figure 13-28 (just like the remote IP phones). This was done just to take advantage of the configurations already in place. In a real-life scenario, the softphones would be installed on an internal Data VLAN.

Summary

A good understanding of IP Telephony (IPT) protocols behavior is critical for successful implementation of Unified Communication (UC) solutions.

This chapter analyzes the main IPT signaling protocols through the eyes of Cisco ASA inspection engines. The theory of operations of SCCP, H.323, SIP, and MGCP are largely exemplified.

The chapter also covers ASA's advanced inspection mechanisms known as TLS-Proxy and Phone-Proxy, which make the coexistence of stateful firewall with voice encryption a possibility.

Further Reading

Cisco Voice Gateways and Gatekeepers (Denise Donohue, David Mallory, Ken Salhoff)

<http://www.ciscopress.com/bookstore/product.asp?isbn=158705258X>

SIP Trunking (Christina Hattingh, Darryl Sladden, ATM Zakaria Swapan)

<http://www.ciscopress.com/bookstore/product.asp?isbn=1587059444>

Identity on Cisco Firewalls

This chapter covers the following topics:

- Selecting the Authentication Protocol
- ASA User-level control with Cut-Through Proxy
- IOS User-level control with Auth-Proxy
- User-based Zone Policy Firewall
- Administrative Access Control on IOS
- Administrative Access Control on ASA

“One may understand the cosmos, but never the ego; the self is more distant than any star.”—G. K. Chesterton

The IP address is a basic piece of information associated with computer systems that rely on the TCP/IP protocol stack to establish network connectivity. In this book, you studied in detail how to define access control rules based upon attributes present in the IP, TCP, UDP, and application headers. Although this protocol-based approach has demonstrated to be powerful, the demand for networks that are flexible enough to accommodate multiple classes of users (employees, contractors, and guests) and the increasing need for network mobility have motivated the search for alternative forms of implementing access rules.

This chapter focuses on the creation of rules that can include not only the IP addresses of source and destination systems interconnected by Cisco Firewalls, but also *Identity* information related to the users initiating the service requests. You need to understand the key terms in the list that follows:

- **Identity:** A descriptor that refers to a given subject (user, computer, and process) in an individualized manner. In the context of computer networks, the most typical descriptor for users is the *username*. (The *username* is not secret information.)

- **Authentication:** Process that can verify an alleged Identity by comparing the credentials presented by the subject with predefined ones. Username/password is the most widespread identity/credential pair and is used quite often in this chapter.
- **Authorization:** Post-authentication process that can differentiate access rights that will be assigned to authenticated users. The distinction among users is typically materialized by placing them in different user groups in a central authentication database (*User DB*) and defining attributes that reflect the tasks that members of the group are allowed to perform.
- **Accounting:** Process by which the network device (accounting client) that enforces the access policy, collects user activity information, and sends it to the accounting server.
- **AAA:** Reference architecture designed to help on building secure and scalable access control to network resources. AAA stands for Authentication, Authorization, and Accounting and defines a modular way for these three security functions to interact. AAA uses a client-server model and promotes the centralized handling of users.
- **Authentication Proxy Protocols:** Set of protocols used for communication between AAA clients and the centralized *User DB*. The two prominent members of this set, RADIUS and TACACS+, deserve special attention in this chapter. These protocols are hereafter simply referred to as *authentication protocols*.
- **Network Access Server (NAS):** Term frequently used, within the context of the AAA architecture to refer to a network device that acts as an AAA client. Some examples of AAA clients are routers, LAN switches, firewalls, VPN concentrators, and Wireless Access Points. In this book, the NASes are ASA-based firewalls and routers supporting the Cisco IOS Firewall feature set.
- **AAA Server:** Implementation of the server portion of an Authentication protocol. In this book, this functionality resides in the Cisco Security Access Control Server (CS-ACS) product for both RADIUS and TACACS+.
- **Regular users:** Represents users who request the establishment of connections *through* the firewalls. This category typically is composed of the great majority of users.
- **Admin users:** Represents users who need administrative access *to* the AAA client devices (firewalls). It normally includes a limited number of users who are in charge of configuration, monitoring, and troubleshooting tasks.

Now that you have a high-level overview of these fundamental concepts, some of them are explored in more detail.

The first important requirement for authentication to succeed is the existence of the username in the User DB. Some designs consider the definition of less restrictive policies to deal with the special case of *unknown* or *undefined* users but, rigorously speaking, such a process would not be considered “true” authentication.

The credentials that a known user presents to the NAS during the authentication phase are used to validate its Identity, which, in turn, serves as a search key to locate the user in a particular group from which it inherits the policy attributes. This capability to establish

a distinction between access profiles is what makes Identity-based security policies valuable. The reasoning is similar to what is done in the Quality of Service (QoS) domain: If traffic from any user receives the highest priority, it is exactly the same effect of receiving the lowest, and there will be no point in spending time on building a QoS policy.

The typical AAA sequence of activities can be summarized as follows:

1. A user (regular or admin) requests a connection *through* or *to* a Cisco Firewall device that requires one or more of the AAA services before being established.
2. The firewall demands user credentials by means of a username/password prompts.
3. User presents the credentials to the firewall (NAS).
4. The firewall sends the authentication request (containing identity/credential) to the AAA Server.
5. The AAA Server responds to the NAS with some sort of *accept* or *reject* message, reflecting the validity of the presented user credentials.
6. Authorization is a process that might be independent of or coupled with authentication. In both options, successful authorization results in the AAA server sending back control criteria that must be locally enforced by the NAS during the whole access session. Although, by definition, authorization is considered an optional feature, as previously stated, an Identity-based service that does not include this process would be virtually meaningless
7. If configured, Accounting data flows from the NAS to the AAA Server. The server processes the Accounting information and acknowledges its receipt.

It is interesting to emphasize that the server in the AAA framework does not need to host the *User DB* and limit the authentication method to static username/password combinations. It can act as a reference point for integration with more sophisticated methods of authentication and alternative user databases. This possibility of delegating the authentication function to a more specialized entity is one of the elements that makes AAA so flexible. Such an approach is convenient because

- Someone in charge of developing NAS devices can direct efforts to the support of the selected authentication protocol instead of worrying about integrating with more elaborate authentication systems.
- Someone developing more complex authentication solutions, such as One Time Password (OTP) systems and Token servers, can focus on the integration with the AAA Server.
- Overall, it can contribute to cost reduction and avoid lots of compatibility efforts.

Selecting the Authentication Protocol

The previous section established that firewalls acting as AAA clients rely on an authentication protocol to communicate with the AAA server and determine *User Identity*. After subjecting the user to centralized authentication, a NAS typically receives a set of access control parameters, enforces them locally, and optionally accounts for user activity. Now you face the challenge of selecting RADIUS or TACACS+ to accomplish the task of creating Identity-based security policies.

Traditional comparisons between RADIUS and TACACS+ discuss topics such as transport layer protocol used (UDP or TCP), what portions of the packets are encrypted (only password or the complete payload), and decoupling of the authentication and authorization processes. Deciding if any of the previous choices is the best path to follow is subjective and controversial. For example, RFC 2865 has a section dedicated to justify “*Why UDP?*”, whereas many customers select TACACS+ because it is carried over TCP, a protocol that was designed with reliable delivery and retransmission as basic requirements.

The approach followed in this book takes into account the classic Cisco principle of “*no technology religion*” to make the choice. The decision process is based on the suitability of the protocol to deliver the two main categories of Identity-based services:

- **RADIUS:** Considering that authorization data is embedded in the authentication response sent by the RADIUS server, this protocol is more appropriate for delivering attributes that remain valid for the duration of the user session rather than authorizing each activity individually. RADIUS is ideal for controlling access to network-based services requested by *regular users* who connect via dial-up, VPN, or Dot1X (for both wired and wireless environments).
- **TACACS+:** Has proved effective to control administrative access to network devices because *EXEC* sessions (requests for execution of commands) are interactive in nature. On a typical TACACS+ session, each attempt of an *admin user* to issue commands on a NAS device is authorized individually.

Note RADIUS and TACACS+ client functions are simultaneously available on Cisco firewall products, therefore reinforcing the approach of selecting each protocol for the category of task to which it is more suited.

Throughout this chapter, you can confirm after analyzing several examples that while TACACS+ is optimized for authorization and accounting of commands, RADIUS is flexible about the attributes it can send back to the NAS to control a noninteractive access session.

It deserves special mention the fact that RADIUS is standardized by the IETF and supports Vendor Specific Attributes (VSA), allowing an always expanding universe of network services that can be controlled.

Because the purpose of this book is to cover Cisco Firewalls, the study of RADIUS and TACACS+ is contextualized in the following way:

- **RADIUS:** Employed to control access *through* the firewall using mechanisms such as *Cut-through Proxy* in the ASA family or the correspondent *Auth-Proxy* on the IOS Firewall. After authenticating the user with one of these features, authorization ACLs can be downloaded via RADIUS to the NAS, therefore specifying traffic allowed to flow through the firewall.
- **TACACS+:** Used to authenticate potential *admin users* who request access *to* the firewall devices, individually authorizing the command execution attempts. This can be achieved in a scalable manner by creating authorization profiles known as *Shell Command Authorization Sets* on Cisco Secure ACS. A natural follow-on to command authorization is the accounting of allowed commands and registering the unauthorized attempts of issuing commands. This is valuable for configuration change control and contributes to minimize the operational risk in the network.

Note Cisco Secure ACS implements simultaneously the RADIUS and TACACS+ server functions, therefore enabling the security administrator to use this product to define access control policies for both regular and admin users.

ASA User-Level Control with Cut-Through Proxy

In many practical situations, there is a requirement to validate identity and associate the pertinent access profile before allowing user traffic to flow *through* the firewall. Cisco ASA devices can fulfill this service need by using the *Cut-Through Proxy* feature.

If configured for intercepting packets that carry HTTP, HTTPS, Telnet, or FTP, the *Cut-through Proxy* feature can leverage the authentication functionality embedded on these protocols, to present the user with a characteristic username/password prompt of the application protocol being intercepted. The user information thus obtained is forwarded to the AAA server for authentication.

If the user is successfully validated, the authentication information is cached locally and the ASA creates dynamic permissions for the application protocol under consideration. For the duration of some configurable timers (*uauth timeouts*) these permissions remain valid and there is no need to reauthenticate the user, which translates in efficiency for traffic flow.

Figure 14-1 provides an overview of Cut-Through Proxy operation using HTTP as the triggering protocol.

A simple description of the steps represented in Figure 14-1 follows:

1. The user attempts to establish an HTTP connection through the firewall.
2. Cut-Through Proxy intercepts the HTTP request and presents a browser-based authentication prompt to the user.

3. The ASA receives the user credentials and sends an *Authentication Request* message to the RADIUS server (CS-ACS).
4. The RADIUS server authenticates the user and sends an *Access-Accept* message to the ASA.
5. ASA establishes a connection between the user browser and the web server.

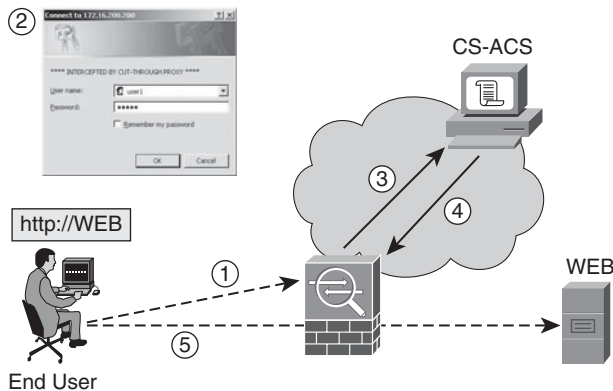


Figure 14-1 *Cut-Through Proxy Operation Using HTTP as the Triggering Protocol*

This is a brief analysis of the simplest form of operation of the *Cut-Through Proxy* mechanism. This initial discussion just considered the task of creating permissions for the intercepted protocol. But this resource can be used in many more situations to produce the interesting effect of authorizing connections associated with protocols that do not natively support authentication.

Study the instructive usage scenarios presented throughout this section. They can help unveil the power of Identity-based control.

Cut-Through Proxy Usage Scenarios

Figure 14-2 shows the base network topology for the Cut-through Proxy scenarios investigated hereafter. HTTP is the protocol triggering the authentication process.

Note All the ASA Cut-Through Proxy scenarios use RADIUS as the authentication protocol. The reasons for considering that a convenient choice were already outlined in the section “Selecting the Authentication Protocol.”

Example 14-1 is an assembly of the relevant configurations needed on ASA to implement the Cut-Through Proxy feature. Except for a command added in Example 14-3, only CS-ACS configurations vary on scenarios 1 through 4.

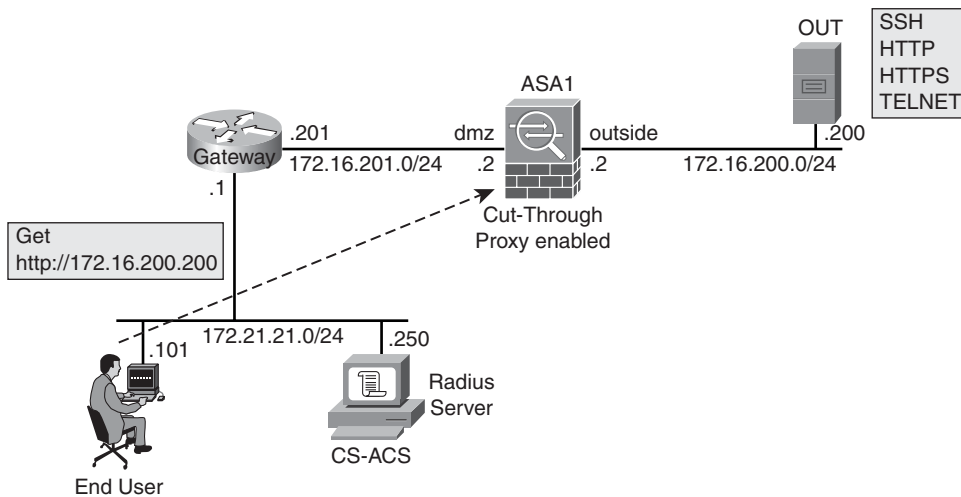


Figure 14-2 Network Topology for the Cut-Through Proxy Scenarios

Example 14-1 Baseline Configuration for Cut-Through Proxy Scenarios

```
!Defining the static ACL called "DMZ" and assigning to "dmz" interface

access-list DMZ extended permit tcp 172.21.21.0 255.255.255.0 172.16.200.0
255.255.255.0 eq www
access-list DMZ extended permit tcp 172.21.21.0 255.255.255.0 172.16.200.0
255.255.255.0 eq https
access-group DMZ in interface dmz
!
!Defining an AAA server-group called "RADIUS1"

aaa-server RADIUS1 protocol radius
aaa-server RADIUS1 (dmz) host 172.21.21.250
key cisco123
authentication-port 1812
accounting-port 1813
!
!Defining interesting traffic for authentication (using RADIUS)

access-list DMZ-AUTH1 extended permit tcp any 172.16.200.0 255.255.255.0 eq www
access-list DMZ-AUTH1 extended permit tcp any 172.16.200.0 255.255.255.0 eq https
aaa authentication match DMZ-AUTH1 dmz RADIUS1
!
! Defining interesting traffic for accounting (using RADIUS)

access-list DMZ-ACCT1 extended permit tcp any 172.16.200.0 255.255.255.0 eq www
access-list DMZ-ACCT1 extended permit tcp any 172.16.200.0 255.255.255.0 eq https
```

```

access-list DMZ-ACCT1 extended permit tcp any 172.16.200.0 255.255.255.0 eq telnet
aaa accounting match DMZ-ACCT1 dmz RADIUS1
!
!Customizing the authentication prompt presented to users
auth-prompt prompt **** INTERCEPTED BY CUT-THROUGH PROXY ****

```

Scenario 1: Simple Cut-Through Proxy (No Authorization)

Example 14-2 depicts the operation of Cut-Through Proxy for HTTP, according to the configuration in Example 14-1. Authorization is analyzed starting on Scenario 2.

Note For the examples covered in this chapter “user1” and “user2”, respectively belonging to groups “GROUP1” and “GROUP2” on CS-ACS, are always the reference usernames.

Example 14-2 HTTP Connection Is Intercepted by Cut-Through Proxy

```

! HTTP to 172.16.200.200 is intercepted by Cut-Through Proxy (first prompt
appears)
%ASA-6-302013: Built outbound TCP connection 26 for outside:172.16.200.200/80
(172.16.200.200/80) to dmz:172.21.21.101/1148 (172.21.21.101/1148)
%ASA-6-109001: Auth start for user '???' from 172.21.21.101/1148 to
172.16.200.200/80
!
! User enters credentials and ASA sends them to the RADIUS server (UDP/1812)

%ASA-6-302015: Built outbound UDP connection 27 for dmz:172.21.21.250/1812
(172.21.21.250/1812) to identity:172.16.201.2/1025 (172.16.201.2/1025)
%ASA-6-113004: AAA user authentication Successful : server = 172.21.21.250 :
user =user1
%ASA-7-734003: DAP: User user1, Addr 172.21.21.101: Session Attribute
aaa.radius["25"]["1"] = CACS:0/13e/ac10c902/4
%ASA-7-734003: DAP: User user1, Addr 172.21.21.101: Session Attribute
aaa.cisco.grouppolicy = DfltGrpPolicy
%ASA-7-734003: DAP: User user1, Addr 172.21.21.101: Session Attribute
aaa.cisco.username = user1
%ASA-6-734001: DAP: User user1, Addr 172.21.21.101, Connection Cut-Through-Proxy:
The following DAP records were selected for this connection: DfltAccessPolicy
%ASA-2-109011: Authen Session Start: user 'user1', sid 4
%ASA-6-109005: Authentication succeeded for user 'user1' from 172.21.21.101/1148
to 172.16.200.200/80 on interface dmz
! ASA starts RADIUS Accounting connection (UDP/1813)

%ASA-6-302015: Built outbound UDP connection 28 for dmz:172.21.21.250/1813
(172.21.21.250/1813) to identity:172.16.201.2/1026 (172.16.201.2/1026)
%ASA-6-113004: AAA user accounting Successful : server = 172.21.21.250 : user =
user1

```

```

!
!Displaying the authenticated users

ASA1# show uauth

                        Current      Most Seen
Authenticated Users      1          1
Authen In Progress       0          1
user 'user1' at 172.21.21.101, authenticated
    absolute timeout: 0:05:00
    inactivity timeout: 0:00:00

```

Figure 14-3 presents a sample RADIUS accounting record in CS-ACS. In this particular example, you can see in the “cisco-av-pair” column that both HTTP and HTTPS activities are registered.

RADIUS Accounting active.csv [Refresh](#) [Download](#)

Regular Expression: Start Date & Time: End Date & Time: Rows per Page:

Results from 09/13/2009,20:17:00 to 09/13/2009,20:18:50

Date	Time	Acct-Status-Type	User-Name	Group-Name	Access Device	NAS-IP-Address	Acct-Session-Id	cisco-av-pair
09/13/2009	20:18:47	Stop	user1	GROUP1	ASA1_RADIUS	172.16.201.2	003E048A	ip:source-port=1152 ip:destination-port=443 ip:source-ip=172.21.21.101 ip:destination-ip=172.16.200.200
09/13/2009	20:18:46	Start	user1	GROUP1	ASA1_RADIUS	172.16.201.2	003E048A	ip:source-port=1152 ip:destination-port=443 ip:source-ip=172.21.21.101 ip:destination-ip=172.16.200.200
09/13/2009	20:17:25	Start	user1	GROUP1	ASA1_RADIUS	172.16.201.2	003DE96A	ip:source-port=1149 ip:destination-port=80 ip:source-ip=172.21.21.101 ip:destination-ip=172.16.200.200
09/13/2009	20:17:25	Stop	user1	GROUP1	ASA1_RADIUS	172.16.201.2	003DE96A	ip:source-port=1149 ip:destination-port=80 ip:source-ip=172.21.21.101 ip:destination-ip=172.16.200.200
09/13/2009	20:17:25	Stop	user1	GROUP1	ASA1_RADIUS	172.16.201.2	00366A42	ip:source-port=1148 ip:destination-port=80 ip:source-ip=172.21.21.101 ip:destination-ip=172.16.200.200
09/13/2009	20:17:00	Start	user1	GROUP1	ASA1_RADIUS	172.16.201.2	00366A42	ip:source-port=1148 ip:destination-port=80 ip:source-ip=172.21.21.101 ip:destination-ip=172.16.200.200

Figure 14-3 Example of RADIUS Accounting Session on CS-ACS (Reports and Activity)

Scenario 2: Cut-Through Proxy with Downloadable ACEs

Example 14-3 introduces the usage of the **per-user-override** option when defining an **access-group**. This instructs the ASA to prefer any kind of dynamic authorization attributes (individual ACEs, local ACL, and Downloadable ACL) over an existent static ACL.

Example 14-4 summarizes the relevant user group parameters that need to be configured in CS-ACS so that individual ACEs can be downloaded after authentication.

Example 14-3 Instructing the ASA to Prefer Dynamic ACEs

```

!Instructing ASA to prefer downloaded Access Control Entries over static ones.

```

```
access-group DMZ in interface dmz per-user-override
```

Example 14-4 *Defining Individual ACEs on CS-ACS for Cut-Through Proxy*

```
ACS/Group Settings : GROUP1  
[009\001] cisco-av-pair  
ip:inacl#1=permit tcp any any eq 80  
ip:inacl#2=permit tcp any any eq 23
```

Example 14-5 illustrates how CS-ACS partners with ASA to deliver and enforce individual ACEs after the Cut-Through Proxy feature comes into play. Notice the Cisco-AV-pairs (ip:inacl) in the RADIUS *Response* message. Example 14-6 shows how to verify the downloaded attributes.

Example 14-5 *RADIUS Server Downloads Individual ACEs to ASA*

```
! ASA receives individual ACEs from the RADIUS server  
  
RADIUS packet decode (response)  
[output suppressed]  
Radius: Type = 26 (0x1A) Vendor-Specific  
Radius: Length = 43 (0x2B)  
Radius: Vendor ID = 9 (0x00000009)  
Radius: Type = 1 (0x01) Cisco-AV-pair  
Radius: Length = 37 (0x25)  
Radius: Value (String) =  
69 70 3a 69 6e 61 63 6c 23 31 3d 70 65 72 6d 69 | ip:inacl#1=permi  
74 20 74 63 70 20 61 6e 79 20 61 6e 79 20 65 71 | t tcp any any eq  
20 38 30 | 80  
Radius: Type = 26 (0x1A) Vendor-Specific  
Radius: Length = 43 (0x2B)  
Radius: Vendor ID = 9 (0x00000009)  
Radius: Type = 1 (0x01) Cisco-AV-pair  
Radius: Length = 37 (0x25)  
Radius: Value (String) =  
69 70 3a 69 6e 61 63 6c 23 32 3d 70 65 72 6d 69 | ip:inacl#2=permi  
74 20 74 63 70 20 61 6e 79 20 61 6e 79 20 65 71 | t tcp any any eq  
20 32 33 | 23
```

Example 14-6 *Verifying Downloaded ACEs for an Authorized User*

```
! Displaying the downloaded access-list for user1  
  
ASA1# show uauth user1
```

```

user 'user1' at 172.21.21.101, authenticated
  access-list AAA-user-user1-E3040000 (*)
    absolute timeout: 0:05:00
    inactivity timeout: 0:00:00
!
! Displaying the details of the downloaded access-list

ASA1# show access-list AAA-user-user1-E3040000
access-list AAA-user-user1-E3040000; 2 elements; name hash: 0x3dbead96 (dynamic)
access-list AAA-user-user1-E3040000 line 1 extended permit tcp any any eq www
(hitcnt=2) 0xf104c9e5
access-list AAA-user-user1-E3040000 line 2 extended permit tcp any any eq telnet
(hitcnt=0) 0x6d5c94fc

```

Example 14-7 characterizes the precedence of dynamic permissions over the static ones, as a consequence of the **per-user-override** option (Example 14-3). Refer to Example 14-6 to verify that the dynamic ACEs just downloaded do not permit HTTPS, as opposed to the DMZ static **access-list** configured in Example 14-1. Telnet is now allowed through, despite not being part of the original interface ACL.

Example 14-7 Authorization ACL Takes Precedence over Interface ACL

```

! HTTPS, not allowed on downloaded ACEs, gets blocked

%ASA-6-109025: Authorization denied (acl=AAA-user-user1-E3040000) for user
'user1' from
172.21.21.101/1229 to 172.16.200.200/443 on interface dmz using TCP

! Telnet, not originally allowed on static ACL, gets through the ASA

%ASA-6-302013: Built outbound TCP connection 86 for outside:172.16.200.200/23
(172.16.200.200/23) to dmz:172.21.21.101/1230 (172.21.21.101/1230) (user1)

```

Scenario 3: Cut-Through Proxy with Locally Defined ACL

This scenario illustrates how CS-ACS can instruct the firewall (NAS) to assign a locally defined ACL as part of the authorization process. This may be interesting if the administrator does not want to download long ACLs but has the inconvenience that per-user ACLs need to be maintained in each NAS.

Example 14-8 defines a local ACL called GROUP2 referenced by the value of the IETF *Filter-ID* attribute, as shown in Example 14-9.

Example 14-8 Defining a Local ACL That Might Be Referenced by CS-ACS

```

access-list GROUP2 extended permit tcp any any eq www
access-list GROUP2 extended permit tcp any any eq ssh

```

Example 14-9 *Assigning the “Filter-ID” Attribute to a User Group on CS-ACS*

```
ACS/Group Settings : GROUP2
IETF RADIUS Attributes
[011] Filter-Id
GROUP2
```

Example 14-10 illustrates the delivery of the *Filter-ID* attribute on the RADIUS Response message, whereas Example 14-11 demonstrates how to verify the association of the local ACL (GROUP2) to the authenticated user *user2* (a member of GROUP2). To better differentiate what is going on in each scenario, it is interesting to compare the user-related information in Examples 14-11 and 14-6.

Example 14-10 *ASA Receives “Filter-ID” from the RADIUS Server*

```
RADIUS packet decode (response)
[output suppressed]
Radius: Type = 11 (0x0B) Filter-Id
Radius: Length = 8 (0x08)
Radius: Value (String) =
47 52 4f 55 50 32 | GROUP2
[output suppressed]
```

Example 14-11 *Verifying the Dynamic ACL Assigned via “Filter-ID” Attribute*

```
! Displaying the dynamic ACL assigned to user2

ASA1# show uauth user2
user 'user2' at 172.21.21.101, authenticated
access-list GROUP2 (*)
absolute timeout: 0:05:00
inactivity timeout: 0:00:00
!
! Displaying the details of the dynamic access-list

ASA1# show access-list GROUP2
access-list GROUP2; 2 elements; name hash: 0xd5211e1e
access-list GROUP2 line 1 extended permit tcp any any eq www (hitcnt=2) 0x64e09b05
access-list GROUP2 line 2 extended permit tcp any any eq ssh (hitcnt=1) 0x37d057b3
!

! dynamic ACL takes precedence over static interface ACL

%ASA-6-109025: Authorization denied (acl=GROUP2) for user 'user2' from
172.21.21.101/1236 to 172.16.200.200/23 on interface dmz using TCP
```

Scenario 4: Cut-Through Proxy with Downloadable ACLs

Scenario 4 differs from Scenarios 2 and 3 because it uses another form of authorization ACL called *Downloadable ACL (DACL)*. These are ACLs defined as Shared Profile Components in CS-ACS and can be later assigned to user groups. This is a flexible and scalable option for controlling authorization because changes can be centralized in one place and ACL definitions can be reused on several CS-ACS User Groups.

Example 14-12 summarizes the creation and assignment of a DACL in CS-ACS.

Example 14-12 *Defining (and Assigning) a Downloadable ACL on CS-ACS*

```
! Defining the contents of a Downloadable IP ACL

ACS/Shared Profile Components/Downloadable IP ACLs
Name : DACL1
ACL Contents : ACL1
    permit tcp any any eq 80
    permit icmp any any echo
!
! Assigning the DACL named "DACL1" to User Group "GROUP1"

ACS/Group Settings : GROUP1
Downloadable ACLs - Assign IP ACL: DACL1
```

Example 14-13 illustrates how CS-ACS delivers a DACL to ASA. There are two *Authentication Request* messages and two corresponding *RADIUS Response* messages. The first Request-Response pair is similar to what has been studied so far, with the distinction that only the name of the DACL is passed to the NAS in the form of a Cisco-AV-Pair (*ACS:CiscoSecure-Defined-ACL*). After receiving the name of the DACL to be applied, ASA sends a second *Authentication-Request* using the DACL name as username and a "NULL" value for the password. ASA also sends another Cisco-AV-Pair (*aaa:event=acl-download*) and receives as response the individual components of the DACL (*ip:inacl* attributes).

Example 14-13 *CS-ACS Delivers a Downloadable ACL to ASA*

```
! ASA sends regular Authentication Request for user "user1" to RADIUS Server

RADIUS packet decode (authentication request)
[output suppressed]
Radius: Type = 1 (0x01) User-Name
Radius: Length = 7 (0x07)
Radius: Value (String) =
75 73 65 72 31 | user1
[output suppressed]
```



```

!
! ASA receives first RADIUS response containing the name of the DACL to be down-
loaded
RADIUS packet decode (response)
[output suppressed]
Radius: Type = 26 (0x1A) Vendor-Specific
Radius: Length = 62 (0x3E)
Radius: Vendor ID = 9 (0x00000009)
Radius: Type = 1 (0x01) Cisco-AV-pair
Radius: Length = 56 (0x38)
Radius: Value (String) =
41 43 53 3a 43 69 73 63 6f 53 65 63 75 72 65 2d      | ACS:CiscoSecure-
44 65 66 69 6e 65 64 2d 41 43 4c 3d 23 41 43 53    | Defined-ACL=#ACS
41 43 4c 23 2d 49 50 2d 44 41 43 4c 31 2d 34 61     | ACL#-IP-DACL1-4a
61 63 36 31 38 64                                   | ac618d
[output suppressed]
!
! ASA sends new Authentication Request using the DACL name as username (null
password)
RADIUS packet decode (authentication request)
[output suppressed]
Radius: Type = 1 (0x01) User-Name
Radius: Length = 28 (0x1C)
Radius: Value (String) =
23 41 43 53 41 43 4c 23 2d 49 50 2d 44 41 43 4c    | #ACSACL#-IP-DACL
31 2d 34 61 61 63 36 31 38 64                       | 1-4aac618d
[output suppressed]
Radius: Type = 26 (0x1A) Vendor-Specific
Radius: Length = 23 (0x17)
Radius: Vendor ID = 9 (0x00000009)
Radius: Type = 1 (0x01) Cisco-AV-pair
Radius: Length = 17 (0x11)
Radius: Value (String) =
61 61 61 3a 73 65 72 76 69 63 65 3d 76 70 6e      | aaa:service=vpn
Radius: Type = 26 (0x1A) Vendor-Specific
Radius: Length = 30 (0x1E)
Radius: Vendor ID = 9 (0x00000009)
Radius: Type = 1 (0x01) Cisco-AV-pair
Radius: Length = 24 (0x18)
Radius: Value (String) =
61 61 61 3a 65 76 65 6e 74 3d 61 63 6c 2d 64 6f   | aaa:event=acl-do
77 6e 6c 6f 61 64                                   | wnload
[output suppressed]
!

```

```

! ACS sends a second Response detailing the DACL contents (as individual ACEs)

RADIUS packet decode (response)
[output suppressed]
Radius: Type = 26 (0x1A) Vendor-Specific
Radius: Length = 43 (0x2B)
Radius: Vendor ID = 9 (0x00000009)
Radius: Type = 1 (0x01) Cisco-AV-pair
Radius: Length = 37 (0x25)
Radius: Value (String) =
69 70 3a 69 6e 61 63 6c 23 31 3d 70 65 72 6d 69      | ip:inacl#1=permi
74 20 74 63 70 20 61 6e 79 20 61 6e 79 20 65 71      | t tcp any any eq
20 38 30                                               | 80
Radius: Type = 26 (0x1A) Vendor-Specific
Radius: Length = 43 (0x2B)
Radius: Vendor ID = 9 (0x00000009)
Radius: Type = 1 (0x01) Cisco-AV-pair
Radius: Length = 37 (0x25)
Radius: Value (String) =
69 70 3a 69 6e 61 63 6c 23 32 3d 70 65 72 6d 69      | ip:inacl#2=permi
74 20 69 63 6d 70 20 61 6e 79 20 61 6e 79 20 65      | t icmp any any e
63 68 6f                                               | cho
[output suppressed]

```

Example 14-14 displays the *Downloadable ACL* and illustrates the creation of an ICMP connection after authorization. It is interesting to observe ASA's identity awareness in the *Built Connection* message.

Example 14-14 Verifying the Downloadable ACL Details

```

! Displaying the DACL assigned to user1

ASA1# show uauth user1
user 'user1' at 172.21.21.101, authenticated
  access-list #ACSACL#-IP-DACL1-4aac618d (*)
  absolute timeout: 0:05:00
  inactivity timeout: 0:00:00
!
! User "user1" pings the address 172.16.200.200 (notice the identity-awareness)

%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.200.200/0 gaddr
172.21.21.101/512 laddr 172.21.21.101/512 (user1)
!
! Verifying the DACL details (notice the hitcount in the ICMP entry)

```

```

ASA1# show access-list #ACSACL#-IP-DACL1-4aac618d
access-list #ACSACL#-IP-DACL1-4aac618d; 2 elements; name hash: 0x7df6ced9 (dynamic)
access-list #ACSACL#-IP-DACL1-4aac618d line 1 extended permit tcp any any eq www
(hitcnt=2) 0x3bb3ba32
access-list #ACSACL#-IP-DACL1-4aac618d line 2 extended permit icmp any any echo
(hitcnt=1) 0x003bbecc

```

Scenario 5 - HTTP Listener

The analyses of Scenarios 1 through 4 focus on illustrating the authorization processes that can extend the basic Cut-Through Proxy authentication functionality.

Knowing now what happens behind the scenes in those scenarios, it is time to shift gears to the study of the *HTTP Listener* technique, a feature designed to enhance the *authentication experience* of a user subject to Cut-Through Proxy interception.

The HTTP Listener mechanism, working with the **aaa authentication match** command, redirects all the web requests intended to traverse the firewall to an authentication web page served by the ASA.

Example 14-15 documents the additional commands necessary to enable the HTTP Listener on a given interface already configured for Cut-Through Proxy. This example assumes that the commands documented in Examples 14-1 and 14-3 are already in place.

Example 14-15 *Enabling HTTP Listener on an Interface*

```

! Enabling the HTTP Listener mechanism on interface "dmz"

! aaa authentication listener http dmz port www redirect
!

! Enabling the HTTPS listener mechanism on interface "dmz"

! aaa authentication listener https dmz port https redirect

```

Note The **redirect** keyword instructs the ASA to direct HTTP and HTTPS requests to an authentication web page served by the firewall itself.

Note The tool shown in the browser's windows at the bottom of Figures 14-4 and 14-5 is the *HTTP Watch Basic Edition*, an HTTP viewer that provides visibility of the HTTP methods invoked, the operations happening within a session, and the corresponding status codes. This information may be useful for revealing, for instance, the redirect messages employed by the HTTP Listener process.

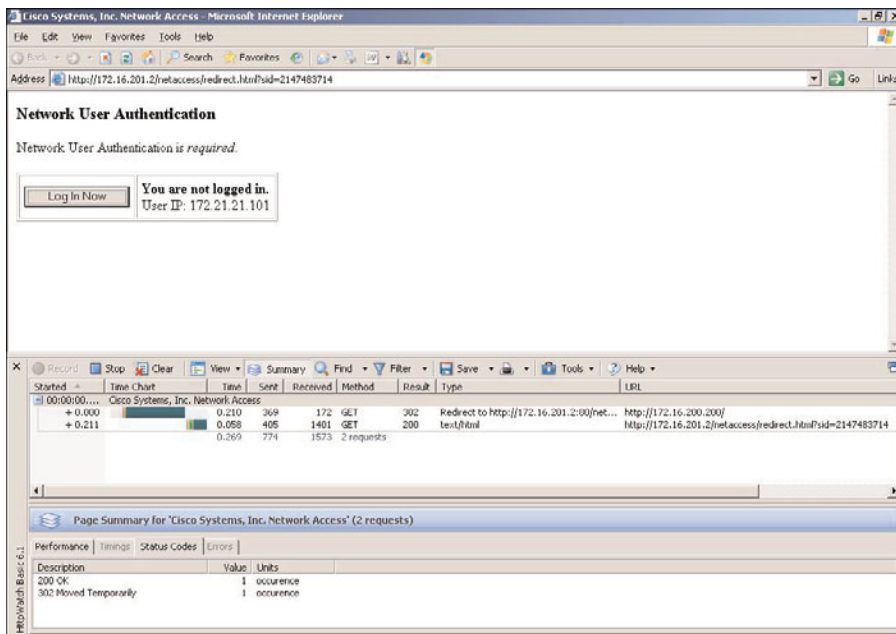


Figure 14-4 HTTP Listener - User Perspective (1)

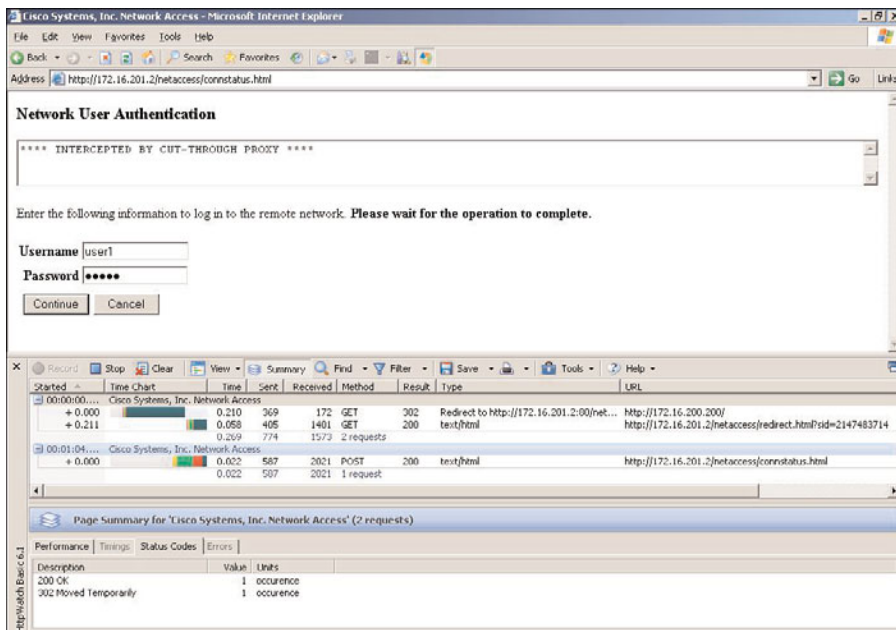


Figure 14-5 HTTP listener - User Perspective (2)

Figure 14-4 shows the browser screen presented to the user after it attempts to open an HTTP connection to **172.16.200.200**. The IP address included in the URL shown in this figure has changed to **172.16.201.1**, which corresponds to the “dmz” interface of the ASA (refer to Figure 14-2 for addressing details).

Figure 14-5 displays the browser screen presented to the users after they click on the **Log In Now** button presented on Figure 14-4. This new process helps characterize that the original connection attempt has been intercepted and that user credentials are required before traffic can be allowed through the ASA.

Note If the command `aaa authentication secure-http-client` is added to the configuration analyzed in Example 14-15, it instructs the ASA to always receive the user credentials via HTTPS, even if the original connection used HTTP.

Figure 14-6 is an attempt to consolidate the sequence of operations that a packet can undergo when it arrives at an ASA interface configured for Cut-Through Proxy. The conception of the flowchart assumes that authorization is performed with the contribution of some of the RADIUS attributes that have been discussed so far.

It is indispensable to keep in mind that this flowchart covers the order of operations that are part of the ASA algorithm, in the inbound direction, before any eventual NAT process comes into play. The focus here is to visualize some processes that have precedence over the Cut-Through Proxy feature (existence of the flow, availability of routing information, and explicit permission for the triggering protocol in the interface ACL).

IOS User-Level Control with Auth-Proxy

The previous section presented a thorough analysis of the Cut-Through Proxy operation on the ASA family. In the current one a similar IOS mechanism called *Auth-Proxy* is covered in detail. Although the conception and purpose of the features are similar, they have some distinct operational behaviors. The differences that deserve special mention follow:

- Although the dynamic permissions created by Cut-Through Proxy are natively stateful, IOS Auth-Proxy permissions are originally stateless. Nevertheless, after being combined with CBAC or Zone Policy Firewall, the Auth-Proxy permissions will undergo stateful inspection and behave much like as ASA's.
- In ASA, inbound Interface ACLs initially have precedence over Cut-Through Proxy derived permissions (refer to Figure 14-6). It is therefore necessary to explicitly allow the application protocol in this ACL before the interception can take place. If the **per-user-override** option is enabled and a DACL is downloaded, the dynamic permissions take precedence over the static ones.
- In IOS, the Auth-Proxy intercepts the application protocol that triggers authentication before it reaches the inbound interface ACL. You can see how this works

through the analysis of some usage scenarios. (Examples 14-19 and 14-22 cover this behavior.)

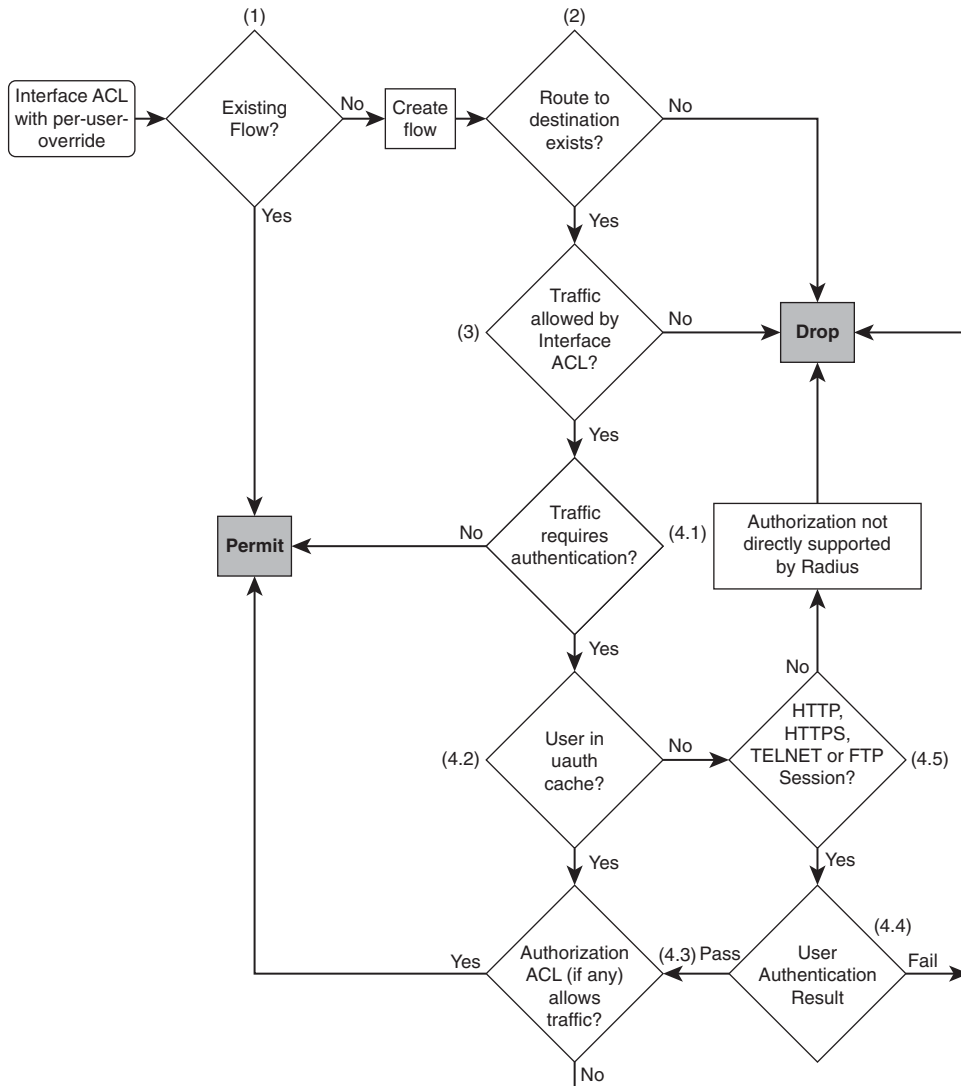


Figure 14-6 Cut-Through Proxy Flowchart Using RADIUS Attributes for Authorization

- Although IOS uses the *proxyacl* RADIUS VSA to download individual ACEs to the NAS, ASA uses the *ip:inacl* VSA to accomplish this task. (The comparison between Examples 14-5 and 14-19 reveals this distinction.)
- The RADIUS server can send the IETF *Filter-ID* attribute pointing to an ASA locally defined ACL. The activation of such an ACL in IOS currently requires the usage of

the *tag-name* VSA. This is discussed in the section “User-Based Zone Policy Firewall.”

After this brief introduction, you can now get back to a set of practical usage scenarios that serve to emphasize the potential of the Auth-Proxy feature.

For the following examples Telnet is chosen as the triggering protocol because its connections are long living compared to HTTP. This makes life easier when dealing with **debug** commands and viewing established sessions. After becoming familiar with Auth-Proxy concepts, you are greatly encouraged to proceed an equivalent analysis using HTTP (or even better, HTTPS) as the triggering protocol.

Figure 14-7 shows the reference topology used for the Auth-Proxy scenarios that follow.

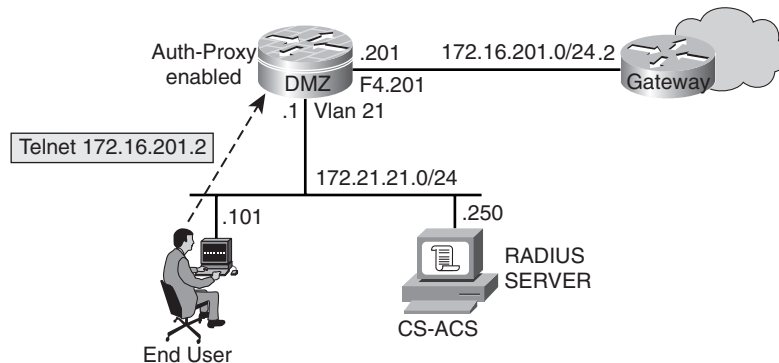


Figure 14-7 Network Topology for the Auth-Proxy Usage Scenarios

Note All the IOS Auth-Proxy scenarios use RADIUS as the authentication protocol. The reasons for considering that a convenient choice are outlined in the section “Selecting the Authentication Protocol.”

Example 14-16 shows the relevant AAA commands for the Auth-Proxy scenarios.

Example 14-17 complements the previous one, by including the necessary commands to enable Auth-Proxy for Telnet interception.

Example 14-16 Baseline AAA Configuration for Auth-Proxy Scenarios

```
aaa new-model
!
! Instructing the NAS to receive, send and process Vendor Specific Attributes (VSAs)
radius-server vsa send accounting
radius-server vsa send authentication
```

```

!
! Instructing NAS to send the IETF "Service Type" attribute to the RADIUS Server

radius-server attribute 6 on-for-login-auth
!
! Defining the source interface for RADIUS packets

ip radius source-interface Vlan21
!
! Defining an AAA server-group called "RADIUS1"

aaa group server radius RADIUS1
  server 172.21.21.250 auth-port 1812 acct-port 1813
  server-private 172.21.21.250 auth-port 1812 acct-port 1813 key 7
  13061E010803557878
!
! This method list will be applied to the console and VTY lines

aaa authentication login CONSOLE none
!
! Auth-Proxy service uses the AAA server-group "RADIUS1" previously defined

aaa authentication login default group RADIUS1
aaa authorization network default group RADIUS1
aaa authorization auth-proxy default group RADIUS1
aaa accounting auth-proxy default start-stop group RADIUS1
!
! Excluding console and VTY lines from the "default" login method (that uses
RADIUS)
line con 0
login authentication CONSOLE
line vty 0 4
  login authentication CONSOLE
  transport input telnet ssh

```

Example 14-17 Baseline Auth-Proxy Configuration

```

! Defining an ACL to be applied to the same interface as Auth-Proxy

access-list 100 permit udp host 172.21.21.250 eq 1812 host 172.21.21.1
access-list 100 permit udp host 172.21.21.250 eq 1813 host 172.21.21.1
access-list 100 permit tcp any 172.16.201.0 0.0.0.255 eq telnet

! Defining the Auth-Proxy policy to intercept Telnet traffic

ip admission name ADMISSION1 proxy telnet

```



```

!
! Applying the Auth-Proxy policy to interface Vlan21 (Auth-Proxy incoming inter-
face)
interface Vlan21
  description *** INSIDE interface ***
  ip address 172.21.21.1 255.255.255.0
  ip access-group 100 in
  ip admission ADMISSION1

```

Scenario 1: IOS Auth-Proxy with Downloadable Access Control Entries

This is completely analogous to what was done for ASA in Scenario 2 of the “ASA User-Level Control with Cut-Through Proxy” section. Example 14-18 is compared with Example 14-4 to see what changes in CS-ACS configuration. (IOS needs the `priv-lvl=15` setting and uses `proxyacl` instead of `ip:inacl`.)

Example 14-19 illustrates Auth-proxy performing the interception and interacting with the RADIUS server, whereas Example 14-20 shows the authentication and authorization results.

Example 14-18 Defining Individual ACEs on CS-ACS for IOS Auth-Proxy

```

ACS/Group Settings : GROUP1
[009\001] cisco-av-pair
  priv-lvl=15
  proxyacl#1=permit tcp any any eq 22
  proxyacl#2=permit tcp any any eq 23

```

Example 14-19 Telnet Session Intercepted by Auth-Proxy

```

! Telnet Session is intercepted by Auth-Proxy process (before reaching interface
ACL)
AUTH-PROXY creates info:
  cliaddr - 172.21.21.101, cliport - 1562
  seraddr - 172.16.201.2, serport - 23
  ip-srcaddr 172.21.21.101
  pak-srcaddr 0.0.0.0

! NAS sends request to CS-ACS and receives individual ACEs (proxyacl)

RADIUS(00000000C): Send Access-Request to 172.21.21.250:1812 id 1645/12, len 104
RADIUS: authenticator 73 DC D7 7B 91 B4 61 38 - 4E 65 CB A5 B3 4F AD 9D
RADIUS: User-Name          [1] 7 "user1"
[output suppressed]
RADIUS: Received from id 1645/12 172.21.21.250:1812, Access-Accept, len 148
RADIUS: authenticator ED 65 FB F6 64 B9 33 6D - A3 5E B8 5F 14 36 D4 21
RADIUS: Vendor, Cisco      [26] 19

```

```

RADIUS: Cisco AVpair [1] 13 "priv-lvl=15"
RADIUS: Vendor, Cisco [26] 43
RADIUS: Cisco AVpair [1] 37 "proxyacl#1=permit tcp any any eq 22"
RADIUS: Vendor, Cisco [26] 43
RADIUS: Cisco AVpair [1] 37 "proxyacl#2=permit tcp any any eq 23"
[output suppressed]

```

Example 14-20 Verifying Authenticated Users and Downloaded ACEs

```

DMZ# show ip auth-proxy cache
Authentication Proxy Cache
Client Name user1, Client IP 172.21.21.101, Port 1562, timeout 60, Time Remaining
60, state INTERCEPT
!
! Details about the current Auth-Proxy session

DMZ# show epm session ip 172.21.21.101
Admission feature      : Authproxy
AAA Policies           :
Proxy ACL              : permit tcp any any eq 22
Proxy ACL              : permit tcp any any eq 23

!
! Viewing Dynamic Entries (for host 172.21.21.101) added to the interface ACL

DMZ# show access-list
Extended IP access list 100
    permit tcp host 172.21.21.101 any eq 22 (18 matches)
    permit tcp host 172.21.21.101 any eq telnet (70 matches)
10 permit udp host 172.21.21.250 eq 1812 host 172.21.21.1 (1 match)
20 permit udp host 172.21.21.250 eq 1813 host 172.21.21.1 (1 match)
30 permit tcp any 172.16.201.0 0.0.0.255 eq telnet

```

Scenario 2: IOS Auth-Proxy with Downloadable ACLs

This is completely analogous to what was done for ASA in Scenario 4 of the Cut-Through Proxy analysis. It is instructive to compare Examples 14-21 and 14-12 to detect their similarities. (Actually the only difference is the `priv-lvl=15` setting on IOS.)

Example 14-21 Assigning a Downloadable ACL to a User Group on CS-ACS

```

ACS/Group Settings : GROUP1
Downloadable ACLs - Assign IP ACL: DACL1
    permit tcp any any eq 80
    permit icmp any any echo

```

```
[009\001] cisco-av-pair
priv-lvl=15
```

Example 14-22 details the delivery of the DACL to IOS. Similarly to what was described in Example 14-13, there are two Access-Request (*Authentication Request*) messages and two RADIUS Access-Accept (*Response*) messages. The first Request-Accept pair shows the ACS:CiscoSecure-Defined-ACL Cisco AV-Pair, which contains the name of the DACL to be assigned. The second Access-Request uses the DACL name as username and a “NULL” value for the password. IOS also sends another Cisco-AV-Pair (*aaa:event=acl-download*) and receives as *response* the individual components of the DACL (*ip:inac*l attributes).

Before sending the Cisco AV-Pair *aaa:event=acl-download*, ASA sends the Cisco AV-Pair *aaa:service=vpn*. IOS, in contrast, sends the AV-Pair *aaa:service=ip_admission* before the AV-Pair *aaa:event=acl-download*.

Example 14-23 displays the details of the user session created in Example 14-22. IOS clearly identifies the DACL as a “per-user” ACL and inserts its entries in the static interface ACL (before the original ACEs). Figure 14-8 shows a sample **Passed Authentications** log on CS-ACS, including the DACL assigned to the user.

Passed Authentications active.csv Refresh Download

Regular Expression Start Date & Time End Date & Time Rows per Page

09/13/2009, 10:00:00 mm/dd/yyyy, hh:mm:ss 50

Apply Filter Clear Filter

Results from 09/13/2009, 10:00:00

Date	Time	Message-Type	User-Name	Group-Name	Caller-ID	Access Device	NAS-IP-Address	Downloadable ACL
09/13/2009	10:00:54	Authen OK	user1	GROUP1	172.21.21.101	DMZ_RADIUS	172.21.21.1	DACL1

Figure 14-8 Example of “Passed Authentications” Including DACL Assignment in CS-ACS

Example 14-22 CS-ACS Delivers Downloadable ACL to IOS

```
AUTH-PROXY creates info:
    cliaddr - 172.21.21.101, cliport - 1085
    seraddr - 172.16.201.2, serport - 23
    ip-srcaddr 172.21.21.101
    pak-srcaddr 0.0.0.0
!
! NAS sends Access Request to CS-ACS and receives name of the DACL to be applied

RADIUS(00000006): Send Access-Request to 172.21.21.250:1812 id 1645/4, len 104
RADIUS: authenticator 67 06 F7 BB F1 81 BE 96 - 29 2D C9 24 89 00 2B 31
RADIUS: User-Name [1] 7 "user1"
!
```

```
[output suppressed]
RADIUS: Received from id 1645/4 172.21.21.250:1812, Access-Accept, len 124
RADIUS: authenticator 6D 19 94 84 EF C0 28 C3 - EF AB 8E FE 1F E9 7B 28
RADIUS: Vendor, Cisco [26] 19
RADIUS: Cisco AVpair [1] 13 "priv-lvl=15"
RADIUS: Vendor, Cisco [26] 62
RADIUS: Cisco AVpair [1] 56 "ACS:CiscoSecure-Defined-ACL=#ACSACL#-IP-DACL1-4aac618d"
[output suppressed]
!
! NAS sends second Access Request using DACL name as username (null password)

RADIUS(00000000): Send Access-Request to 172.21.21.250:1812 id 1645/5, len 134
RADIUS: authenticator 94 3C 9D F1 C1 93 25 2A - F3 9E DA C9 B0 15 FC B2
RADIUS: NAS-IP-Address [4] 6 172.21.21.1
RADIUS: User-Name [1] 28 "#ACSACL#-IP-DACL1-4aac618d"
RADIUS: Vendor, Cisco [26] 32
RADIUS: Cisco AVpair [1] 26 "aaa:service=ip_admission"
RADIUS: Vendor, Cisco [26] 30
RADIUS: Cisco AVpair [1] 24 "aaa:event=ac1-download"
!
! ACS sends second Response detailing the DACL contents (as individual ACEs)

RADIUS: Received from id 1645/5 172.21.21.250:1812, Access-Accept, len 179
RADIUS: authenticator 69 A2 A7 BB 15 AF 3C EB - A3 D7 12 F0 F5 04 54 F2
RADIUS: Vendor, Cisco [26] 43
RADIUS: Cisco AVpair [1] 37 "ip:inacl#1=permit tcp any any eq 80"
RADIUS: Vendor, Cisco [26] 43
RADIUS: Cisco AVpair [1] 37 "ip:inacl#2=permit icmp any any echo"
[output suppressed]
```

Example 14-23 Verifying the Downloadable ACL Details

```
DMZ# show ip auth-proxy cache
Authentication Proxy Cache
  Client Name user1, Client IP 172.21.21.101, Port 1085, timeout 60, Time Remaining
  60, state INTERCEPT
!
DMZ# show epm session ip 172.21.21.101
Admission feature : Authproxy
AAA Policies :
ACS ACL : xACSACLx-IP-DACL1-4aac618d
!
! After Auth-Proxy "user1" pings 172.16.201.2 and opens the page
http://172.16.200.200
```

```
DMZ# show access-list
Extended IP access list 100
    permit tcp host 172.21.21.101 any eq www (12 matches)
    permit icmp host 172.21.21.101 any echo (4 matches)
    10 permit udp host 172.21.21.250 eq 1812 host 172.21.21.1 (2 matches)
    20 permit udp host 172.21.21.250 eq 1813 host 172.21.21.1 (2 matches)
    30 permit tcp any 172.16.201.0 0.0.0.255 eq telnet (31 matches)
Extended IP access list xACSACLx-IP-DACL1-4aac618d (per-user)
    10 permit tcp any any eq www
    20 permit icmp any any echo
```

Scenario 3: Combining Classic IP Inspection (CBAC) and Auth-Proxy

As stated earlier, despite performing authorization, Auth-proxy is not inherently stateful. This behavior can be changed through interactions with technologies such as CBAC or ZFW. The current scenario dedicates some time to the study of CBAC and Auth-Proxy integration.

Example 14-24 shows the commands that should be added to those in Examples 14-16 and 14-17, to make the integration of Auth-Proxy and CBAC come true. The purpose of ACL 199 is just to state clearly that no inbound connections are accepted. Dynamic openings for the return traffic are created as needed by the **ip inspect** rule called TCP1.

Example 14-25 details some important aspects of this particular environment:

- Auth-Proxy happens first.
- CBAC creates the temporary openings for return traffic. This happens for the protocols that are part of the authorization ACL provided by Auth-Proxy.
- The user might need to Telnet again to the destination host after successful Auth-Proxy authentication and authorization. That is the motivation for configuring a customized **ip admission auth-proxy-banner** in this scenario.

Example 14-24 Adding CBAC to an Auth-Proxy Enabled Interface

```
! Defining an ACL for the OUTSIDE interface (f4.201) - no static permissions
inbound
access-list 199 deny ip any any log
!
! Creating a CBAC Rule for TCP (this rule dynamically opens ACL 199 for return
traffic)
ip inspect name TCP1 tcp audit-trail off
!
interface Vlan21
description *** INSIDE interface ***
ip address 172.21.21.1 255.255.255.0
```

```

ip access-group 100 in
ip admission ADMISSION1
ip inspect TCP1 in
!
interface FastEthernet4.201
description *** connection to ASA (DMZ) ***
encapsulation dot1Q 201
ip address 172.16.201.201 255.255.255.0
ip access-group 199 in
!
! Defining a banner for Auth-Proxy

ip admission auth-proxy-banner telnet ^C
*****
                INTERCEPTED BY IOS AUTH-PROXY FEATURE
        AFTER AUTHENTICATION YOU MAY NEED TO RECONNECT TO DESTINATION HOST
*****
^C

```

Example 14-25 Visualizing Auth-Proxy and CBAC Interactions

```

! User telnets to 172.16.201.2 and Auth-Proxy intercepts the connection

AUTH-PROXY creates info:
    cliaddr - 172.21.21.101, cliport - 1092
    seraddr - 172.16.201.2, serport - 23
    ip-srcaddr 172.21.21.101
    pak-srcaddr 0.0.0.0
!
! User enters credentials in the Firewall prompt and CBAC entry is created.

FIREWALL OBJ_CREATE: Pak 83B379A8 sis 84332968 initiator_addr (172.21.21.101:1092)
responder_addr (172.16.201.2:23)
initiator_alt_addr (172.21.21.101:1092) responder_alt_addr (172.16.201.2:23)
FIREWALL OBJ_CREATE: sid 84862F6C acl 199 Prot: tcp
Src 172.16.201.2 Port [23:23]
Dst 172.21.21.101 Port [1092:1092]
FIREWALL OBJ_CREATE: create host entry 84652D64 addr 172.16.201.2 bucket 119
(vrf 0:0)
insp_cb 0x84C71B00
FIREWALL OBJ_DELETE: delete host entry 84652D64 addr 172.16.201.2
!

! User is authenticated and authorized as in previous examples

```

```

AUTH-PROXY: Allocate Unique_id C
[output suppressed]
RADIUS(0000000C): Send Access-Request to 172.21.21.250:1812 id 1645/10, len 104
RADIUS: Received from id 1645/10 172.21.21.250:1812, Access-Accept, len 124
!
! Displaying information about the authenticated user

DMZ# show ip auth-proxy cache
Authentication Proxy Cache
  Client Name user1, Client IP 172.21.21.101, Port 1092, timeout 60, Time Remaining
59, state ESTAB
!
DMZ# show epm session ip 172.21.21.101
Admission feature      : Authproxy
AAA Policies           :
ACS ACL                : xACSACLx-IP-DACL1-4aac618d
!
DMZ# show access-list xACSACLx-IP-DACL1-4aac618d
Extended IP access list xACSACLx-IP-DACL1-4aac618d (per-user)
    10 permit tcp any any eq www
    20 permit icmp any any echo
!
! User was authenticated but no Telnet session to the end host has been created
yet.
! User telnets again and traffic gets inspected by CBAC. No Auth-Proxy anymore.

FIREWALL* OBJ_CREATE: Pak 83D1D13C sis 843326A0 initiator_addr
(172.21.21.101:1093) responder_addr (172.16.201.2:23)
initiator_alt_addr (172.21.21.101:1093) responder_alt_addr (172.16.201.2:23)
FIREWALL OBJ-CREATE: sid 84862F18 acl 199 Prot: tcp
  Src 172.16.201.2 Port [23:23]
  Dst 172.21.21.101 Port [1093:1093]
[output suppressed]
!
! Displaying the inspect sessions created by CBAC

DMZ# show ip inspect sessions
Established Sessions
Session 843326A0 (172.21.21.101:1093)=>(172.16.201.2:23) tcp SIS_OPEN
!
! The matches in ACL 199 are not directly visible. The following command is need-
ed:
DMZ# show ip inspect sis detail
Established Sessions
Session 843326A0 (172.21.21.101:1093)=>(172.16.201.2:23) tcp SIS_OPEN
  Created 00:07:43, Last heard 00:03:26

```

```
Bytes sent (initiator:responder) [129:8777]
Initiator->Responder Window size 65201 Scale factor 0
Responder->Initiator Window size 8192 Scale factor 0
In SID 172.16.201.2[23:23]>172.21.21.101[1093:1093] on ACL 199 (274 matches)
```

User-Based Zone Policy Firewall

The previous section examined in detail the operation of the IOS Auth-Proxy feature and how it may be integrated with CBAC to create dynamic stateful permissions before allowing traffic to cross the firewall.

The current section initially analyzes two techniques that leverage Auth-Proxy to bring user-group membership awareness in the IOS device. This concept of *membership* means that after successful user authentication and authorization, IOS will have local knowledge of the group to which the user belongs.

Later in the section zone-based policies that use as an additional criterion the membership data previously obtained are created. This is another way of inducing a stateful behavior for Auth-Proxy and is referred to as the *User-based Zone Policy Firewall*.

Note A question might arise at this point. “What is meant by this *membership* concept?” (In all the previous discussions, the users were always members of some user group.) What has changed? Well, it is true that the membership information was already in there, but it was on the CS-ACS side only. There was no visibility on the NAS. This is the difference that is explored next.

Establishing user-group Membership Awareness in IOS - Method 1

The method relies on a Cisco AV-Pair called *supplicant-group*, which directly corresponds to the **user-group** to which the user should be assigned. This is the simplest way to establish **user-group** membership visibility on the NAS side.

Example 14-26 displays the CS-ACS Group settings that enables the *supplicant-group* AV-Pair to be sent to IOS. Example 14-27 depicts the delivery of this information to the NAS, by means of the authorization process that the reader has already gotten acquainted with.

Example 14-26 Assigning the supplicant-group AV Pair to a user-group in CS-ACS

```
ACS/Group Settings : GROUP1
[009\001] cisco-av-pair
priv-lvl=15
supplicant-group=GROUP1
```


Example 14-27 *ACS Delivers supplicant-group Attribute to NAS After Auth-Proxy*

```
AUTH-PROXY creates info:
    cliaddr - 172.21.21.101, cliport - 1108
    seraddr - 172.16.201.2, serport - 23
    ip-srcaddr 172.21.21.101
    pak-srcaddr 0.0.0.0

RADIUS(00000015): Send Access-Request to 172.21.21.250:1812 id 1645/21, len 104
RADIUS: authenticator 61 3B 1D 21 54 8A C5 3C - 14 6F C7 5E 73 E9 72 36
RADIUS: User-Name [1] 7 "user1"
[output suppressed]
RADIUS: Received from id 1645/21 172.21.21.250:1812, Access-Accept, len 93
RADIUS: authenticator 43 A9 2F 23 EC 7F 7B 19 - B5 AF 6D 1B 40 81 85 25
RADIUS: Vendor, Cisco [26] 19
RADIUS: Cisco AVpair [1] 13 "priv-lvl=15"
RADIUS: Vendor, Cisco [26] 31
RADIUS: Cisco AVpair [1] 25 "supplicant-group=GROUP1"

! Visualizing user to group association in the NAS

DMZ# show ip auth-proxy cache
Authentication Proxy Cache
  Client Name user1, Client IP 172.21.21.101, Port 1108, timeout 60, Time Remaining
  60, state INTERCEPT
!
DMZ# show epm session ip 172.21.21.101
Admission feature      : Authproxy
AAA Policies           :
Supplicant-Group      : GROUP1
!
DMZ# show user-group
Usergroup : GROUP1

-----
User Name      Type      Interface      Learn      Age (min)
-----
172.21.21.101  IPv4      Vlan21        Dynamic      0
```

Establishing user-group Membership Awareness in IOS - Method 2

This is the second method to provide IOS with **user-group** membership visibility. This second technique uses a lot of indirect references, which brings more complexity.

Example 14-28 shows how to assign the *tag-name* attribute in CS-ACS Group Settings. In this particular scenario, the value for the attribute was not simply GROUP2 but

GROUP2-TAG instead. The reason for this choice is to clearly define each match operation executed in Example 14-29.

Example 14-29 introduces some new commands that work with the AAA configuration presented in Example 14-16 to match the *tag-name* value received from CS-ACS. The summary of operations is basically the following:

- Auth-Proxy intercepts the Telnet traffic and receives authorization information from the RADIUS server, containing a value for the *tag-name* attribute. This value is matched by a control tag **class-map**, which is used to define an Identity policy.
- The Identity policy just born enables the specification of a local user-group and an associated ACL.
- The user-group can be later matched to decide which security policy is enforced. (This type of operation is covered on Example 14-33.)

Example 14-28 *Assigning the tag-name AV Pair to a user-group in CS-ACS*

```
ACS/Group Settings : GROUP2
[009\001] cisco-av-pair
priv-lvl=15
tag-name=GROUP2-TAG
```

Example 14-29 *New Auth-Proxy Configuration Matching the tag-name AV-Pair*

```
! Defining a local ACL that will be assigned by the Identity policy
ip access-list extended ACL2
 permit tcp any 172.16.0.0 0.0.255.255 eq telnet
!
! Defining a class-map that matches the tag-name AV-Pair (control tag)

class-map type control tag match-all GROUP2-CLASS
 match tag GROUP2-TAG
!
! The Identity policy receives tag-name and assigns local user-group and ACL

identity policy GROUP2-TAG
 access-group ACL2
 user-group GROUP2
!
! This control tag policy-map is evoked by the new Auth-Proxy (IP Admission) policy
policy-map type control tag TAG-NAME
 class type control tag GROUP2-CLASS
 identity policy GROUP2-TAG
```

```

!
! Defining the Auth-Proxy policy to intercept Telnet traffic

ip admission name ADMISSION2 proxy telnet service-policy type tag TAG-NAME
!
! Assigning the Auth-Proxy policy to the input interface

interface Vlan21
ip address 172.21.21.1 255.255.255.0
ip access-group 100 in
ip admission ADMISSION2

```

Example 14-30 documents the delivery of the *tag-name* attribute to IOS, in the conventional RADIUS authorization fashion. After that, you can see how the *Tag and Template* process matches the received value inside the control tag **policy-map**.

Example 14-31 details the user-group information just obtained. To establish a clear distinction between the two methods just discussed, it is helpful to compare the output of **show epm session** command with that in Example 14-27. Example 14-31 also makes the dynamic opening, based on ACL2, immediately noticeable on access-list 100.

Example 14-30 CS-ACS Delivers the tag-name AV-Pair to the NAS

```

AUTH-PROXY creates info:
    cliaddr - 172.21.21.250, cliport - 4575
    seraddr - 172.16.201.2, serport - 23
    ip-srcaddr 172.21.21.250
    pak-srcaddr 0.0.0.0
AUTH-PROXY: Allocate Unique_id 18

RADIUS(00000018): Send Access-Request to 172.21.21.250:1812 id 1645/23, len 104
RADIUS: authenticator 0D 38 6E 4E DB 64 F8 EF - EB 55 0B BD E9 13 E8 B2
RADIUS: User-Name [1] 7 "user2"
[output suppressed]
RADIUS: Received from id 1645/23 172.21.21.250:1812, Access-Accept, len 89
RADIUS: authenticator 60 9A AB D3 40 F8 34 AC - B8 B9 84 FA 05 22 AE 97
RADIUS: Vendor, Cisco [26] 19
RADIUS: Cisco AVpair [1] 13 "priv-lvl=15"
RADIUS: Vendor, Cisco [26] 27
RADIUS: Cisco AVpair [1] 21 "tag-name=GROUP2-TAG"
[output suppressed]
!
! Tag and Template process matches the tag-name attribute

TT_EVE_DEB:Set the tag filter type value GROUP2-TAG

```

```

TT_EVE_DEB:Filter head 84775414 filter head new 8472C04C
TT_EVE_DEB:In function tt_client_tag_query
TT_EVE_DEB:Verifying the match parameters for policy-map TAG-NAME
TT_EVE_DEB:In function tt_is_match_valid
TT_EVE_DEB:Match tag filter
TT_EVE_DEB:Comparing tag : GROUP2-TAG to GROUP2-TAG

```

Example 14-31 Verifying User Group Membership on the NAS

```

DMZ# show ip auth-proxy cache
Authentication Proxy Cache
  Client Name user2, Client IP 172.21.21.250, Port 4575, timeout 60, Time Remaining
  60, state INTERCEPT
!
! Information about the indirect model used (class-map and policy-map)

DMZ# show epm session ip 172.21.21.250
Admission feature      : Authproxy
Tag Received         : GROUP2-TAG
Policy map used        : TAG-NAME
Class map matched    : GROUP2-CLASS
!
DMZ# show user-group
Usergroup : GROUP2

```

User Name	Type	Interface	Learn	Age (min)
172.21.21.250	IPv4	Vlan21	Dynamic	0

```

!
! A dynamic entry is created on ACL 100 using the definitions of ACL2

DMZ# show access-list
Extended IP access list 100
  permit tcp host 172.21.21.250 172.16.0.0 0.0.255.255 eq telnet (34 matches)
  10 permit udp host 172.21.21.250 eq 1812 host 172.21.21.1 (1 match)
  20 permit udp host 172.21.21.250 eq 1813 host 172.21.21.1 (1 match)
  30 permit tcp any 172.16.201.0 0.0.0.255 eq telnet
Extended IP access list ACL2
  10 permit tcp any 172.16.0.0 0.0.255.255 eq telnet

```

Integrating Auth-Proxy and the ZFW

The previous sections detailed two methods of associating users to local groups in IOS. The acquired user-to-group mapping information will now be applied to create distinct ZFW policies.

Example 14-32 summarizes ACS settings for GROUP1 and GROUP2, which respectively include *user1* and *user2*. Notice that the group definitions use method 1.

Example 14-32 CS-ACS Settings for Scenario 3

```
! Members of GROUP1 are assigned a DACL after authentication
ACS/Group Settings : GROUP1
Downloadable ACLs - Assign IP ACL: DACL1
permit tcp any any eq 80
permit icmp any any echo
[009\001] cisco-av-pair
priv-lvl=15
supplicant-group=GROUP1

! Members of GROUP2 are assigned an individual ACE after authentication
ACS/Group Settings : GROUP2
[009\001] cisco-av-pair
priv-lvl=15
supplicant-group=GROUP2
proxyacl#1=permit tcp any any eq 22
```

Example 14-33 shows a set of commands used to create a **zone-pair** security policy that takes into account the local user-group associations.

Example 14-33 Baseline Configuration for ZFW and Auth-Proxy Integration

```
! Defining inspect class-maps that match local user-group information

class-map type inspect match-all CLASS1
match user-group GROUP1
match protocol tcp
class-map type inspect match-all CLASS2
match user-group GROUP1
match protocol icmp
class-map type inspect match-all CLASS21
match user-group GROUP2
match protocol tcp
!
! Defining a policy-map for inspection
```

```

policy-map type inspect IN-OUT
  class type inspect CLASS11
    inspect
  class type inspect CLASS12
    inspect
    police rate 16000 burst 3000
  class type inspect CLASS21
    inspect
class class-default
  drop log
!
! Defining zones and zone-pairs

zone security INSIDE
zone security OUTSIDE
zone-pair security OUTBOUND source INSIDE destination OUTSIDE
  service-policy type inspect IN-OUT
!
! Defining an Auth-Proxy policy to intercept Telnet traffic

ip admission name ADMISSION proxy telnet inactivity-time 60
!
! Assigning interfaces to zones and applying the Auth-Proxy policy to VLAN21

interface Vlan21
  ip admission ADMISSION
  zone-member security INSIDE
!
interface FastEthernet4.201
  zone-member security OUTSIDE

```

Example 14-34 displays the following processes for *user2*, a member of GROUP2:

- Auth-Proxy intercepting Telnet traffic
- Supplicant-group=GROUP2 AV-Pair being assigned to IOS
- The original Telnet session being created by the ZFW
- A new SSH session controlled only by ZFW.

Example 14-34 *Auth-Proxy and Zone Firewall in Action*

```

AUTH-PROXY creates info:
    cliaddr - 172.21.21.250, cliport - 1496
    seraddr - 172.16.201.2, serport - 23
    ip-srcaddr 172.21.21.250
    pak-srcaddr 172.21.21.101
AUTH-PROXY: Allocate Unique_id 1E

RADIUS(0000001E): Send Access-Request to 172.21.21.250:1812 id 1645/27, len 104
RADIUS: authenticator BD B2 75 D4 36 9A FE CF - D4 D5 D4 ED 43 A8 4A 34
RADIUS: User-Name [1] 7 "user2"
[ output suppressed]
RADIUS: Received from id 1645/27 172.21.21.250:1812, Access-Accept, len 136
RADIUS: authenticator F3 CD C1 47 F2 76 FB 1B - D5 4C 58 44 07 19 15 DD
RADIUS: Vendor, Cisco [26] 19
RADIUS: Cisco AVpair [1] 13 "priv-lvl=15"
RADIUS: Vendor, Cisco [26] 31
RADIUS: Cisco AVpair [1] 25 "supplicant-group=GROUP2"
RADIUS: Vendor, Cisco [26] 43
RADIUS: Cisco AVpair [1] 37 "proxyacl#1=permit tcp any any eq 22"
[ output suppressed]

FIREWALL sis 84668480: Session Created
FIREWALL sis 84668480: Pak 84184CC4 init_addr (172.21.21.250:1496) resp_addr
(172.16.201.2:23) init_alt_addr (172.21.21.250:1496) resp_alt_addr
(172.16.201.2:23)

! User "user2" starts SSH session after Auth-Proxy (ZFW inspection comes into
play)
FIREWALL* sis 84668680: Session Created
FIREWALL* sis 84668680: Pak 83D1D13C init_addr (172.21.21.250:1500) resp_addr
(172.16.200.200:22) init_alt_addr (172.21.21.250:1500) resp_alt_addr
(172.16.200.200:22)

! Displaying ZFW sessions

DMZ# show policy-map type inspect zone-pair sessions | include Session
    Number of Established Sessions = 2
    Established Sessions
        Session 84668480 (172.21.21.250:1496)=>(172.16.201.2:23) tcp SIS_OPEN
        Session 84668680 (172.21.21.250:1500)=>(172.16.200.200:22) tcp SIS_OPEN

```

Example 14-35 summarizes user-group information for GROUP1 and GROUP2, defined in Example 14-32.

Example 14-35 *Displaying User Group Information*

```

! 02 users connected with 02 different authorization parameters (DACL and AV-
Pair)
DMZ# show user-group
Usergroup : GROUP1
-----
User Name      Type      Interface      Learn      Age (min)
-----
172.21.21.101  IPv4      Vlan21         Dynamic     8

Usergroup : GROUP2
-----
User Name      Type      Interface      Learn      Age (min)
-----
172.21.21.250  IPv4      Vlan21         Dynamic     10
!
DMZ# show epm session ip 172.21.21.250
Admission feature      : Authproxy
AAA Policies           :
Supplicant-Group       : GROUP2
Proxy ACL              : permit tcp any any eq 22
!
DMZ# show epm session ip 172.21.21.101
Admission feature      : Authproxy
AAA Policies           :
ACS ACL               : xACSAClX-IP-DACL1-4aac618d
Supplicant-Group       : GROUP1
!
DMZ# show access-list xACSAClX-IP-DACL1-4aac618d
Extended IP access list xACSAClX-IP-DACL1-4aac618d (per-user)
    10 permit tcp any any eq www
    20 permit icmp any any echo

```


Administrative Access Control on IOS

After studying in detail how to create and enforce identity-based rules for regular users who need to pass traffic *through* firewalls, you can now turn your attention to potential admin users attempting to execute commands on IOS devices. As discussed earlier, TACACS+ is a natural choice for this type of demand because two of its basic attributes (*cmd* and *cmd-arg*) lend themselves well to the challenge of individually authorizing any available command on IOS software.

- Although the selection of the authentication protocol is one essential facet of the problem you need to solve, there are some important aspects to take into account:
- Modern networks are characterized by an increasing number of infrastructure devices (routers, switches, firewalls, Wireless Access Points, and so on) that need to be managed. Scalability is a key topic here.
- It is not typical for companies to have the level of expertise evenly distributed across branches and headquarters. Logical configuration of remote devices is frequently performed from the central site. Some sort of *configuration profile* that could be built once and deployed multiple times for similar devices would be of great value.
- There are many examples of products that integrate several functional domains. Cisco Integrated Service Routers (ISR), for instance, can combine routing, switching, WLAN connectivity, WAN optimization, telephony, and security features, just to name a few. The challenge resides is that most of the time, there are distinct technical teams in charge of each knowledge segment, with different sets of metrics, and so on. How to ensure service integration while still separating teams? Traditional solutions that provide only *all-or-nothing* types of access control do not meet the flexibility requirements of complex environments. Granularity is fundamental.

Cisco Secure ACS implementation of the TACACS+ server portion helps to deal with the issues just raised. By supporting the creation of *Shared Profile Components* that can be applied to any number of user groups, scalability and manageability result. Two kinds of profiles deserve explicit reference:

- **Shell Command Authorization Sets:** Flexible collections of commands that can include any command or, even more specifically, any command argument. Figures 14-9 and 14-10 show two sample command sets that were constructed using different logic. Although the first, named CMD1-Routers enables most commands and denies only specific ones, the second, CMD2-Routers, does the opposite, denying by default and enabling only what was explicitly permitted. It is relevant to point out that in the second command set some arguments of the **show** command were also denied. This provides an illustration of how powerful this resource is.
- **Network Device Groups (NDG):** Sets of AAA Clients (NASes).

These types of CS-ACS profiles can be combined with command sets inside User groups to form a matrix of access privileges similar to that one illustrated in Table 14-1. Notice

that members of one User group may now be assigned a different command set for each NDG. This section examines these concepts by presenting practical usage scenarios.

Permit Unmatched Commands

Shell Command Authorization Set

Name:

Description:

Some "dangerous" commands are denied while the great majority is permitted.

Unmatched Commands: ☐ Permit Unmatched Args

delete
erase
format
reload
rmdir

delete
erase
format
reload
rmdir

Figure 14-9 Sample Shell Command Set That Denies Explicitly Listed Commands

Deny listed arguments Deny Unmatched Commands

Shell Command Authorization Set

Name:

Description:

Unspecified commands are denied. "Show command" is allowed but some arguments may be denied.

Unmatched Commands: ☒ Permit Unmatched Args

show

deny running config
deny privilege
deny users
deny aaa

Show

Figure 14-10 Sample Shell Command Set That Permits Explicitly Defined Commands

Table 14-1 Combining NDGs and Command Sets Inside User Groups

	NDG INTERNET	NDG INTRANET
User Group GROUP1	Command Set CMD1	Command Set CMD2
User Group GROUP2	Command Set CMD3	Command Set CMD1

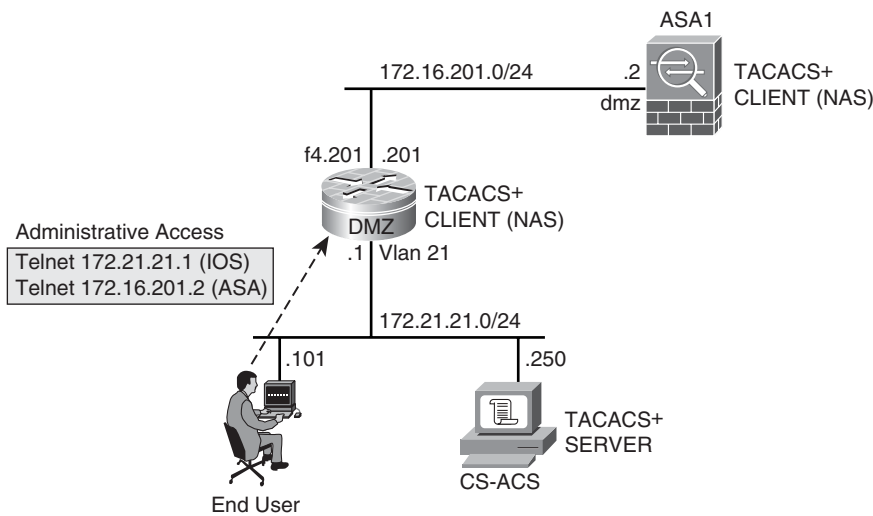


Figure 14-11 Reference Topology for the Analysis of Administrative Access Control

Example 14-36 reinforces the concept that individual command authorization is not supported by RADIUS. This is not a bug. It is just the nature of the protocol.

Example 14-36 Individual Command Authorization Is Not Supported by RADIUS

```
! Defining an AAA Server Group that uses the RADIUS protocol

aaa group server radius RADIUS1
 server 172.21.21.250 auth-port 1812 acct-port 1813
 server-private 172.21.21.250 auth-port 1812 acct-port 1813 key 7
 13061E010803557878
!

! Attempting to configure command authorization using RADIUS

OUT(config)#aaa authorization commands 1 CMD1 group RADIUS1
%AAAA-4-SERVNOTACPLUS: The server-group "RADIUS1" is not a tacacs+ server group.
Please define "RADIUS1" as a tacacs+ server group.
```

Example 14-37 assembles the relevant commands to implement command authorization and accounting on IOS. Notice that IOS enables the usage of named method lists instead of just the *default* method list. This is useful when there is a need to implement different methods for the various types of access (Console, VTY, PPP, and so on).

Tip It is good practice to always use named method lists even if you might have perceived it as more complex in a first contact with the resource. This approach guarantees much more flexibility if you need to add another AAA type of control later. Examples 14-38 through 14-41 illustrate the process of calling method lists into action.

Example 14-37 *Basic Configuration for IOS Command Authorization and Accounting*

```

aaa new-model
!
! Defining an AAA server-group called "TACACS1"

aaa group server tacacs+ TACACS1
  server 172.21.21.250
  server-private 172.21.21.250 key 7 14141B180F0B7B7977
!
! Defining the source interface for TACACS+ packets

ip tacacs source-interface Vlan21
!
! Defining a LOGIN method for the serial console (local authentication for this
line)
aaa authentication login CONSOLE local

! Defining a LOGIN method for the VTY lines (Telnet and SSH)

aaa authentication login TERMINAL group TACACS1
!
! Authorization for EXEC sessions and execution of individual commands

aaa authorization exec EXEC1 group TACACS1
aaa authorization config-commands
aaa authorization commands 1 CMD1 group TACACS1
aaa authorization commands 15 CMD15 group TACACS1
!
! Accounting for EXEC sessions and execution individual commands

aaa accounting exec EXECLOG1 start-stop group TACACS1
aaa accounting commands 1 ACCT1 start-stop group TACACS1
aaa accounting commands 15 ACCT15 start-stop group TACACS1
!

```

```

! Applying the LOGIN authentication and EXEC accounting methods for the console
line
line con 0
  login authentication CONSOLE
  accounting exec EXECLOG1
!
! Applying the LOGIN, Authorization and Accounting named Method Lists to the VTY
lines
line vty 0 4
  login authentication TERMINAL
  authorization exec EXEC1
  authorization commands 1 CMD1
  authorization commands 15 CMD15
  accounting exec EXECLOG1
  accounting commands 1 ACCT1
  accounting commands 15 ACCT15
  transport input telnet ssh

```

Example 14-38 depicts the operation of the named method lists defined in Example 14-37 to control console line access. *EXEC Authorization* is not defined for the console line.

Example 14-39 illustrates access control to a VTY line (via telnet, in this case). In this example, a value of “15” to the privilege-level (priv-lvl=15) is assigned after *EXEC Authorization*.

Example 14-38 Console Session (Local Authentication and TACACS+ Accounting)

```

! Locally defined user "admin" connects to the console line

AAA/BIND(00000017): Bind i/f
AAA/ACCT/EVENT/(00000017): CALL START
Getting session id for NET(00000017) : db=8466336C
AAA/ACCT(00000000): add node, session 43
AAA/ACCT/NET(00000017): add, count 1
Getting session id for NONE(00000017) : db=8466336C
AAA/AUTHEN/LOGIN (00000017): Pick method list 'CONSOLE'
Username: admin
Password:
DMZ>
! Accounting for the EXEC session

AAA/ACCT/EXEC(00000017): Pick method list 'EXECLOG1'
AAA/ACCT/SETMLIST(00000017): Handle ED000006, mlist 84664930, Name EXECLOG1
Getting session id for EXEC(00000017) : db=8466336C
AAA/ACCT(00000017): add common node to avl failed
AAA/ACCT/EXEC(00000017): add, count 2

```

```

AAA/ACCT/EVENT/(00000017): EXEC UP
AAA/ACCT/EXEC(00000017): Queueing record is START
AAA/ACCT(00000017): Accounting method=TACACS1 (TACACS+)
AAA/ACCT/EXEC(00000017): START protocol reply PASS
AAA/ACCT(00000017): Send START accounting notification to EM successfully

```

Example 14-39 Telnet Session - Highlighting EXEC Authorization

```

! Login Authentication, EXEC Authorization and EXEC Accounting succeed for VTY
line
AAA/AUTHEN/LOGIN (00000018): Pick method list 'TERMINAL'
AAA/AUTHOR (0x18): Pick method list 'EXEC1'
AAA/AUTHOR/EXEC(00000018): processing AV cmd=
AAA/AUTHOR/EXEC(00000018): processing AV priv-lvl=15
AAA/AUTHOR/EXEC(00000018): Authorization successful
AAA/ACCT/EXEC(00000018): Pick method list 'EXECLOG1'
AAA/ACCT/SETMLIST(00000018): Handle ED000006, mlist 84664930, Name EXECLOG1
Getting session id for EXEC(00000018) : db=83CABEE4
AAA/ACCT(00000018): add common node to avl failed
AAA/ACCT/EXEC(00000018): add, count 2
AAA/ACCT/EVENT/(00000018): EXEC UP
AAA/ACCT/EXEC(00000018): Queueing record is START
AAA/ACCT(00000018): Accounting method=TACACS1 (TACACS+)
AAA/ACCT/EXEC(00000018): START protocol reply PASS
AAA/ACCT(00000018): Send START accounting notification to EM successfully

```

Example 14-40 illustrates a command authorization session for IOS that is in accordance with the configurations presented in Example 14-37 and with the command set defined in Figure 14-11. In this particular case, the command attempt is denied, therefore not producing an accounting record.

Figure 14-12 shows a sample **Failed Attempts** report in CS-ACS for denied commands. The last column of the table displays the *Device Command Set* that denied the execution of the command.

Example 14-40 Command Authorization Session (Nonauthorized Command)

```

! Following up EXEC authorization, user2 attempts to execute the "show users" com-
mand
! AAA/AUTHOR: auth_need : user= 'user2' ruser= 'DMZ' rem_addr= '172.21.21.101'
priv= 1 list= 'CMD1' AUTHOR-TYPE= 'command'
AAA: parse name=tty2 idb type=-1 tty=-1
AAA: name=tty2 flags=0x11 type=5 shelf=0 slot=0 adapter=0 port=2 channel=0
AAA/MEMORY: create_user (0x8465C574) user='user2' ruser='DMZ' ds0=0
port='tty2' rem_addr='172.21.21.101' authen_type=ASCII service=NONE

```

```
priv=1 initial_task_id='0', vrf= (id=0)
tty2 AAA/AUTHOR/CMD(1549841260): Port='tty2' list='CMD1' service=CMD
AAA/AUTHOR/CMD: tty2(1549841260) user='user2'
tty2 AAA/AUTHOR/CMD(1549841260): send AV service=shell
tty2 AAA/AUTHOR/CMD(1549841260): send AV cmd=show
tty2 AAA/AUTHOR/CMD(1549841260): send AV cmd-arg=users
tty2 AAA/AUTHOR/CMD(1549841260): send AV cmd-arg=<cr>
tty2 AAA/AUTHOR/CMD(1549841260): found list "CMD1"
tty2 AAA/AUTHOR/CMD(1549841260): Method=TACACS1 (tacacs+)
AAA/AUTHOR/TAC+: (1549841260): user=user2
AAA/AUTHOR/TAC+: (1549841260): send AV service=shell
AAA/AUTHOR/TAC+: (1549841260): send AV cmd=show
AAA/AUTHOR/TAC+: (1549841260): send AV cmd-arg=users
AAA/AUTHOR/TAC+: (1549841260): send AV cmd-arg=<cr>
TAC+: (1549841260): received author response status = FAIL
AAA/AUTHOR (1549841260): Post authorization status = FAIL
AAA/MEMORY: free_user (0x8465C574) user='user2' ruser='DMZ'
port='tty2' rem_addr='172.21.21.101' authn_type=ASCII service=NONE
priv=1 vrf= (id=0)
```

Failed Attempts active.csv Refresh Download

Regular Expression Start Date & Time End Date & Time Rows per Page

09/17/2009,21:59:00 09/17/2009,22:13:00 50

Apply Filter Clear Filter

Results from 09/17/2009,21:59:00 to 09/17/2009,22:13:00

Data	Time	User-Name	Group-Name	Caller-ID	Access-Device	NAS-IP-Address	Message-Type	Authn-Failure-Code	Author-Failure-Code	Author-Data	Device-Command-Set
09/17/2009	22:12:59	user2	GROUP2	172.21.21.101	DMZ_TACACS	172.21.21.1	Author failed	..	Command denied	service=shell cmd=show aaa sessions <cr>	CMD2- Routers
09/17/2009	22:12:53	user2	GROUP2	172.21.21.101	DMZ_TACACS	172.21.21.1	Author failed	..	Command denied	service=shell cmd=show users <cr>	CMD2- Routers
09/17/2009	22:12:50	user2	GROUP2	172.21.21.101	DMZ_TACACS	172.21.21.1	Author failed	..	Command denied	service=shell cmd=show privilege <cr>	CMD2- Routers
09/17/2009	22:12:47	user2	GROUP2	172.21.21.101	DMZ_TACACS	172.21.21.1	Author failed	..	Command denied	service=shell cmd=show running-config <cr>	CMD2- Routers
09/17/2009	21:59:45	user1	GROUP1	172.21.21.101	DMZ_TACACS	172.21.21.1	Author failed	..	Command denied	service=shell cmd=reload <cr>	CMD1- Routers
09/17/2009	21:59:07	user1	GROUP1	172.21.21.101	DMZ_TACACS	172.21.21.1	Author failed	..	Command denied	service=shell cmd=format flash: <cr>	CMD1- Routers

Figure 14-12 Sample IOS Command Authorization Failures in CS-ACS (“Failed Attempts”)

Example 14-41 illustrates a command authorization session for IOS that is in accordance with the configurations presented in Example 14-37 and with the command set defined in Figure 14-10.

Figure 14-13 shows a sample TACACS+ Administration report in CS-ACS. An accounting record for the `show ip route` command of example 14-41 is shown in this figure.

Tacacs+ Administration active.csv [Refresh](#) [Download](#)

Regular Expression: Start Date & Time: End Date & Time: Rows per Page:

[Apply Filter](#) [Clear Filter](#)

Results from 09/17/2009,22:04:00

Date ↓	Time	User-Name	Group-Name	Caller-Id	cmd	priv- lvl	Access Device	NAS-IP- Address	NAS- Portname	Network Device Group
09/17/2009	22:13:15	user2	GROUP2	172.21.21.101	show vlan-switch <cr>	1	DMZ_TACACS	172.21.21.1	tty2	IOS_TACACS
09/17/2009	22:13:11	user2	GROUP2	172.21.21.101	show ip route <cr>	1	DMZ_TACACS	172.21.21.1	tty2	IOS_TACACS
09/17/2009	22:13:08	user2	GROUP2	172.21.21.101	show cdp neighbors <cr>	1	DMZ_TACACS	172.21.21.1	tty2	IOS_TACACS
09/17/2009	22:07:12	user1	GROUP1	172.21.21.101	ping 172.21.21.250 <cr>	15	DMZ_TACACS	172.21.21.1	tty2	IOS_TACACS
09/17/2009	22:04:47	user1	GROUP1	172.21.21.101	configure terminal <cr>	15	DMZ_TACACS	172.21.21.1	tty2	IOS_TACACS
09/17/2009	22:04:09	user1	GROUP1	172.21.21.101	show ip route <cr>	1	DMZ_TACACS	172.21.21.1	tty2	IOS_TACACS

Figure 14-13 Sample IOS Command Accounting in CS-ACS (TACACS+ Administration)

Example 14-41 Command Authorization Session (Allowed Command)

```
! Authorized command is issued by user "user1". Command Accounting uses list
"ACCT1"
AAA/AUTHOR: auth_need : user= 'user1' ruser= 'DMZ' rem_addr= '172.21.21.101' priv=
1 list= 'CMD1' AUTHOR-TYPE= 'command'
AAA: parse name=tty2 idb type=-1 tty=-1
AAA: name=tty2 flags=0x11 type=5 shelf=0 slot=0 adapter=0 port=2 channel=0
AAA/MEMORY: create_user (0x84D63A0C) user='user1' ruser='DMZ' ds0=0
port='tty2' rem_addr='172.21.21.101' authen_type=ASCII service=NONE
priv=1 initial_task_id='0', vrf= (id=0)
tty2 AAA/AUTHOR/CMD(3985697951): Port='tty2' list='CMD1' service=CMD
AAA/AUTHOR/CMD: tty2(3985697951) user='user1'
tty2 AAA/AUTHOR/CMD(3985697951): send AV service=shell
tty2 AAA/AUTHOR/CMD(3985697951): send AV cmd=show
tty2 AAA/AUTHOR/CMD(3985697951): send AV cmd-arg=ip
tty2 AAA/AUTHOR/CMD(3985697951): send AV cmd-arg=route
tty2 AAA/AUTHOR/CMD(3985697951): send AV cmd-arg=<cr>
tty2 AAA/AUTHOR/CMD(3985697951): found list "CMD1"
tty2 AAA/AUTHOR/CMD(3985697951): Method=TACACS1 (tacacs+)
AAA/AUTHOR/TAC+: (3985697951): user=user1
AAA/AUTHOR/TAC+: (3985697951): send AV service=shell
AAA/AUTHOR/TAC+: (3985697951): send AV cmd=show
AAA/AUTHOR/TAC+: (3985697951): send AV cmd-arg=ip
AAA/AUTHOR/TAC+: (3985697951): send AV cmd-arg=route
AAA/AUTHOR/TAC+: (3985697951): send AV cmd-arg=<cr>
TAC+: (-309269345): received author response status = PASS_ADD
AAA/AUTHOR (3985697951): Post authorization status = PASS_ADD
AAA/MEMORY: free_user (0x84D63A0C) user='user1' ruser='DMZ'
port='tty2' rem_addr='172.21.21.101' authen_type=ASCII service=NONE
priv=1 vrf= (id=0)
```



```
AAA/ACCT/259(00000018): Pick method list 'ACCT1'
AAA/ACCT/SETMLIST(00000018): Handle 34000007, mlist 846E3288, Name ACCT1
[ output suppressed]
```

Figure 14-14 is intended to consolidate the processes of individual command authorization and accounting that can follow a successful EXEC authorization. In the start point of the flowchart, assume that the user has already been authenticated.

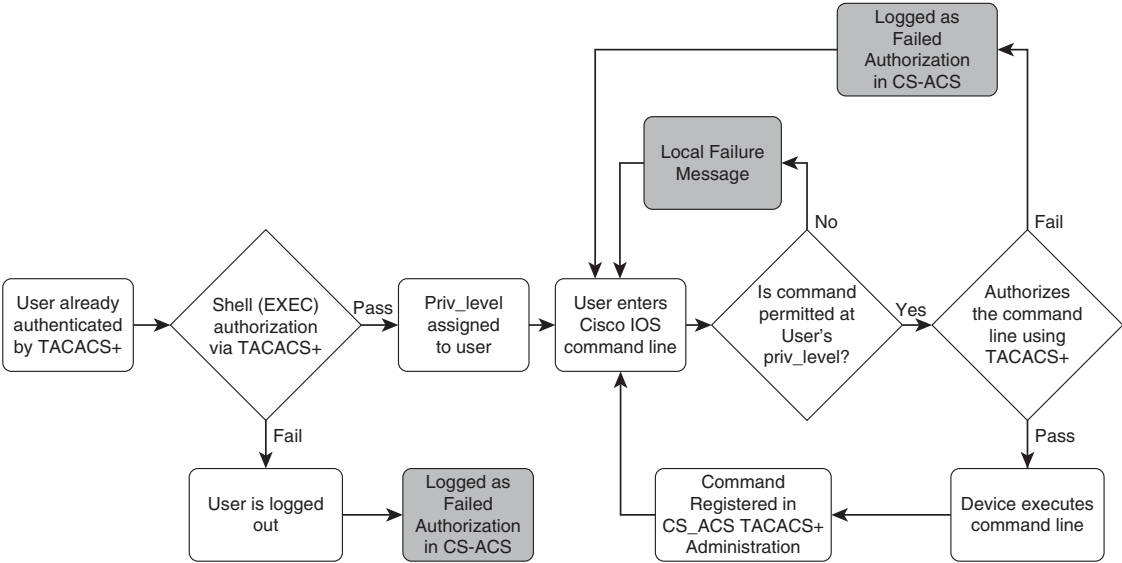


Figure 14-14 Flowchart for IOS TACACS+ Command Authorization and Accounting

Administrative Access Control on ASA

The previous section presented in great level of detail the theory of operations of a flexible and scalable access control architecture that leverages some characteristics of the TACACS+ protocol design and the maturity of Cisco Secure ACS product.

The concepts previously discussed for IOS are still valid for ASA, but some configuration and implementation details are different. The examples analyzed in this section point out the relevant distinctions.

Similarly to what was depicted in Example 14-36 for IOS, Example 14-42 reminds you that individual command authorization is not supported by RADIUS.

Example 14-42 *Individual Command Authorization Is Not Supported by RADIUS*

```
! Verifying the available arguments for the command "aaa authorization command"

ASA1(config)# aaa authorization command ?
configure mode commands/options:
  LOCAL   Predefined server tag for AAA protocol 'local'
  WORD     Specify the name of a TACACS+ aaa-server group to be used for command
authorization
```

Example 14-43 shows the baseline configuration for enabling command authorization and accounting on ASA. It is instructive to compare this scenario with its IOS counterpart in Example 14-37.

Example 14-43 *Basic Configuration for ASA Command Authorization and Accounting*

```
! Defining an AAA server-group called TACACS1

aaa-server TACACS1 protocol tacacs+
aaa-server TACACS1 (dmz) host 172.21.21.250
  key cisco123
!
! Defining the LOGIN authentication method for the console line

aaa authentication serial console LOCAL
!
! Defining TACACS+ as the method for Telnet Authentication

aaa authentication telnet console TACACS1
!
! Defining TACACS+ as the method for Enable Authentication

aaa authentication enable console TACACS1
!
! Defining EXEC session authorization (does not include console line)

aaa authorization exec authentication-server
!
! Accounting, for all the configured forms of access, uses TACACS+

aaa accounting telnet console TACACS1
aaa accounting serial console TACACS1
!
! Defining TACACS+ as the method for command authorization

aaa authorization command TACACS1
!
! Defining TACACS+ as the method for command accounting

aaa accounting command TACACS1
```

Note In IOS, you can assign a value of “15” to the user privilege level (priv-lvl=15) during EXEC authorization. This approach makes it possible for IOS to eliminate the *enable authentication* process and rely on shell command authorization sets for authorizing commands of any level. ASA does not enable the priv-lvl to be directly assigned and does requires the **aaa authentication enable console** command (refer to Example 14-37).

Example 14-44 depicts a command authorization failure for ASA. Authorization failures do not generate accounting records because accounting, strictly speaking, is associated with successful operations (authentication or authorization). Refer to Figure 14-15, which shows a sample log of **Failed Attempts** of command execution in CS-ACS (Reports and Activity session).

Failed Attempts active.csv Refresh Download

Regular Expression Start Date & Time End Date & Time Rows per Page

09/19/2009,12:12:00 mm/dd/yyyy, hh:mm:ss 50

Apply Filter Clear Filter

Results from 09/19/2009,12:12:00

Date	Time	User-Name	Group-Name	Caller-ID	Access-Device	NAS-IP-Address	Message-Type	Authen-Failure-Code	Author-Failure-Code	Author-Data	Device-Command-Set
09/19/2009	12:26:43	user2	GROUP2	172.21.21.101	ASA1_TACACS	172.16.201.2	Author failed	..	Command unknown	service=shell cmd=configure terminal	ASA-Shell2
09/19/2009	12:23:55	user2	GROUP2	172.21.21.101	ASA1_TACACS	172.16.201.2	Author failed	..	Command denied	service=shell cmd=show nameif	ASA-Shell2
09/19/2009	12:23:29	user2	GROUP2	172.21.21.101	ASA1_TACACS	172.16.201.2	Author failed	..	Command denied	service=shell cmd=show route	ASA-Shell2
09/19/2009	12:14:26	user1	GROUP1	172.21.21.101	ASA1_TACACS	172.16.201.2	Author failed	..	Command denied	service=shell cmd=mkdir	ASA-Shell1
09/19/2009	12:12:46	user1	GROUP1	172.21.21.101	ASA1_TACACS	172.16.201.2	Author failed	..	Command denied	service=shell cmd=mkdir	ASA-Shell1

Failed Attempts active.csv Refresh Download

Figure 14-15 Sample ASA Command Authorization Failures in CS-ACS (“Failed Attempts”)

Example 14-44 User Issues Command Not Allowed by Shell Command Set

```
! Command "show route" not authorized

mk_pkt - type: 0x2, session_id: 417
mkpkt - authorize user: user2
cmd=show
cmd-arg=route Tacacs packet sent
Sending TACACS Authorization message. Session id: 417, seq no:1
Received TACACS packet. Session id:1761304772 seq no:2
tacp_procpkt_author: FAIL
TACACS Session finished. Session id: 417, seq no: 1
```

Examples 14-45 and 14-46 contrast ASA behavior about command accounting when a **show** command is issued. Although the **show** command is individually authorized, ASA does not send an accounting message registering it.

Figure 14-16 displays some examples of command accounting for the ASA. Notice the absence of **show** commands in the CS-ACS report.

Tacacs+ Administration active.csv [Refresh](#) [Download](#)

Regular Expression: Start Date & Time: 09/19/2009, 12:21:50 End Date & Time: mm/dd/yyyy, hh:mm:ss Rows per Page: 50

[Apply Filter](#) [Clear Filter](#)

Results from 09/19/2009, 12:21:50

Date	Time	User-Name	Group-Name	Caller-Id	cmd	priv- lvl	Access Device	NAS-IP- Address	NAS- Portname	Network Device Group
09/19/2009	12:38:57	user2	GROUP2	172.21.21.101	ping 172.21.21.1	15	ASA1_TACACS	172.16.201.2	23	ASA_TACACS
09/19/2009	12:35:43	user1	GROUP1	172.21.21.101	configure terminal	15	ASA1_TACACS	172.16.201.2	23	ASA_TACACS
09/19/2009	12:35:29	user1	GROUP1	172.21.21.101	ping 172.21.21.250	15	ASA1_TACACS	172.16.201.2	23	ASA_TACACS
09/19/2009	12:25:08	user2	GROUP2	172.21.21.101	ping 172.21.21.250	15	ASA1_TACACS	172.16.201.2	23	ASA_TACACS
09/19/2009	12:21:56	admin	GROUP1	0.0.0.0	debug tacacs user user2	15	ASA1_TACACS	172.16.201.2	0	ASA_TACACS

Figure 14-16 Sample ASA Command Accounting in CS-ACS (“TACACS+ Administration”)

Example 14-45 User Issues an Authorized “show” Command

```
! Command "show uauth" is authorized. No accounting happens for "show commands"

mk_pkt - type: 0x2, session_id: 421
mkpkt - authorize user: user2
cmd=show
cmd-arg=uauth Tacacs packet sent
Sending TACACS Authorization message. Session id: 421, seq no:1
Received TACACS packet. Session id:1409549404 seq no:2
tacp_procpkt_author: PASS_ADD
tacp_procpkt_author: PASS_REPL
TACACS Session finished. Session id: 421, seq no: 1
```

Example 14-46 User Issues Authorized Command (not a “show” command)

```
! User issues "ping 172.21.21.1" (successful authorization and accounting)

mk_pkt - type: 0x2, session_id: 419
mkpkt - authorize user: user2
cmd=ping
cmd-arg=172.21.21.1 Tacacs packet sent
Sending TACACS Authorization message. Session id: 419, seq no:1
Received TACACS packet. Session id:998474355 seq no:2
tacp_procpkt_author: PASS_ADD
```

```

tacp_procpkt_author: PASS_REPL
TACACS Session finished. Session id: 419, seq no: 1
!
mk_pkt - type: 0x3, session_id: 420
mkpkt - accounting username: user2
remote ip : 172.21.21.101 task_id=40
Tacacs packet sent
Sending TACACS Accounting message. Session id: 420, seq no:1
Received TACACS packet. Session id:15914798 seq no:2
TACACS Session finished. Session id: 420, seq no: 1

```

Summary

In this chapter, the main concepts associated with Identity are revisited and a whole set of features that enable security policies to be built on the basis of user information are analyzed. This becomes increasingly relevant on dynamic network environments that involve different classes of users (such as employees, contractors, and guests).

The AAA architecture stands for Authentication, Authorization, and Accounting and defines a modular way for these three security functions to interact. The AAA components may be easier visualized if associated with the questions they were designed to answer:

- **Authentication:** Deals with answering the question “Who is the user?”
- **Authorization:** Is in charge of defining, “What is the user (previously authenticated) allowed to do?”
- **Accounting:** Relates to the question, “What did the user do?” Through this process, the accounting client collects user activity information and sends it to the accounting server.

The AAA framework is applied throughout the chapter to provide identity-based control on two complementary domains:

- Control of regular users that need to pass traffic through the firewall: The mechanisms of Cut-through Proxy (on ASA family) and Auth-Proxy (on IOS) materialize this functionality.
- Control of administrative users required to configure and monitor the devices themselves.

Without any intention of making a value judgment, the two classic authentication protocols, RADIUS and TACACS+, were employed to face these two categories of challenges. The theoretical discussion and the companion examples explored in this chapter make it clear that RADIUS fits well for controlling access of regular users to network services, and that TACACS+ is the natural choice for administrative access control.

Cut-through proxy is inherently stateful, whereas Auth-Proxy needs to be complemented with other IOS Firewall resources to produce a stateful user-based behavior. The stateful effect derives, in this last case, from the combination with either CBAC (Classic IOS firewall) or Zone-based policy firewall.

This page intentionally left blank

Firewalls and IP Multicast

This chapter covers the following topics:

- Review of Multicast Addressing
- Overview of Multicast Routing and Forwarding
- Multicast Routing with PIM
- Inserting ASA in a Multicast Routing Environment

“A man who listens because he has nothing to say can hardly be a source of inspiration...”—Agnes Repplier

Multicasting is concerned with sending data from a source unicast address to a selected set of receivers identified by a *Group Address*. To accomplish that, there should be processes to deal with tasks like group creation, signaling the interest of hosts in participating of a certain group, and routing multicast packets from the transmitting source toward the network nodes that have connected receivers for that group. There should also be means to tell the multicast routers that all receivers of a certain group have already left and, therefore, that the associated resources for maintaining state information for that group can be freed. Further, the ability to locally deliver traffic solely to the LAN switch ports that have connected receivers (avoiding the unnecessary generation of broadcasts in that segment) is another important issue pertaining to Multicast theory.

Detailing the topics aforementioned and other aspects related to advanced multicast design and deployment would certainly consume hundreds of pages, which is naturally out of the scope of this book. The purpose of the discussion that follows is simply building basic awareness about multicast theory of operations and the issues that arise when multicast packets need to traverse firewalls.

The relevant concepts for achieving these goals are reviewed on an as-needed basis, always using as a reference the behavior of Protocol Independent Multicast (PIM), most of the times in Sparse Mode (PIM-SM) and, for some specific situations, in Dense Mode (PIM-DM).

Review of Multicast Addressing

Following the original classful conception of IPv4, the IANA assigned the *Class D* space (224.0.0.0/4) to represent Multicast Group Addresses. One noteworthy characteristic of Class D is the absence of the host and network portions that exist for Classes A, B, and C. A *multicast address* is defined by the combination of the fixed value “1110” for the initial 4 bits of the first octet and a 28-bit long Group ID, as illustrated in Figure 15-1.

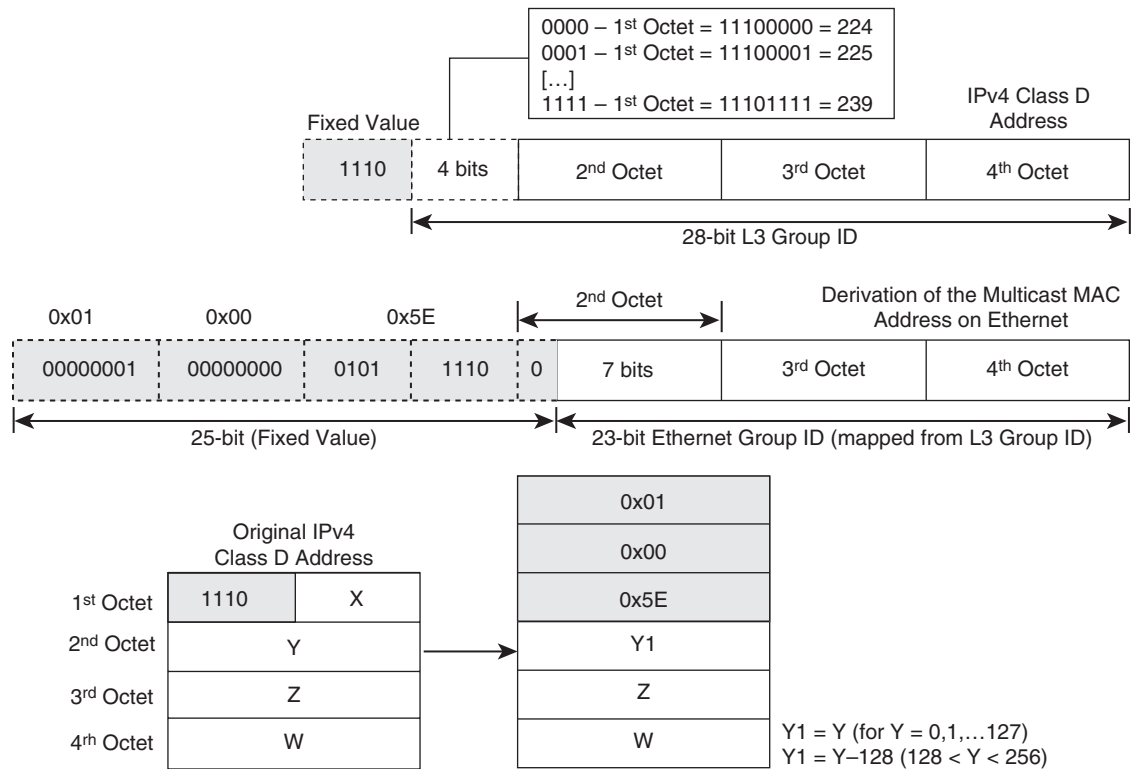


Figure 15-1 Overview of IPv4 Multicast Addressing

Among the 2^{28} possible IPv4 addresses in Class D, some have special meaning:

- **224.0.0.0/8:** Reserved for routing protocols and network maintenance tasks.
- **224.0.0.0/24:** This range defines the *Link-Local Multicast Addresses*, meaning that a packet containing such an address for its destination should be constrained to the local network segment and not be forwarded by multicast routers. Some well-known addresses in this range follow:
 - **224.0.0.1:** All hosts in the subnet
 - **224.0.0.2:** All routers in the subnet

- 224.0.0.5: OSPF routers
- 224.0.0.6: OSPF Designated Routers (DR)
- 224.0.0.9: RIPv2 routers
- 224.0.0.10: EIGRP routers
- 224.0.0.13: Protocol Independent Multicast (PIM) routers
- 224.0.0.18: Virtual Router Redundancy Protocol (VRRP)
- 232.0.0.0/8: reserved for Source Specific Multicast (SSM).
- 239.0.0.0/8: known as the administratively scoped multicast addresses. This range plays a role similar to that of private addresses in the sense of RFC 1918 (such as 10.0.0.0/8).

Figure 15-1 not only documents the structure of a Layer 3 multicast address but also demonstrates how to obtain the Ethernet multicast address from the L3 Group ID. Given that the *Ethernet Group ID* is 23-bits long, there is a 2^5 (= 32) overlapping between the two domains. For instance:

- 224.1.1.1, 225.1.1.1, 226.1.1.1,..., 239.1.1.1 all map to the same Layer 2 multicast address, 0100.5E01.0101
- 239.2.3.1 and 239.130.3.1 map to 0100.5E02.0301

Understanding the conversion process is critical for proper group address planning. This avoids the undesirable situation of multiple groups being mapped to the same Ethernet multicast address in a LAN segment that contains receivers for overlapping groups.

Overview of Multicast Routing and Forwarding

As studied in Chapter 5, “Firewalls in the Network Topology,” IP unicast routing is destination-oriented. End hosts typically rely on a default gateway for Layer 3 connectivity purposes. These gateways are in charge of building and maintaining routing (and forwarding) tables, to help hosts on the task of reaching other destinations throughout the internetwork (using the best possible path, from the standpoint of the routing protocols being used).

The typical tasks associated with multicast routing are summarized as follows:

- Multicast groups employed for delivering traffic related to a given multicast application are advertised to potential users. Hosts throughout the internetwork signal to multicast routers their intention to receive the traffic for one of the available groups by *joining* that specific group. The Internet Group Membership Protocol (IGMP) takes care of this interaction between hosts and routers.
- A multicast routing protocol builds a Multicast Distribution Tree (MDT). The MDT defines the path followed by multicast packets throughout the network.

- A source host running a multicast application starts sending packets to a multicast group. This traffic is delivered to interested receivers by means of the appropriate MDT.

Note The IGMP Snooping solution is frequently used to ensure that multicast traffic on a LAN segment is only delivered to switch ports that have connected receivers. This feature is normally supported on modern L2 switches but for performance considerations, it is advisable that the switch under consideration have a hardware dedicated to this function.

An important characteristic of multicasting is that the source does not need to waste CPU cycles on replication tasks. This is a responsibility of the multicast routing infrastructure in place. The source simply sends a single packet to a multicast group address, and the routers along the MDT forward this packet over the interfaces that lead to receivers for that particular group.

Assuming that the MDT has already been constructed, a multicast router that is near the source receives a packet on an incoming interface and typically sends copies of it on multiple outgoing interfaces. Each multicast router receiving this packet replicates it on all outgoing interfaces leading to receivers that have joined the group.

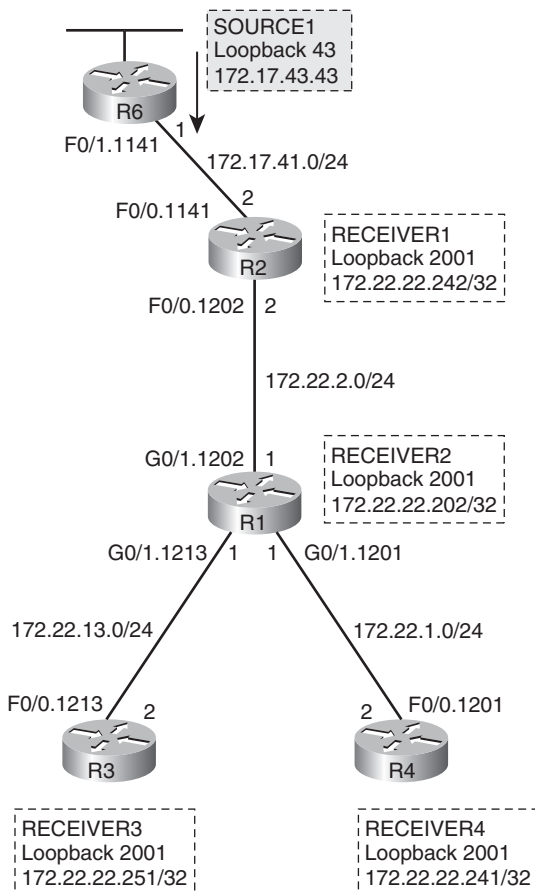
The packet replication work that is inherent to multicast forwarding might be challenging, from a network design-perspective. For instance, in the event a routing loop is formed, it might happen that forwarded packets get injected back to the incoming interface, therefore causing extra replication and, eventually, contributing to a multicast storm (which would be more devastating than an unicast storm).

Instead of using the destination-oriented approach that characterizes unicast routing, IP multicast routers look back toward the sources of multicast traffic to achieve the foundational goal of avoiding routing loops. This source-focused paradigm is referred to as Reverse Path Forwarding (RPF) and is examined, in more detail, later in this section.

The Concept of Upstream and Downstream Interfaces

In the context of multicast routing, the terms *upstream* and *downstream* are often used to establish the relative positioning of router interfaces along the path from a certain source to a particular receiver. This terminology is helpful for stating a key rule for the delivery of multicast traffic: Packets should always be sent downstream from the transmitting source toward the receivers. (And just to emphasize, *never upstream to the source*). For instance, in the scenario in Figure 15-2, R1's interfaces might be classified as follows:

- G0/1.1202 is the upstream interface for group 224.0.1.40 toward Source1 (172.17.43.43). This interface appears in the multicast routing table as the *incoming interface*.
- Loopback 2001 has joined group 224.0.1.40 and, therefore, is a directly connected downstream interface for Source1.
- G0/1.1213 and G0/1.1201 are downstream interfaces for Source1 toward group 224.0.1.40 because they are in the path to reach Receiver3 and Receiver4.
- The downstream interfaces for group 224.0.1.40 (Loopback 2001, G0/1.1213, and G0/1.1201) populate the *Outgoing Interface List (OIL)* of the multicast routing table.



```
R6# show ip mroute 172.17.43.43 224.0.1.40 | begin \
(172.17.43.43, 224.0.1.40), 2d03h/00:02:11, flags: LT
Incoming interface: Loopback43, RPF nbr 0.0.0.0
Outgoing interface list:
FastEthernet0/1.1141, Forward/Sparse-Dense, 2d03h/00:00:00
```

```
R2# show ip mroute 172.17.43.43 224.0.1.40 | begin \
(172.17.43.43, 224.0.1.40), 2d03h/00:02:50, flags: LT
Incoming interface: FastEthernet0/0.1141, RPF nbr 172.17.41.1
Outgoing interface list:
Loopback2001, Forward/Sparse-Dense, 2d03h/00:00:00
FastEthernet0/0.1202, Forward/Sparse-Dense, 2d03h/00:00:00
```

```
R1# show ip mroute 172.17.43.43 224.0.1.40 | begin \
(172.17.43.43, 224.0.1.40), 2d03h/00:02:17, flags: LT
Incoming interface: GigabitEthernet0/1.1202, RPF nbr 172.22.2.2
Outgoing interface list:
Loopback2001, Forward/Sparse-Dense, 2d03h/00:00:00
GigabitEthernet0/1.1213, Forward/Sparse-Dense, 2d03h/00:00:00
GigabitEthernet0/1.1201, Forward/Sparse-Dense, 2d03h/00:00:00
```

```
R3# show ip mroute 172.17.43.43 224.0.1.40 | begin \
(172.17.43.43, 224.0.1.40), 2d03h/00:02:52, flags: LT
Incoming interface: FastEthernet0/0.1213, RPF nbr 172.22.13.1
Outgoing interface list:
Loopback2001, Forward/Sparse-Dense, 2d03h/00:00:00
```

```
R4# show ip mroute 172.17.43.43 224.0.1.40 | begin \
(172.17.43.43, 224.0.1.40), 2d03h/00:02:39, flags: LT
Incoming interface: FastEthernet0/0.1201, RPF nbr 172.22.1.1
Outgoing interface list:
Loopback2001, Forward/Sparse-Dense, 2d03h/stopped
```

Figure 15-2 Reference Topology for RPF Analysis

Note The source reference varies from one multicast routing protocol to another. For example, a PIM-DM upstream interface points toward the actual source of traffic for a

group. PIM-SM, on the other hand, establishes the upstream interface looking toward a central entity called the Rendezvous Point (RP).

RPF Interfaces and the RPF Check

The concepts of upstream and downstream interfaces are used in this section to define Reverse Path Forwarding (RPF) and analyze its application to the multicast forwarding process. This study takes as reference the topologies shown in Figures 15-2 and 15-3, which were built with the aid of PIM Dense Mode (PIM-DM).

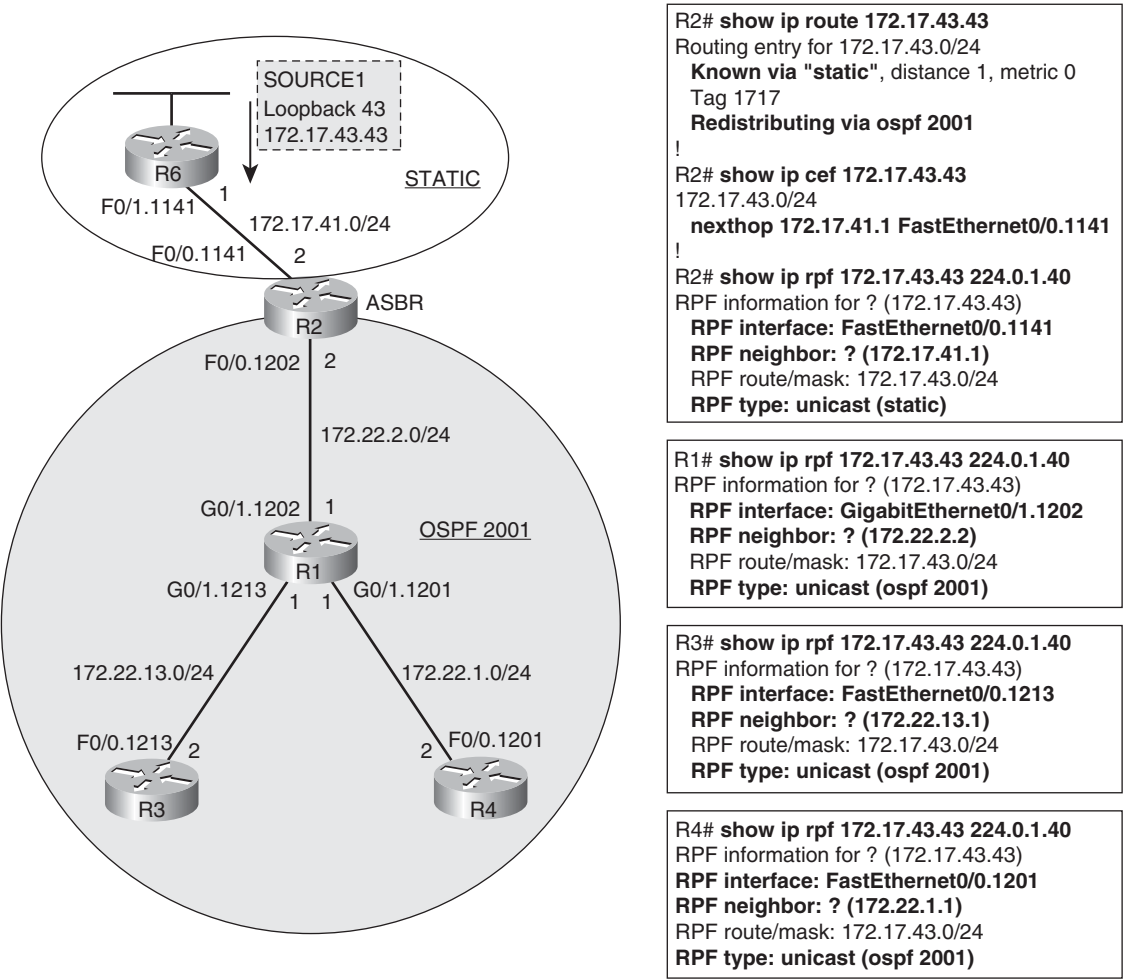


Figure 15-3 Obtaining RPF Information via the Unicast Routing Table

The **show ip mroute** commands that appear in Figure 15-2 include the RPF neighbor (*RPF nbr*) information, which corresponds, for an Interior Gateway Protocol (IGP), to the next-hop address used by the multicast router to reach the source (via the upstream interface). Figure 15-3 details how to define the RPF interface for the multicast network depicted in Figure 15-2 and highlights that PIM can employ a combination of unicast routing protocols to interconnect sources and receivers. In this particular environment, there is an OSPF process in place, which redistributes static routes in such a way, that hosts on the 172.22.0.0/16 range can reach the external networks represented by 172.17.0.0/16, and conversely. R2 acts as the OSPF Autonomous System Boundary Router (ASBR) in this scenario.

After the RPF interface is calculated for a Source-Group (S,G) pair, the RPF check is executed to determine whether an arriving multicast packet should be forwarded or dropped. The RPF check can be briefly described as follows:

- The multicast router verifies if the source address of the arriving packet is reachable via the upstream interface toward the source.
- Packets arriving through the upstream interface succeed the RPF verification and are forwarded.
- Packets that fail the RPF verification are discarded.

Note The strict unicast RPF (uRPF) verification technique, used in Chapter 11, “Additional Protection Mechanisms,” for antispoofing purposes, is one of the possible ways to define the RPF interface. All the examples in this section employ this method as the source of RPF information.

The RPF verification performed by multicast routing protocols is mainly focused on determining a loop free path to a certain source *S* that is transmitting to a group *G*. Another relevant task of a multicast routing protocol is to define the downstream interfaces leading to those subnets that contain receivers for a given group *G*.

The union of the upstream interface to the source, with the downstream interfaces toward the receivers (within every router along the path), constitute a Multicast Distribution Tree (MDT). There are two main categories of MDTs:

- **Source Trees or Shortest Path Trees (SPT):** This type of tree is rooted at the multicast source for a group and uses the shortest path from source to receiver. A separate MDT exists for each (S,G) pair. A sample topology employing this MDT construction is shown in Figure 15-4. For simplicity on computing the shortest path, assume that all links depicted have a cost of 10 units. PIM Dense Mode is an example of multicast routing protocol that uses Source Trees.
- **Shared Trees (or RP Trees):** Rather than placing its root on a particular source, a Shared Tree is rooted on a somewhat centralized place of the multicast topology. For PIM-SM, this point is known as the *Rendezvous Point (RP)*. Figure 15-5 shows an

environment for which R3 has been chosen as the RP. Multicast packets from source SRC1 to group G are sent to the RP (R3) and then downstream from the RP toward the receivers.

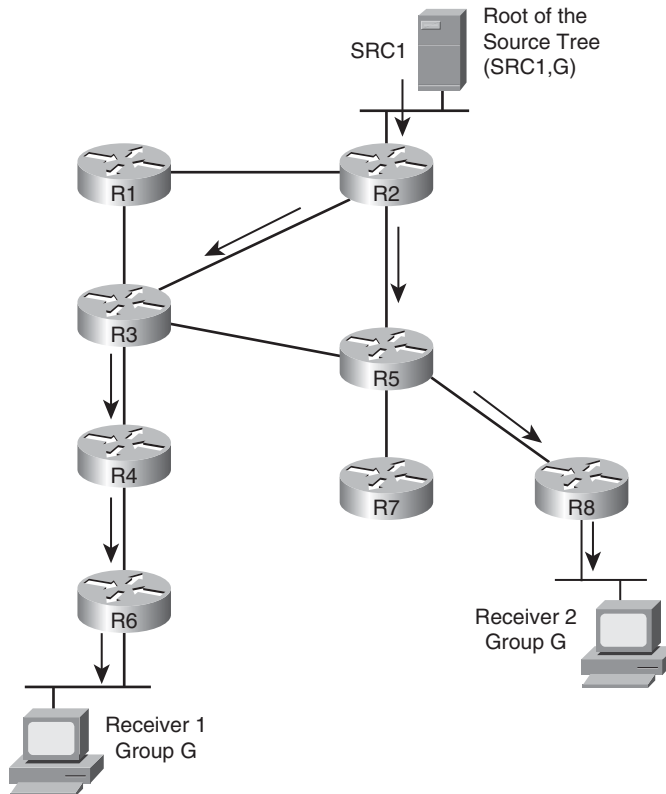


Figure 15-4 *Sample Multicast Network Using a Source Tree*

Multicast Routing with PIM

Protocol Independent Multicast (PIM) is the protocol of choice for implementing real-life multicast networks. Two of its key advantages follow:

- As the name implies, PIM is independent of the underlying unicast routing protocol employed to build Layer 3 connectivity. PIM can even use a combination of routing protocols to derive RPF information, as previously shown in Figure 15-3.
- As opposed to protocols such as Distance Vector Multicast Routing Protocol (DVMRP) and Multicast OSPF (MOSPF), PIM does not maintain its own multicast routing table. It does not need, for instance, to send or receive updates about multicast routes. This results in less overhead and more scalability.

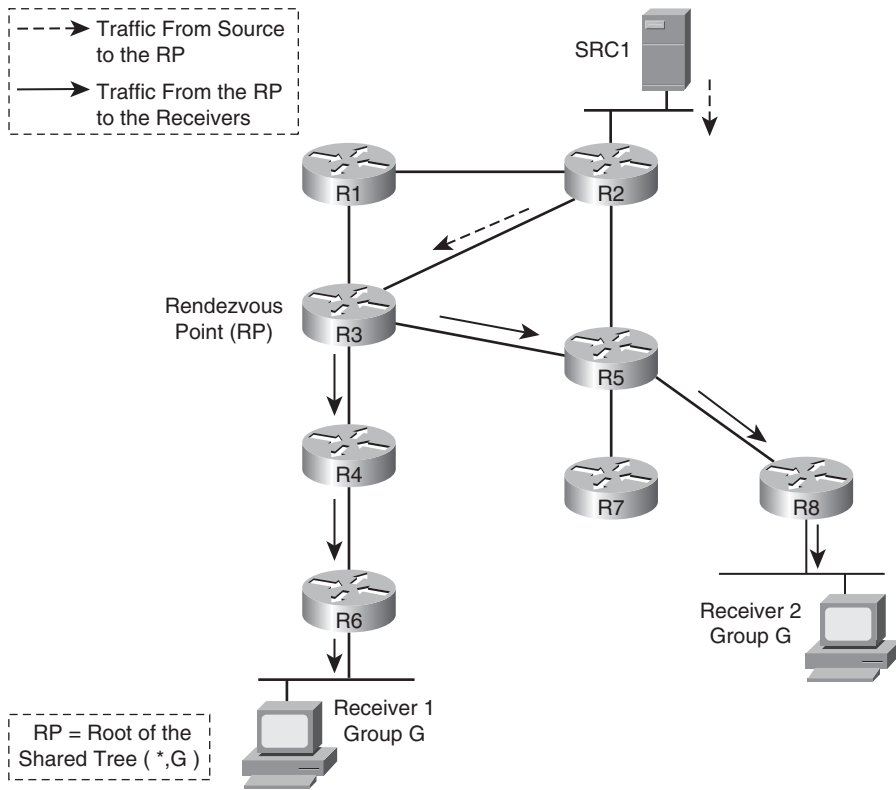


Figure 15-5 Sample Multicast Network Using a Shared Tree

The two classic operation modes for PIM are Dense Mode (PIM-DM) and Sparse Mode (PIM-SM). Newer developments provide options known as PIM-SSM (Source Specific Multicast) and bidirectional PIM, which are not covered in this introductory chapter.

Note Cisco routers support all the PIM operation modes (according to the IOS image installed). ASA appliances can participate in a PIM-SM topology or operate in a mode known as Stub Multicast Routing (SMR). These options are illustrated later in the chapter.

Enabling PIM on Cisco Routers

For the discussion in the following subsections, it is always assumed that the involved routers have already been configured to route multicast packets. The basic steps to achieve this are listed in the following:

- The command `ip multicast-routing` is enabled globally.

- A PIM mode of operation must be defined to all router interfaces that are used as a path to sources or receivers of multicast traffic. This is accomplished with the **ip pim** command, which supports three basic options:
 - **ip pim dense-mode**: Instructs the interface to operate in PIM Dense Mode.
 - **ip pim sparse-mode**: Configures the interface for Sparse Mode operation. As you will study later, some extra work related to Rendezvous Point (RP) selection is needed in this case.
 - **ip pim sparse-dense-mode**: This a flexible option that enables the interface to use Sparse Mode (for groups that have an associated RP) or Dense Mode, if RP information is not available for a given group.
- The **ip pim neighbor-filter** command can be optionally configured to specify the acceptable PIM neighbors for each multicast-enabled interface.

Although enabling PIM on Cisco routers is a straightforward task, you should invest some time to understand basic multicast theory before doing so. This important fact of life was the motivation to include a larger theoretical review before talking about firewall placement or forwarding rules.

Note Before multicast routing is enabled on a Cisco router, packets sent to L3 group addresses are dropped, as characterized by the **show ip cef** command.

```
IOS-FW# show ip cef | include 224.
224.0.0.0/4          drop
224.0.0.0/24        receive
```

This behavior is changed by the **ip multicast-routing** command:

```
IOS-FW# show ip cef | include 224.
224.0.0.0/4          multicast
224.0.0.0/24        receive
```

Note The Cisco multicast routers used in the examples presented in this chapter had all their pertinent interfaces configured with the **ip pim sparse-dense-mode** command.

PIM-DM Basics

PIM-DM relies on Source Trees to distribute multicast traffic to receivers. The Source-based MDT for a (S,G) pair is dynamically built as soon as a source *S* begins sending packets to a group *G*.

PIM-DM assumes that any given subnet in an internetwork will contain at least one receiver for a certain (S,G) combination and uses the *flood-and-prune* mechanism as its traffic distribution method. When a source begins transmitting to a group, it initially

floods all the links between any multicast routers in the network. After that, *pruning* comes into play (on a per-interface basis) to signal that a router wants to stop receiving traffic for a given group on a particular link.

The left side of Figure 15-6 shows the initial flooding process that characterizes the construction of a source-based MDT on a PIM-DM network. The right side shows the resultant topology, after some pruning rules have been applied, and registers the RPF interface leading to the source SRC1 or each router. The pruning sequence is briefly described as follows:

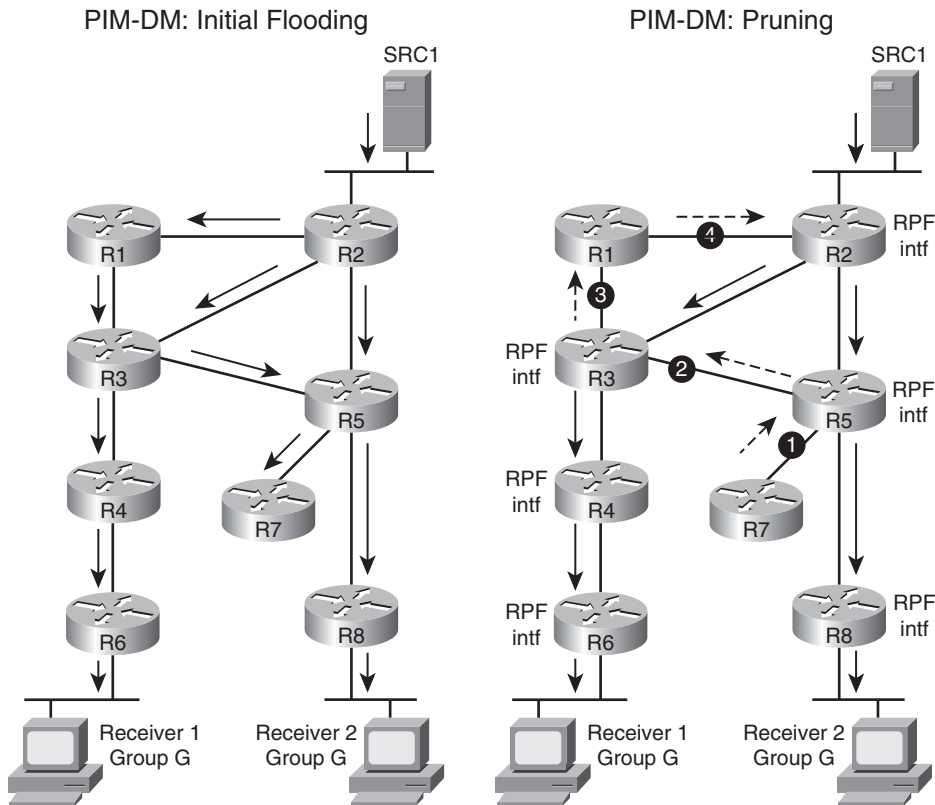


Figure 15-6 Overview of Flooding and Pruning Processes for PIM-DM

- R7, being a *leaf router* with no directly connected receivers, prunes link 1. R5 prunes back its link to R7.
- R5 prunes link 2 because this interface to R3 is a non-RPF point-to-point link. R3 then prunes this direct link to R5.
- R3 prunes link 3 toward R1, for it is a non-RPF point-to-point interface with respect to source SRC1.

- R1, having no directly connected receivers, prunes link 4 as a result of receiving the prune from its single downstream neighbor (R3) over link 3.

The MDT derived after pruning in Figure 15-6 corresponds to the source-based tree previously shown in Figure 15-4.

Some other noteworthy facts pertaining to PIM-DM operation are summarized in the following:

- Every router in the network holds (S,G) entries in their **mroute** tables for all active sources. This is true, even for those groups without any single receiver, for which there is a source transmitting.
- In a PIM-DM network, the arrival of multicast traffic on a router creates (S,G) entries. Even though (*,G) entries are not used for multicast forwarding by PIM-DM, these parental entries are always created first by Cisco routers, whenever there is a need to insert a new (S,G) entry.
- The Outgoing Interface List of a generic (*,G) PIM-DM entry includes interfaces to which receivers are connected or in which other PIM-DM neighbors reside. Although the (*,G) information is always present on the **mroute** tables of Cisco routers, this type of data is much more meaningful for PIM-SM operation. The examples dealing with PIM-DM always focus on (S,G) entries.

The need to maintain state on every router for all the possible (S,G) streams in a network, and the bandwidth waste caused by the flood-and-prune technique, makes PIM-DM a less suitable option for use across WAN links or in large networks. The *Explicit Join* approach employed by PIM-SM renders it a more scalable and reliable option, when compared with PIM-DM, for almost all practical deployments.

Note As a practical example of PIM-DM operation, consider the topology portrayed in Figure 15-2. This consists in a source-based MDT built with PIM-DM for the (S,G) pair (172.17.43.43, 224.0.1.40). The reserved group 224.0.1.40 is known as the *Cisco-RP-Discovery* group and is described later. Figure 15-2 also serves to illustrate the **mroute** table from the standpoint of each participating router.

PIM-SM Basics

The most important similarity between PIM-SM and PIM-DM resides in their way of computing the RPF interface. As seen earlier, the default method employed by PIM, for the sake of obtaining RPF information, is to consult the unicast routing table. Apart from that, these two protocols are different in matters concerning state creation and traffic distribution.

PIM-SM is based upon an Explicit Join model, which establishes that multicast traffic should be sent only to hosts that formally request it. In this model, a host signals its intent to participate in group G by sending an IGMP membership report over the LAN

subnet to which it is attached. A PIM-SM router connected to that segment, understands the IGMP message, generates a corresponding PIM Join Message for the parental group (*,G) and then sends it to the pertinent Rendezvous Point (RP). This PIM Join Message is forwarded hop-by-hop, upstream to the RP, using as its destination the *PIM Routers* Group address, 224.0.0.13.

If a host considers that multicast traffic is no longer needed for a certain group G, it sends an IGMP Leave message in its local subnet. Upon receipt of such an IGMP message from the last receiver in the subnet, the PIM-SM router responsible for that segment (*PIM Designated Router*) sends a Prune message to the RP. This message is forwarded hop-by-hop, upstream to the RP, in a totally analogous fashion to the Join message. PIM-SM routers along the path to the RP, update the (*,G) entries in their *mroute* tables accordingly.

Note For PIM-SM, the term upstream is used to mean the RPF interface toward the RP (rather than to any specific source of multicast traffic).

Figure 15-7 brings an overview of the Explicit Join model on a PIM-SM network in which router R3 has been chosen as the RP. The analysis uses as a reference the host Receiver2, which has R8 as its PIM Designated Router (the only router in a multi-access network containing an active receiver that is allowed to send PIM Joins to the RP). The basic steps associated with the Explicit Join process for the scenario in Figure 15-7 follow:

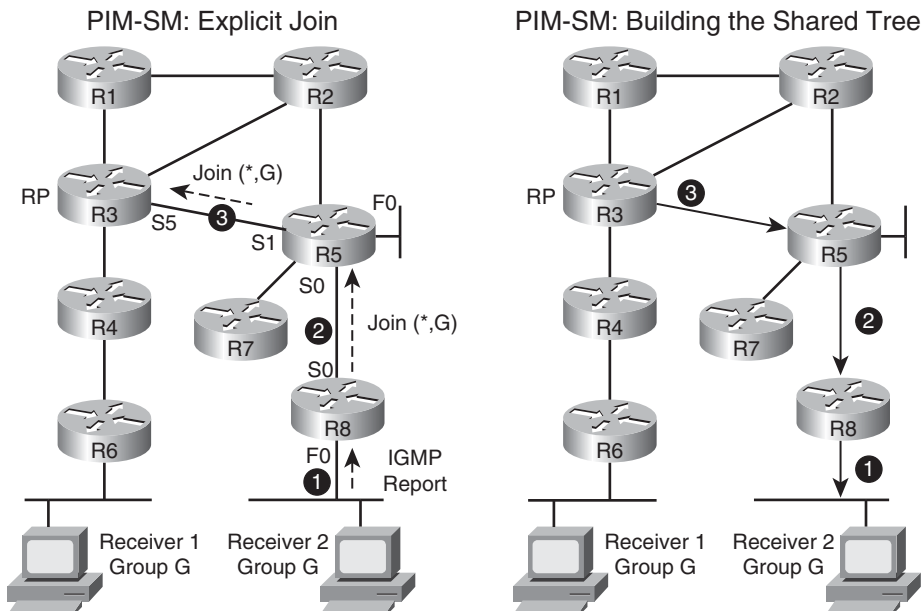


Figure 15-7 Overview of the Explicit Join Model for PIM-SM

- 1. Receiver2 sends an IGMP Membership Report on its LAN segment. Upon detecting the presence of this first receiver in its F0 segment, R8 creates an (*,G) entry in its **mroute** table and sends a PIM (*,G) Join up the Shared Tree. R8's interface F0 is inserted in the Outgoing Interface List (OIL) for (*,G).
- 2. R5 receives the (*,G) Join, and because it did not know of any other receivers within group G, it creates the (*,G) entry. Interface S0 is placed in the *OIL* for this group and R5 forwards the (*,G) Join message over its upstream interface to the RP.
- 3. R3 (the RP servicing group G) receives the Join forwarded by R5 and creates the (*,G) state accordingly. (In this network, R3 had no previous information about group G receivers). R3's interface S5 is added to the OIL for (*,G). After this step, the shared tree for group G has been built from R3 (RP) to Receiver2. This means that any traffic bound to group G arriving at the RP can now flow down the shared tree towards Receiver2.

After a (*,G) state is created on a router, it can be reused to add newer receivers to the shared tree on a later stage. For instance, in the event a new member of Group G is seen on R5's interface F0, R5 simply includes this interface in the OIL associated with the (*,G) entry. There is no change in the RP entry for (*,G), for it already had R5 as a branch on its shared tree associated with group G.

Example 15-1 is used with the scenario shown in Figure 15-8 to bring more details about the PIM-SM Explicit Join procedure. In the figure, the pertinent entries representing (*,G) states are shown in all the routers throughout the path from the RP to the receiver called IGMP-HOST1. Other aspects that deserve special mention follow:

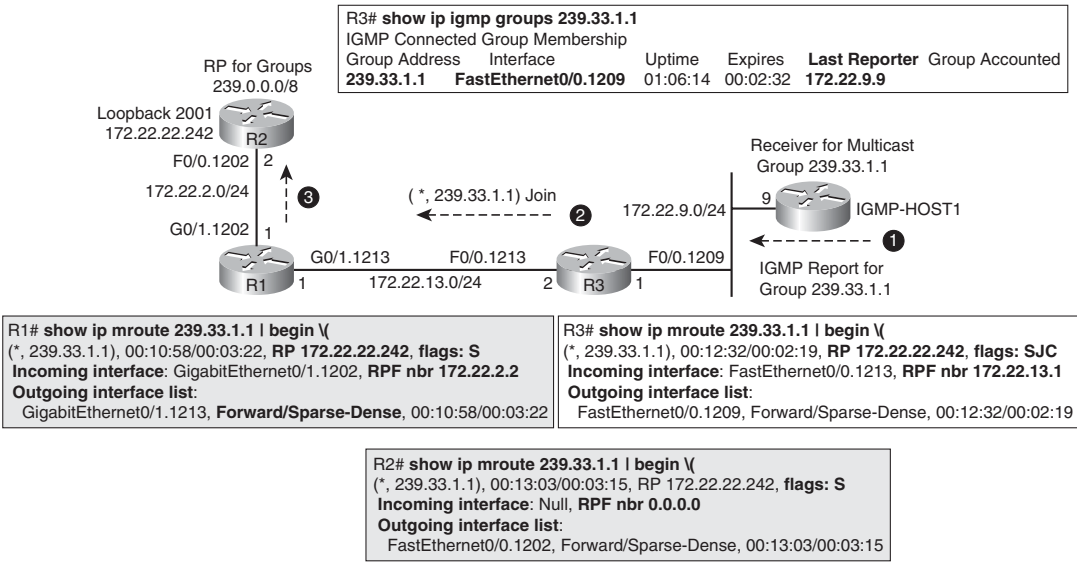


Figure 15-8 More details About the Explicit Join Process

- R3 is the PIM-SM Designated Router for subnet 172.22.9.0/24 and, as such, is in charge of listening to IGMP Reports on this segment and sending the corresponding PIM Join messages to the RP. Because R3 is a router that has a directly connected member of group 239.33.1.1, it is frequently referred to as a *last-hop router* for this group.
- The Flag S in the **mroute** table indicates that the entry has been created in Sparse Mode.
- The incoming interface in each router corresponds to the RPF interface toward the RP. R2, the RP for this topology, sees its incoming interface as *Null*.
- The wildcard bit (WC-bit) assumes the value 1 to indicate that this Join is intended to the Shared Tree (any source matches the wildcard). The RP-Tree bit (RPT-bit) of 1 shows that the message is traveling through the Shared Tree toward the RP.

Example 15-1 PIM-SM Join Process After a Receiver Joins an IGMP Group

```
! The IGMP host sends an IGMP Membership Report on f0/0.1209
IGMP-HOST1(config)#interface f0/0.1209
IGMP-HOST1(config-subif)#ip igmp join-group 239.33.1.1
IGMP(8): WAVL Insert group: 239.33.1.1 interface: FastEthernet0/0.1209 Successful
IGMP(8): Send v2 Report for 239.33.1.1 on FastEthernet0/0.1209

! R3's perspective (last-hop router)
PIM(0): Check RP 172.22.22.242 into the (*, 239.33.1.1) entry
PIM(0): Building Triggered (*,G) Join/ (S,G,RP-bit) Prune message for 239.33.1.1
PIM(0): Insert (*,239.33.1.1) join in nbr 172.22.13.1's queue
PIM(0): Building Join/Prune packet for nbr 172.22.13.1
PIM(0): Adding v2 (172.22.22.242/32, 239.33.1.1), WC-bit, RPT-bit, S-bit Join
PIM(0): Send v2 join/prune to 172.22.13.1 (FastEthernet0/0.1213)

! R1's perspective (PIM-SM router on R3's path towards the RP)
PIM(3): Received v2 Join/Prune on GigabitEthernet0/1.1213 from 172.22.13.2, to us
PIM(3): Join-list: (*, 239.33.1.1), RPT-bit set, WC-bit set, S-bit set
PIM(3): Check RP 172.22.22.242 into the (*, 239.33.1.1) entry
PIM(3): Add GigabitEthernet0/1.1213/172.22.13.2 to (*, 239.33.1.1), Forward
state, by
PIM *G Join
PIM(3): Building Triggered (*,G) Join / (S,G,RP-bit) Prune message for 239.33.1.1
PIM(3): Insert (*,239.33.1.1) join in nbr 172.22.2.2's queue
PIM(3): Building Join/Prune packet for nbr 172.22.2.2
PIM(3): Adding v2 (172.22.22.242/32, 239.33.1.1), WC-bit, RPT-bit, S-bit Join
PIM(3): Send v2 join/prune to 172.22.2.2 (GigabitEthernet0/1.1202)

! What is seen by R2 (RP)
PIM(0): Received v2 Join/Prune on FastEthernet0/0.1202 from 172.22.2.1, to us
PIM(0): Join-list: (*, 239.33.1.1), RPT-bit set, WC-bit set, S-bit set
```

```
PIM(0): Check RP 172.22.22.242 into the (*, 239.33.1.1) entry
PIM(0): Add FastEthernet0/0.1202/172.22.2.1 to (*, 239.33.1.1), Forward state, by
PIM *G Join
```

PIM-SM relies on an unidirectional shared tree over which multicast traffic flows, from the RP, down to the receivers. Given that the actual challenge is to connect a source to the receivers of a certain group, there should be some means for this source to inform the RP about its intent to send traffic to a particular group, without the need to join it (as a receiver).

By using the PIM source registration process, a source can send multicast packets to the RP, which in turn forwards them to the receivers down the Shared Tree. The basic steps involved in the source registration procedure for a scenario in which Receiver2 has already joined the (*,G) tree, are represented on the left side of Figure 15-9 and briefly described in the following:

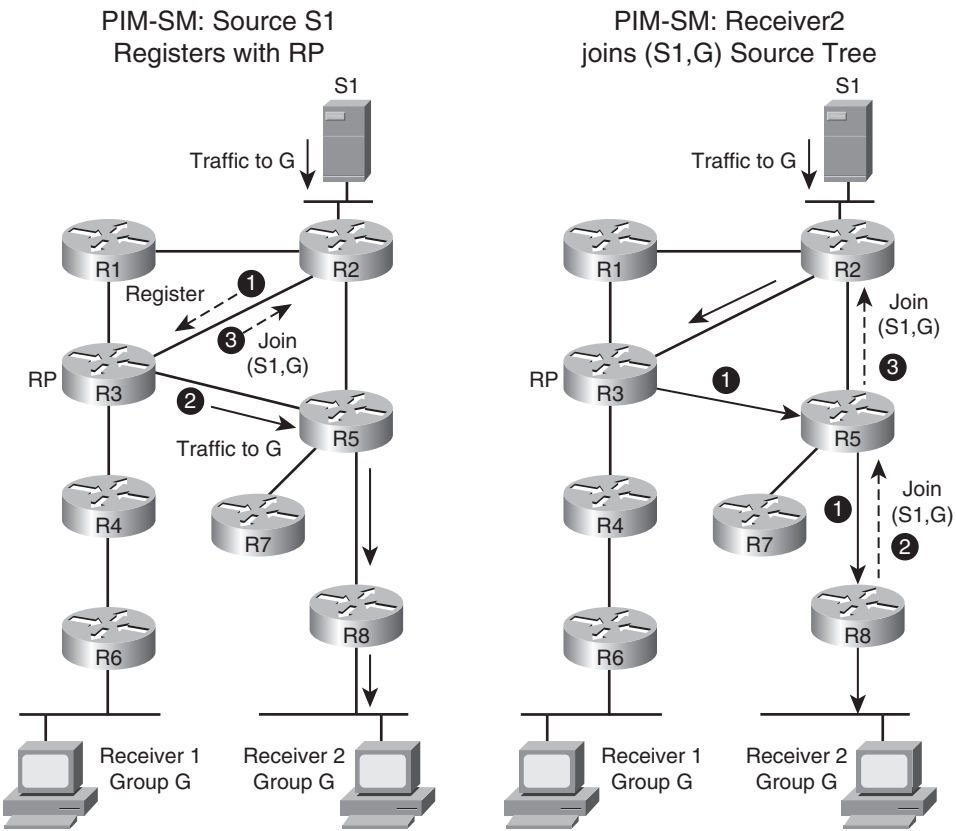


Figure 15-9 Overview of Source Registering and SPT Join Processes for PIM-SM

1. Source S1, directly connected to router R2, starts sending multicast traffic to group G. R2, the first-hop router for S1, encapsulates the received packets in PIM Register messages and unicasts them to the RP.
2. Upon receipt of the PIM Register message, the RP (R3) decapsulates it and sends it to group G over the interfaces in its Outgoing Interface List (*OIL*).
3. The RP multicasts an (S1,G) Join back to S1 so that it can attract the traffic sent by this source.
4. After successfully joining the Source Tree (S1,G), the RP concludes that there is no need to keep encapsulating multicast packets in PIM Register messages. The RP then unicasts a Register-Stop message to the first-hop router R2. (For simplicity, this step is not shown in Figure 15-9).

One classic doubt concerning PIM-SM design relates to the intuitive idea that multicast traffic always flows through the RP. This is not true. PIM-SM enables *last-hop routers* (routers with directly connected receivers for a group G) to explicitly join a Source Tree, using a process similar to a regular Shared Tree (SPT) join. This possibility results in optimal routing of multicast traffic from sources down to receivers and keeps the RP from becoming a bottleneck. The Source Tree Join procedure (SPT Join) for PIM-SM is briefly illustrated on the right side of Figure 15-9:

1. R8, the last-hop router for Receiver2 for Group G, starts receiving traffic from source S1, to group G, via the Shared Tree.
2. R8 computes the RPF interface to S1 (using its unicast routing table) and sends a (S1,G) Join over this interface.
3. R5 receives the (S1,G) Join from R8, creates the (*,G) and (S1,G) states, and includes its downstream link to R8 on the OIL for these entries. R5 then sends a (S1,G) Join using its RPF interface to S1.

Note Source registration typically precedes the process of a receiver joining the SPT on a PIM-SM network.

Examples 15-2 and 15-3 refer to the topology represented in Figure 15-10. They were conceived to provide more details about the source registration and SPT Join processes.

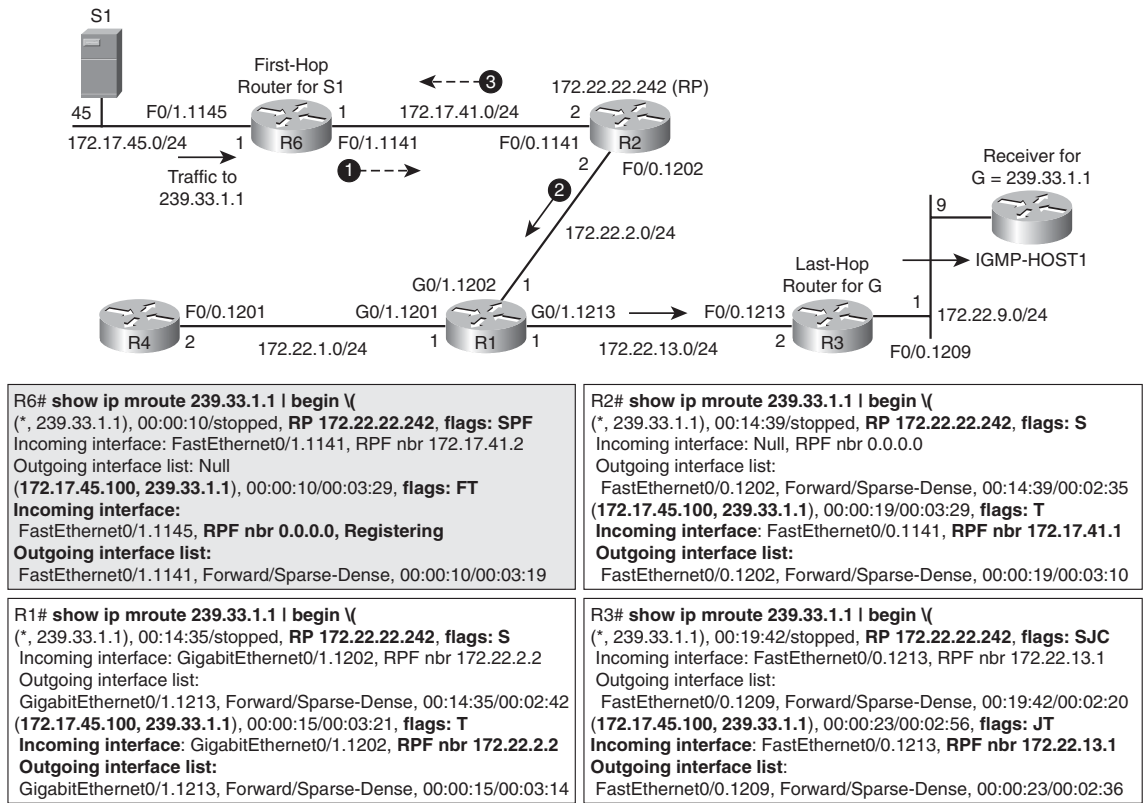


Figure 15-10 Details About the Source Registering Process on a PIM-SM Environment

Example 15-2 highlights the message exchange between R6 (first-hop router for source S1) and R2 (RP). It is interesting to compare the PIM Join Message in this example (SPT Join from RP to source) with that of Example 15-1: (*,G) from the last-hop router up to the RP (shared tree join).

Example 15-3 shows an SPT Join from R3 (PIM-SM Designated Router for subnet 172.22.9.0/24) to source 172.17.45.45. In this specific network, the RP is on the shortest path from R3 to R6. The SPT Join on the last segment (from the RP to R6) had already been sent during the source registration process in Example 15-2.

Example 15-2 Source Registering on a PIM-SM Network

```
! R6 Perspective (first-hop router for source S1)
PIM(0): Check RP 172.22.22.242 into the (*, 239.33.1.1) entry
PIM(0): Send v2 Register to 172.22.22.242 for 172.17.45.100, group 239.33.1.1
PIM(0): Received v2 Join/Prune on FastEthernet0/1.1141 from 172.17.41.2, to us
PIM(0): Join-list: (172.17.45.100/32, 239.33.1.1), S-bit set
PIM(0): Add FastEthernet0/1.1141/172.17.41.2 to (172.17.45.100, 239.33.1.1),
```

Forwardstate, by PIM SG Join

```

PIM(0): Send v2 Register to 172.22.22.242 for 172.17.45.100, group 239.33.1.1
PIM(0): Received v2 Register-Stop on FastEthernet0/1.1141 from 172.22.22.242
PIM(0):   for source 172.17.45.100, group 239.33.1.1
PIM(0): Clear Registering flag to 172.22.22.242 for (172.17.45.100/32, 239.33.1.1)
!
! Source registration process as seen by R2 (RP)

PIM(0): Received v2 Register on FastEthernet0/0.1141 from 172.17.41.1
        for 172.17.45.100, group 239.33.1.1
PIM(0): Insert (172.17.45.100,239.33.1.1) join in nbr 172.17.41.1's queue
PIM(0): Forward decapsulated data packet for 239.33.1.1 on FastEthernet0/0.1202
PIM(0): Building Join/Prune packet for nbr 172.17.41.1
PIM(0):   Adding v2 (172.17.45.100/32, 239.33.1.1), S-bit Join
PIM(0): Send v2 join/prune to 172.17.41.1 (FastEthernet0/0.1141)
PIM(0): Received v2 Register on FastEthernet0/0.1141 from 172.17.41.1
        for 172.17.45.100, group 239.33.1.1
PIM(0): Send v2 Register-Stop to 172.17.41.1 for 172.17.45.100, group 239.33.1.1

```

Example 15-3 *A Receiver (Already in the Shared Tree) Joins the SPT*

```

! R3's perspective : this last-hop router multicasts the (S1,G) join to R1
PIM(0): Insert (172.17.45.100,239.33.1.1) join in nbr 172.22.13.1's queue
PIM(0): Building Join/Prune packet for nbr 172.22.13.1
PIM(0):   Adding v2 (172.17.45.100/32, 239.33.1.1), S-bit Join
PIM(0): Send v2 join/prune to 172.22.13.1 (FastEthernet0/0.1213)
!
! R1's perspective (intermediate router on the path from Receiver2 to S1)

PIM(1): Received v2 Join/Prune on GigabitEthernet0/1.1213 from 172.22.13.2, to us
PIM(1): Join-list: (172.17.45.100/32, 239.33.1.1), S-bit set
PIM(1): Add GigabitEthernet0/1.1213/172.22.13.2 to (172.17.45.100, 239.33.1.1),
Forward state, by PIM SG Join
PIM(1): Insert (172.17.45.100,239.33.1.1) join in nbr 172.22.2.2's queue
PIM(1): Building Join/Prune packet for nbr 172.22.2.2
PIM(1): Adding v2 (172.17.45.100/32, 239.33.1.1), S-bit Join
PIM(1): Send v2 join/prune to 172.22.2.2 (GigabitEthernet0/1.1202)
!
! R2's perspective ( given that it is own the shortest path from Receiver2 to S1)
PIM(0): Received v2 Join/Prune on FastEthernet0/0.1202 from 172.22.2.1, to us
PIM(0): Join-list: (172.17.45.100/32, 239.33.1.1), S-bit set
PIM(0): Update FastEthernet0/0.1202/172.22.2.1 to (172.17.45.100, 239.33.1.1),
Forward state, by PIM SG Join

```

Note The `ip pim accept-register` global configuration command can be deployed on a candidate RP to keep undesired sources from registering with this RP. If the first-hop router connected to such a source sends a PIM Register message to the RP, the RP immediately sends back a Register-Stop message. This is a control-plane feature that complements regular data-plane ACLs applied to the ingress interfaces of first-hop routers.

Figure 15-11 shows a PIM-SM scenario in which a source S3 started sending multicast packets before any receiver had joined the Shared Tree. Some relevant aspects related with this type of situation are listed here:

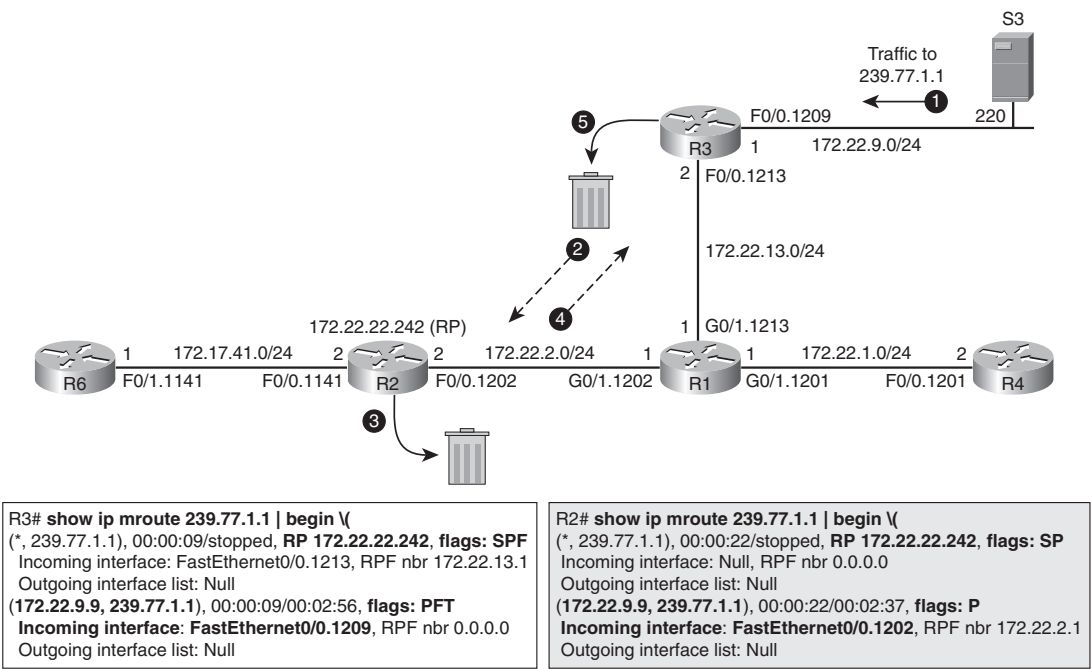


Figure 15-11 PIM-SM Scenario for Which a Source Registers Before Receivers Join the Group

1. Source S3 (172.22.9.220) begins sending traffic to Group 239.77.1.1
2. After receiving traffic bound to group G, from S3, R3 (which is the first-hop router for this source) unicasts PIM Registers messages to R2 (RP).
3. Given that the RP has no receivers for group G in the Shared Tree, it discards the received packets.

4. The RP unicasts a Register-Stop message back to R3.
5. R3 stops encapsulating multicast packets in Register messages and drops subsequent traffic sent by the source S3.

Example 15-5 complements Example 15-4 by showing what happens when a receiver connected to R6 joins the Shared Tree:

1. R6 sends a (*,G) Join to the RP.
2. The RP sends a (S3,G) Join toward the source S3. The RP would have sent as many (S,G) joins as there were sources in the topology.
3. The RP receives source traffic along the SPT it has just built to R3.
4. The RP forwards traffic received over the (S3,G) MDT down its Shared Tree to the receivers that have joined group 239.77.1.1

Example 15-4 PIM-SM Scenario with a Multicast Source Registering First

```
! R3's perspective (first hop router with respect to source S3)
PIM(0): Check RP 172.22.22.242 into the (*, 239.77.1.1) entry
PIM(0): Send v2 Register to 172.22.22.242 for 172.22.9.220, group 239.77.1.1
PIM(0): Received v2 Register-Stop on FastEthernet0/0.1213 from 172.22.22.242
PIM(0):   for source 172.22.9.220, group 239.77.1.1
PIM(0): Clear Registering flag to 172.22.22.242 for (172.22.9.220/32, 239.77.1.1)
!
! What R2 (RP) sees initially (no receiver)
PIM(0): Received v2 Register on FastEthernet0/0.1202 from 172.22.13.2
        for 172.22.9.220, group 239.77.1.1
PIM(0): Check RP 172.22.22.242 into the (*, 239.77.1.1) entry
PIM(0): Send v2 Register-Stop to 172.22.13.2 for 172.22.9.220, group 239.77.1.1
```

Example 15-5 Receiver Joins Shared Tree After Source Has Registered

```
! RP receives a Shared Tree Join (PIM *G Join) from R6
PIM(0): Received v2 Join/Prune on FastEthernet0/0.1141 from 172.17.41.1, to us
PIM(0): Join-list: (*, 239.77.1.1), RPT-bit set, WC-bit set, S-bit set
PIM(0): Add FastEthernet0/0.1141/172.17.41.1 to (*, 239.77.1.1), Forward state,
by PIM *G Join
PIM(0): Add FastEthernet0/0.1141/172.17.41.1 to (172.22.9.220, 239.77.1.1),
Forward state, by PIM *G Join
!
! RP sends SPT-Join to source 172.22.9.220 (connected to R3)
PIM(0): Insert (172.22.9.220,239.77.1.1) join in nbr 172.22.2.1's queue
PIM(0): Building Join/Prune packet for nbr 172.22.2.1
PIM(0): Adding v2 (172.22.9.220/32, 239.77.1.1), S-bit Join
PIM(0): Send v2 join/prune to 172.22.2.1 (FastEthernet0/0.1202)
```

Finding the Rendezvous Point on PIM-SM Topologies

PIM-SM topologies must include at least one RP. The two methods for making RP information available to Cisco IOS PIM-SM routers are summarized in the following:

- **Static RP definition:** At least one RP is manually defined on every PIM-SM router within the topology (including the RP itself). This is accomplished using **ip pim rp-address** global configuration commands. There can be various RPs in the network, each of them being the root for a set of multicast groups whose traffic needs to be distributed. Static RP is the only method supported by ASA appliances.
- **Cisco Auto-RP mechanism:** This dynamic mapping technique eliminates the need to manually define RP information in all the PIM-SM routers within the topology. Cisco Auto-RP defines two special types of routers, namely RP Mapping Agents and Candidate RPs, which respectively transmit information to the reserved multicast groups 224.0.1.40 (*Cisco-RP-Discovery*) and 224.0.1.39 (*Cisco-RP-Announce*). These two entities work in a collaborative way to provide PIM-SM routers with group-to-RP mapping information.

Note The PIMv2 Bootstrap Router (BSR) mechanism represents a standards-based method to dynamically distribute group-to-RP mapping information. You can find the description of this technique in RFC 2362 or in Cisco IOS documentation.

- The basic operation of Auto-RP can be summarized as follows:
 - Cisco routers configured for PIM automatically join group 224.0.1.40 so that they can be provided with the group-to-RP mapping information that is periodically advertised by RP Mapping Agents.
 - RP Mapping Agents join the group 224.0.1.39 to receive the periodic announcements generated by Candidate RPs. Mapping Agents send RP Discovery messages to group 224.0.1.40 to share the contents of its group-to-RP mapping cache with all PIM-SM routers in the domain.

Example 15-6 refers to the scenario represented in Figure 15-12 and assembles important information about Auto-RP:

- R6 is instructed to act as the RP Mapping Agent by means of the command **ip pim send-rp-discovery**. Given that there are no Candidate RPs originally defined, R6 concludes that there are no mappings to send. The **show ip pim rp mapping** command reveals that R6 is acting as a Mapping Agent.
- R2 is selected to be the Candidate RP for groups in the range 239.0.0.0/8. (This group restriction is accomplished by attaching an ACL to the **ip pim send-rp-announce** command.)
- R2 starts sending RP-Announce packets destined to group (224.0.1.39) out the interfaces shown in Figure 15-12. These RP-Announces arrive at the Mapping Agent,

which in turn builds group-to-RP mappings and advertises them via RP-Discovery packets directed to group 224.0.1.40. All Cisco PIM routers automatically join this group and, therefore, they can receive the mapping updates sourced from R6.

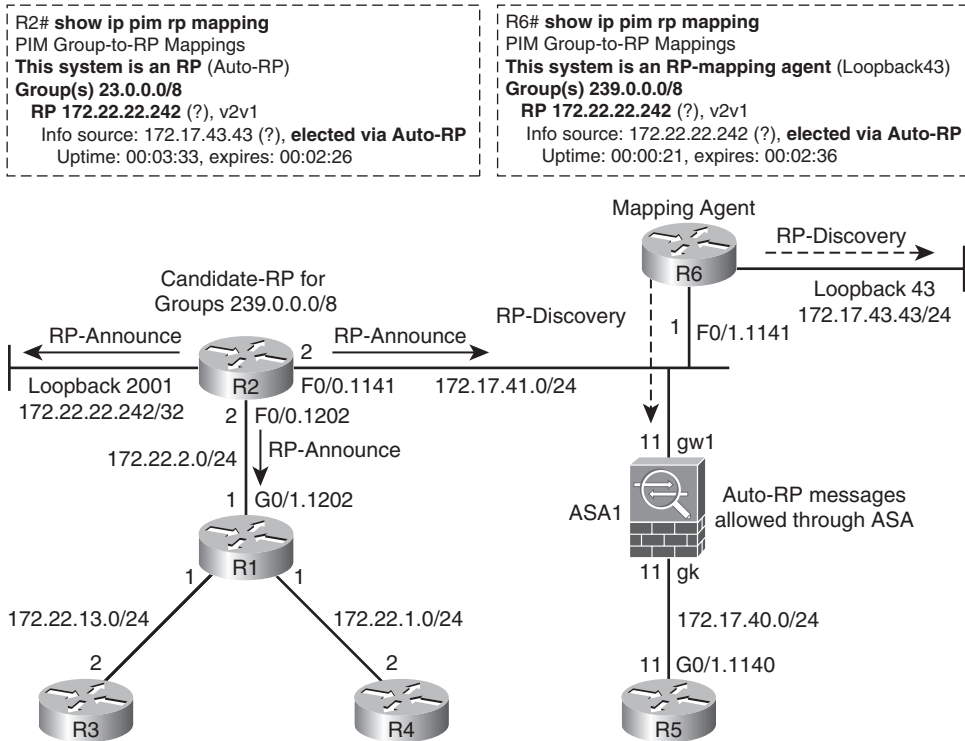


Figure 15-12 *Auto-RP Overview*

- It is always recommended you use a loopback interface as the source of RP-Announce and RP-Discovery packets.
- For successful Auto-RP deployment, it is emphatically recommended that all router interfaces be configured with the **ip pim sparse-dense-mode** command. When this interface mode is in place, the two reserved Auto-RP groups can be created in dense mode and have the control information they provide flooded throughout the network, without depending on any previous RP definition. The other groups (used for actual distribution of multicast application data) are created in Sparse Mode.
- Figure 15-2, in the beginning of this chapter, presents **mroute** information for the entry (172.17.43.43, 224.0.1.40), which corresponds to the Mapping Agent R6 transmitting, in dense mode, to the RP Discovery group 224.0.1.40.

Although Auto-RP messages relate with PIM-SM, the messages themselves are not PIM-based. The Auto-RP protocol uses UDP as its transport, as multicast applications typically do. The service port reserved for Auto-RP is UDP/496, as shown in Example 15-7. This

example also shows that R5 (a router behind ASA in the topology of Figure 15-12) successfully receives the Auto-RP mapping information (because ASA has been configured to enable this type of traffic through).

Note It is common to use the same Cisco router to perform the Mapping Agent and RP functions. The decision of not using this option in the examples is twofold:

- Better illustrating operation of the individual elements.
- Explore filtering options (data-plane or control-plane based) that can end up suggesting placement of these two special routing entities, mainly for large deployments.

Example 15-6 Basic Information About Auto-RP Messages

```
! Instructing R6 to act as an RP Mapping Agent
R6(config)# ip pim send-rp-discovery Loopback43 scope 16
!
! Initially, no RP Candidates are configured for Auto-RP in the network
Auto-RP(0): Build RP-Discovery packet
Auto-RP: no RP periodic mappings to send
!
R6# show ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP-mapping agent (Loopback43)
!
! Configuring R2 as a Candidate-RP for the group range 239.0.0.0/8

R2(config)# access-list 40 permit 239.0.0.0 0.255.255.255
R2(config)# ip pim send-rp-announce Loopback2001 scope 16 group-list 40
Auto-RP(0): Build RP-Announce for 172.22.22.242, PIMv2/v1, ttl 16, ht 181
Auto-RP(0): Build announce entry for (239.0.0.0/8)
Auto-RP(0): Send RP-Announce packet of length 48 on FastEthernet0/0.1202
Auto-RP(0): Send RP-Announce packet of length 48 on FastEthernet0/0.1141
Auto-RP(0): Send RP-Announce packet of length 48 on Loopback2001(*)
Auto-RP(0): Received RP-discovery packet of length 48, from 172.17.43.43,
RP_cnt 1, ht 181
Auto-RP(0): Added with (239.0.0.0/8, RP:172.22.22.242), PIMv2 v1
!
R2# show ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP (Auto-RP)
Group(s) 239.0.0.0/8
  RP 172.22.22.242 (?), v2v1
    Info source: 172.17.43.43 (?), elected via Auto-RP
      Uptime: 00:03:33, expires: 00:02:26
```

```

!
! Following receipt of RP-Announces, the Mapping Agent (R6) sends RP Discovery
messages
R6# Auto-RP(0): Received RP-announce packet of length 48, from 172.22.22.242,
RP_cnt 1, ht 181
Auto-RP(0): Added with (239.0.0.0/8, RP:172.22.22.242), PIMv2 v1
Auto-RP(0): Build RP-Discovery packet
Auto-RP: Build mapping(239.0.0.0/8, RP:172.22.22.242), PIMv2 v1,
Auto-RP(0): Send RP-discovery packet of length 48 on FastEthernet0/1.1141
(1 RP entries)
Auto-RP(0): Send RP-discovery packet of length 48 on Loopback43(*) (1 RP entries)
!
R6# show ip pim rp mapping
PIM Group-to-RP Mappings
This system is an RP-mapping agent (Loopback43)
Group(s) 239.0.0.0/8
  RP 172.22.22.242 (?), v2v1
    Info source: 172.22.22.242 (?), elected via Auto-RP
    Uptime: 00:00:21, expires: 00:02:36

```

Note When using a single RP for all the groups, you need to remember that the Auto-RP reserved groups should be excluded from the **group-list** associated with that RP, as shown here:

```
access-list 50 deny host 224.0.1.39
```

```
access-list 50 deny host 224.0.1.40
```

```
access-list 50 permit any
```

```
ip pim send-rp-announce Loopback2001 scope 16 group-list 50
```

The **deny** statements in the **access-list** are used to characterize that the RP service is not used for these groups. The groups operate in dense mode.

Example 15-7 *Auto-RP Messages Through ASA*

```

! ASA is configured to allow Auto-RP messages through (RP-Announce and RP-
Discovery)
access-list GW1 extended permit udp host 172.17.43.43 host 224.0.1.40 eq pim-auto-
rp
access-list GW1 extended permit udp host 172.22.22.242 host 224.0.1.39 eq pim-
auto-rp
!
%ASA-6-302015: Built inbound UDP connection 250546 for
gw1:172.22.22.242/496 (172.22.22.242/496) to identity:224.0.1.39/496
(224.0.1.39/496)
%ASA-6-302015: Built inbound UDP connection 250549 for gw1:172.17.43.43/496
(172.17.43.43/496) to identity:224.0.1.40/496 (224.0.1.40/496)
!
! Sample view of the PIM topology for the Auto-RP groups
ASA1# show pim topology 224.0.1.39
IP PIM Multicast Topology Table
Entry state: (*S,G)[RPT/SPT] Protocol Uptime Info
Entry flags: KAT - Keep Alive Timer, AA - Assume Alive, PA - Probe Alive,
    RA - Really Alive, LH - Last Hop, DSS - Don't Signal Sources,
    RR - Register Received, SR - Sending Registers, E - MSDP External,
    DCC - Don't Check Connected
Interface state: Name, Uptime, Fwd, Info
Interface flags: LI - Local Interest, LD - Local Disinterest,
    II - Internal Interest, ID - Internal Disinterest,
    LH - Last Hop, AS - Assert, AB - Admin Boundary
(*,224.0.1.39) DM Up: 5d06h RP: 0.0.0.0
JP: Null(never) RPF: ,0.0.0.0 Flags: LH DSS
    gw1                5d06h        off LI LH
(172.22.22.242,224.0.1.39)SPT DM Up: 2d23h
JP: Null(never) RPF: gw1,172.17.41.2 Flags: KAT(00:03:16)
    gk                2d23h        fwd
    gw1                2d23h        fwd
!
ASA1# show pim topology 224.0.1.40
IP PIM Multicast Topology Table
Entry state: (*S,G)[RPT/SPT] Protocol Uptime Info
Entry flags: KAT - Keep Alive Timer, AA - Assume Alive, PA - Probe Alive,
    RA - Really Alive, LH - Last Hop, DSS - Don't Signal Sources,
    RR - Register Received, SR - Sending Registers, E - MSDP External,
    DCC - Don't Check Connected
Interface state: Name, Uptime, Fwd, Info
Interface flags: LI - Local Interest, LD - Local Disinterest,
    II - Internal Interest, ID - Internal Disinterest,

```

```

    LH - Last Hop, AS - Assert, AB - Admin Boundary
(*,224.0.1.40) DM Up: 5d06h RP: 0.0.0.0
JP: Null(never) RPF: ,0.0.0.0 Flags: LH DSS
    gk                5d06h      off LI LH
(172.17.43.43,224.0.1.40)SPT DM Up: 2d23h
JP: Null(never) RPF: gw1,172.17.41.1 Flags: KAT(00:03:12)
    gk                2d23h      fwd
    gw1               2d23h      fwd
!
! R5 (regular PIM router) receives RP-Discovery messages

Auto-RP(0): Received RP-discovery packet of length 48, from 172.17.43.43,
RP_cnt 1, ht 181
Auto-RP(0): Added with (239.0.0.0/8, RP:172.22.22.242), PIMv2 v1
!
R5# show ip pim rp mapping
PIM Group-to-RP Mappings
Group(s) 239.0.0.0/8
  RP 172.22.22.242 (?), v2v1
    Info source: 172.17.43.43 (?), elected via Auto-RP
      Uptime: 00:01:07, expires: 00:02:47

```

Tip Many versions of IOS enable you to completely avoid operation in dense mode, even when Auto-RP is in place. To achieve that for interfaces configured with the **ip pim sparse-mode** command, you should configure the following global commands:

- **ip pim autorp listener:** interfaces configured for sparse-mode still operate in dense mode for Auto-RP.
- **no ip pim dm-fallback:** prevents fallback to dense mode, even when there is no remaining RP for a given multicast group.

Figure 15-13 exhibits an environment in which R7 has been configured as a Candidate RP for group range 224.224.0.0/16. The RP-Announce messages are sourced from address 172.17.17.17. Upon receipt of such an advertisement, R6 updates its cache and distributes new mapping information (using group 224.0.1.40).

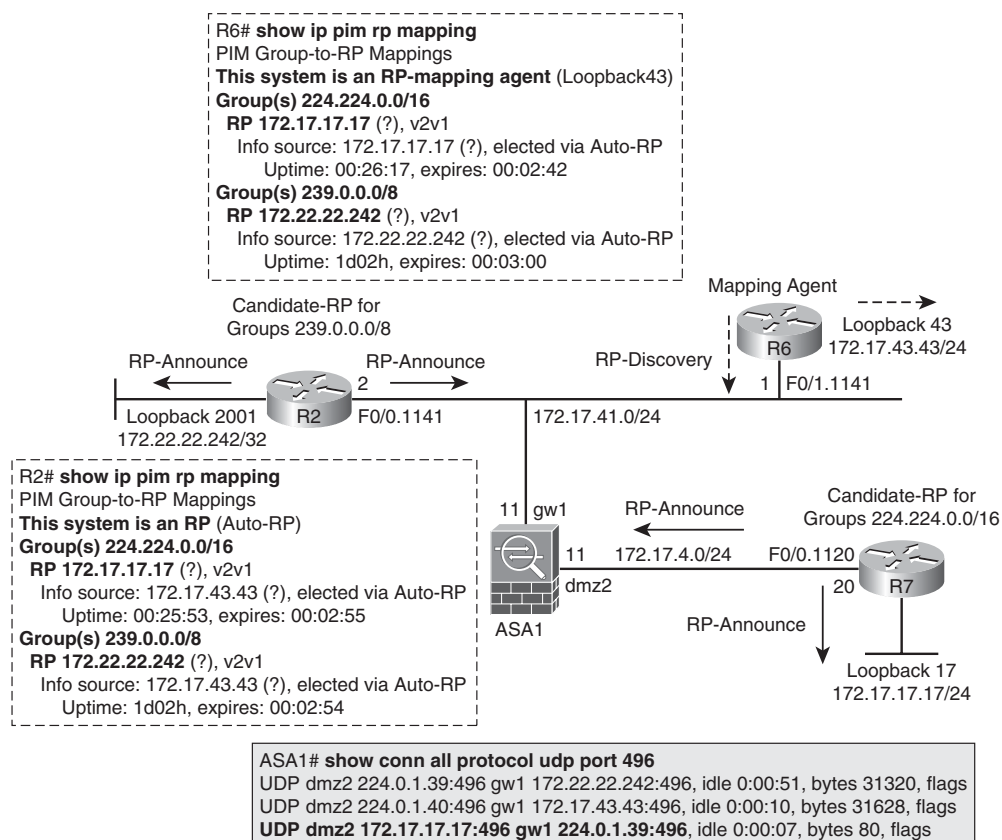


Figure 15-13 Reference Topology for RP-announcement Filtering

Example 15-8 refers to the topology of Figure 15-13 and documents a type of filter that can be applied to RP Mapping Agents. In the example, R7 starts announcing itself as a Candidate RP for Groups 224.100.0.0/16 (in addition to its previously assigned range). Given that the Mapping Agent expected this RP to be a candidate only for range 224.224.0.0/16, this announcement is filtered out (by R6) and is not included in the RP Discovery packet. It is interesting to observe that the RP Discovery packet contains only two RP entries instead of three.

Example 15-8 A Special Filter to Be Applied on RP Mapping Agents

```
! Controlling group-to-RP mappings on R6 (Mapping Agent)
access-list 10 permit 172.22.22.242
access-list 40 permit 239.0.0.0 0.255.255.255
ip pim rp-announce-filter rp-list 10 group-list 40
!
access-list 11 permit 172.17.17.17
```

```

access-list 41 permit 224.224.0.0 0.0.255.255
ip pim rp-announce-filter rp-list 11 group-list 41
!
! Filtering in action: 172.17.17.17 (R7) announces itself as RP for
224.100.0.0/16
R6# Auto-RP(0): Received RP-announce packet of length 48, from 172.22.22.242,
RP_cnt 1, ht 181
Auto-RP(0): Update (239.0.0.0/8, RP:172.22.22.242), PIMv2 v1

Auto-RP(0): Received RP-announce packet of length 54, from 172.17.17.17,
RP_cnt 1, ht 181
Auto-RP(0): Update (224.224.0.0/16, RP:172.17.17.17), PIMv2 v1
%SEC-6-IPACCESSLOGNP: list 41 denied 0 224.100.0.0 -> 0.0.0.0, 1 packet
Auto-RP(0): Filtered 224.100.0.0/16 for RP 172.17.17.17

Auto-RP(0): Build RP-Discovery packet
Auto-RP: Build mapping (224.224.0.0/16, RP:172.17.17.17), PIMv2 v1,
Auto-RP: Build mapping (239.0.0.0/8, RP:172.22.22.242), PIMv2 v1.
Auto-RP(0): Send RP-discovery packet of length 60 on FastEthernet0/1.1141
(2 RP entries)

```

Note Auto-RP is enabled by default on Cisco routers. To avoid rogue routers posing as mapping agents or candidate RPs, Auto-RP must be disabled on the edge of your IP network. This is achieved with the **ip multicast boundary** interface-level command (in conjunction with an ACL that denies the Auto-RP reserved groups).

Inserting ASA in a Multicast Routing Environment

Following are three basic options for inserting an ASA appliance in a multicast routing topology:

1. As a PIM-SM Router: By using this approach, ASA actively participates in multicast routing activities as a regular PIM router, with the particularity that it just supports static RP definition (with the aid of **pim rp-address** commands). In a similar fashion to IOS routers, ASA appliances also enable multiple RPs deployed throughout the network, each of them being in charge of particular set of multicast groups. It is convenient to keep in mind that although ASA does not consume Auto-RP information, it can be configured to enable these UDP messages through, as shown in Example 15-7.
2. As an IGMP Proxy Agent: In this type of design, ASA forwards IGMP messages, sent by interested receivers connected to one of its interfaces, to another interface where a designated multicast router (always a PIM router within the context of this book) resides. In this scenario, ASA does not participate directly in the PIM topology. This type of design is known as Stub Multicast Routing (SMR) and is discussed in more detail later.

3. In Transparent Mode: This last method enables PIM adjacencies (which require routers to be part of the same IP subnet) to be established through the appliance. Considering that the two previous methods require **multicast-routing** to be globally enabled in the ASA appliance (an option that is not supported in multiple-context mode), transparent mode corresponds to the only feasible alternative for devices configured for multiple context operation. Regular filtering resources are still available, with the advantage that ASA does not need to worry about RPF concepts, PIM operating modes, and so on. This capability of being decoupled from the underlying multicast routing topology renders bridge mode an inviting option for Data Center deployments.

Note Recall from Chapter 6, “Virtualization in the Firewall World,” that, while FWSM supports any combination of contexts (running in either routed or bridged mode), ASA does not. After the operating mode for a context is selected on an ASA appliance, all other possible contexts need to use the same choice. So, for ASA multiple-context deployments requiring interaction with multicast routing, all the contexts need to run in transparent mode.

Enabling Multicast Routing in ASA

The **multicast-routing** global configuration command enables an ASA appliance running in routed mode (single-context) to forward multicast packets. This command automatically enables PIM and IGMP on all of the interfaces, thus providing ASA with visibility of the activities related to such protocols.

Example 15-9 relates to the topology depicted in Figure 15-14 and illustrates some aspects of ASA’s behavior regarding multicast traffic:

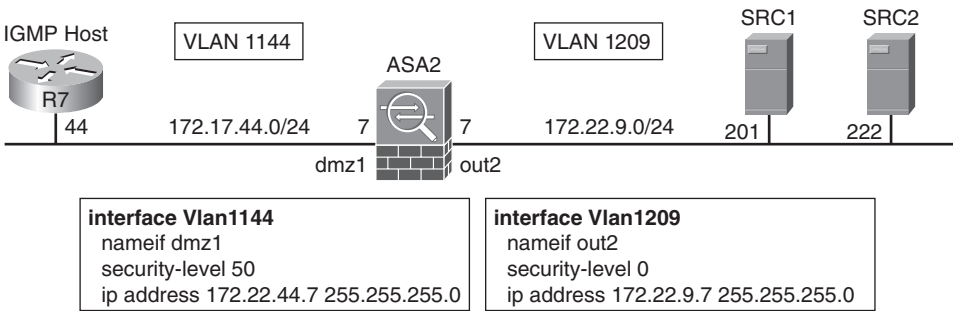


Figure 15-14 Simple Multicast Routing Scenario Involving an ASA Appliance

- ASA interfaces join the PIM-Routers group (224.0.0.13) and the All-Hosts group (224.0.0.1).
- The receiver on R7 (connected to ‘dmz1’) joins the group 239.33.33.33.

- A new IGMP Identity connection is created for group 239.33.33.33 on interface 'dmz1' of the ASA appliance.
- The (*,G) entry for this group is inserted in the **mroute** table. Although there is no PIM RP defined yet, the entry is created in Sparse Mode. From a PIM router point of view, ASA acts as a *last-hop router* for this receiver.

Example 15-9 does not necessarily represent a real-life multicast routing scenario. It simply illustrates ASA behavior when initially enabled for multicast routing.

Example 15-9 Basic Multicast Information on ASA

```
! Initial situation on ASA (no receivers seen on any of its interfaces)
ASA2# show conn all
4 in use, 10 most used
PIM out2 224.0.0.13 NP Identity Ifc 172.22.9.7, idle 0:00:18, bytes 12198
IGMP out2 224.0.0.1 NP Identity Ifc 172.22.9.7, idle 0:01:55, bytes 8
PIM dmz1 224.0.0.13 NP Identity Ifc 172.22.44.7, idle 0:00:11, bytes 12198
IGMP dmz1 224.0.0.1 NP Identity Ifc 172.22.44.7, idle 0:01:36, bytes 8
!
! Receiver R7 joins multicast group 239.33.33.33

R7(config)# interface f0/0.1144
R7(config-subif)#ip igmp join-group 239.33.33.33
IGMP(9): WAVL Insert group: 239.33.33.33 interface: FastEthernet0/0.1144
Successful
IGMP(9): Send v2 Report for 239.33.33.33 on FastEthernet0/0.1144
!
! What ASA sees (just after receiving an IGMP Report from the first receiver)

IGMP: Received v2 Report on dmz1 from 172.22.44.44 for 239.33.33.33
IGMP: group_db: add new group 239.33.33.33 on dmz1
IGMP: MRIB updated (*,239.33.33.33) : Success
IGMP: Switching to EXCLUDE mode for 239.33.33.33 on dmz1
IGMP: Updating EXCLUDE group timer for 239.33.33.33

IPv4 PIM: [0] (*,239.33.33.33/32) MRIB update (a=0,f=0,t=1)
IPv4 PIM: [0] (*,239.33.33.33/32) dmz1 MRIB update (f=100,c=100)
IPv4 PIM: (*,239.33.33.33) Create entry
IPv4 PIM: [0] (*,239.33.33.33/32) MRIB modify DC
IPv4 PIM: [0] (*,239.33.33.33/32) NULLIF-skip MRIB modify A
IPv4 PIM: (*,239.33.33.33) dmz1 Local state changed from Null to Join
IPv4 PIM: (*,239.33.33.33) dmz1 Start being last hop
IPv4 PIM: (*,239.33.33.33) Start being last hop
IPv4 PIM: (*,239.33.33.33) Start signaling sources
IPv4 PIM: [0] (*,239.33.33.33/32) NULLIF-skip MRIB modify NS
IPv4 PIM: (*,239.33.33.33) dmz1 FWD state change from Prune to Forward
```

```

IPv4 PIM: [0] (*,239.33.33.33/32) dmz1 MRIB modify F NS
IPv4 PIM: (*,239.33.33.33) Updating J/P status from Null to Join
IPv4 PIM: (*,239.33.33.33) J/P scheduled in 0.0 secs
IPv4 PIM: (*,239.33.33.33) Processing timers
IPv4 PIM: (*,239.33.33.33) J/P processing
IPv4 PIM: (*,239.33.33.33) Periodic J/P scheduled in 50 secs
IPv4 PIM: (*,239.33.33.33) No RPF interface to send J/P
IPv4 PIM: (*,239.33.33.33) dmz1 Processing timers
!
! The connection table changes (please compare with the initial table in this
example)
ASA2# show conn all
5 in use, 10 most used
PIM out2 224.0.0.13 NP Identity Ifc 172.22.9.7, idle 0:00:28, bytes 14098
IGMP out2 224.0.0.1 NP Identity Ifc 172.22.9.7, idle 0:01:50, bytes 8
PIM dmz1 224.0.0.13 NP Identity Ifc 172.22.44.7, idle 0:00:18, bytes 14098
IGMP dmz1 224.0.0.1 NP Identity Ifc 172.22.44.7, idle 0:01:31, bytes 8
IGMP dmz1 172.22.44.44 NP Identity Ifc 239.33.33.33, idle 0:01:24, bytes 8
!
! Obtaining information about IGMP groups
ASA2# show igmp groups
IGMP Connected Group Membership


| Group        | Address | Interface | Uptime   | Expires  | Last Reporter |
|--------------|---------|-----------|----------|----------|---------------|
| 239.33.33.33 |         | dmz1      | 00:04:58 | 00:02:31 | 172.22.44.44  |


!
! Displaying multicast routing information for a certain group
ASA2# show mroute 239.33.33.33
Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group,
C - Connected, L - Local, I - Received Source Specific Host Report,
P - Pruned, R - RP-bit set, F - Register flag, T - SPT-bit set,
J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, State
(*, 239.33.33.33), 00:05:57/never, RP 0.0.0.0, flags: SCJ
Incoming interface: Null
RPF nbr: 0.0.0.0
Immediate Outgoing interface list:
dmz1, Forward, 00:05:57/never

```

Example 15-10 refers to the topology depicted in Figure 15-14:

- A Source (172.22.9.201) connected to the 'out2' subnet starts sending UDP packets to group 239.33.33.33.

- The (S,G) entry (172.22.9.201, 239.33.33.33) is derived from the (*,G) entry earlier created in Example 15-9.
- Another source (172.22.9.222) sends a ping to group 239.33.33.33, thus inducing the creation of a corresponding (S,G) entry. Given that **inspect icmp** is not configured, the **echo-reply** is decoupled from the **echo-request**. The **show conn all** command then displays three ICMP connections (instead of the two entries seen for UDP).

Example 15-10 Sample Multicast Traffic Through ASA

```

! Host 172.22.9.201 sends UDP traffic to group 239.33.33.33
%ASA-6-302015: Built inbound UDP connection 1012 for
out2:172.22.9.201/13000 (172.22.9.201/13000) to
identity:239.33.33.33/123 (239.33.33.33/123)
!
! (S,G) state is created from (*,G) (S = 172.22.9.201, G = 239.33.33.33)

ASA2# show mroute 239.33.33.33
Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group,
      C - Connected, L - Local, I - Received Source Specific Host Report,
      P - Pruned, R - RP-bit set, F - Register flag, T - SPT-bit set,
      J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, State
(*, 239.33.33.33), 00:44:10/never, RP 0.0.0.0, flags: SCJ
  Incoming interface: Null
  RPF nbr: 0.0.0.0
  Immediate Outgoing interface list:
    dmz1, Forward, 00:44:10/never
(172.22.9.201, 239.33.33.33), 00:00:38/00:02:51, flags: SFJT
  Incoming interface: out2
  RPF nbr: 172.22.9.201
  Inherited Outgoing interface list:
    dmz1, Forward, 00:44:10/never
!
! SRC2 (172.22.9.222) pings the multicast group 239.33.33.33

%ASA-6-302020: Built inbound ICMP connection for faddr 172.22.9.222/0 gaddr
239.33.33.33/0 laddr 239.33.33.33/0
%ASA-6-302020: Built outbound ICMP connection for faddr 172.22.9.222/0 gaddr
172.22.44.44/0 laddr 172.22.44.44/0
!
ASA2# show conn all | include ICMP
ICMP out2 172.22.9.222:0 dmz1 172.22.44.44:0, idle 0:00:11, bytes 0
ICMP out2 172.22.9.222:0 dmz1 239.33.33.33:0, idle 0:00:11, bytes 0

```



```
ICMP out2 172.22.9.222:0 NP Identity Ifc 239.33.33.33:0, idle 0:00:11, bytes 0
```

Stub Multicast Routing in ASA

Stub Multicast Routing (SMR) provides a way for IGMP hosts that connect to a given ASA interface to dynamically register with a multicast router located on a distinct interface of the appliance. When configured for SMR, instead of acting as a full multicast router, an ASA simply forwards the IGMP messages between hosts and multicast routers, playing a role that can be denominated an IGMP Proxy Agent.

Figure 15-15 portrays a simple SMR scenario for which Host C1, connected to ASA2 interface 'dmz1', has joined group 239.33.33.33. The interface level command **igmp forward interface out2** is configured on the interface that has connected receivers (in this case, 'dmz1') and represents the basic instruction for SMR operation. There is no need for any additional configuration on interface 'out2'.

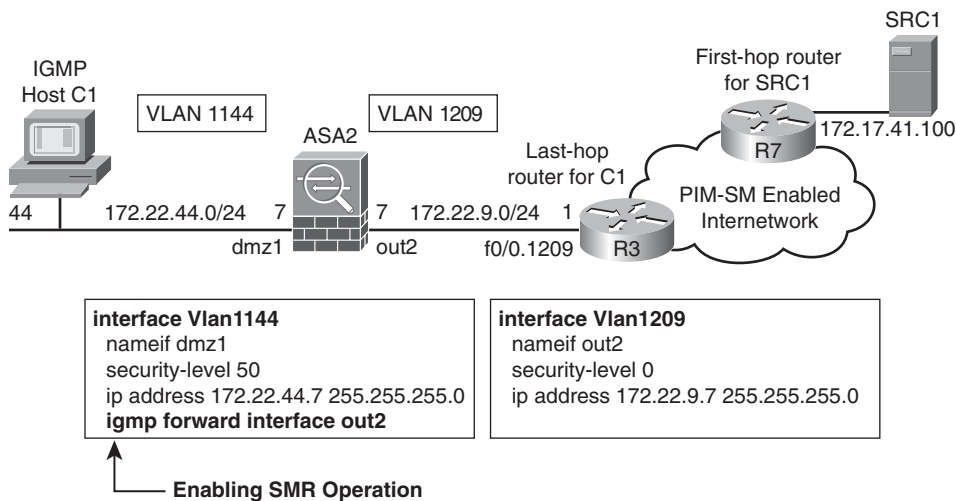


Figure 15-15 Reference Topology for SMR Analysis

Example 15-11 relates to the topology in Figure 15-15 and registers the IGMP forwarding performed by ASA when an IGMP Report for group 239.33.33.33 is sent by receiver 172.22.44.44 (C1). Other important concepts revealed by this example follow:

- Group 239.33.33.33 is now seen on interface 'out2'. (Compare the **show conn all** output with that in Example 15-10.) The **conn** table also reveals that the Identity connections to the PIM-Routers group (224.0.0.13) have been removed.
- IGMP is automatically disabled on interface 'out2'. This interface is used only as a path to reach Router (R3), who plays the role of PIM Designated Router for subnet 172.22.9.0/24.

- The **mroute** to group 239.33.33.33 was created in Dense Mode (in contrast with the Sparse Mode operation in Example 15-10).

Example 15-11 Basic Information for Stub Multicast Routing Scenario

```

! ASA is configured to forward IGMP messages from 'dmz1' to 'out2'
ASA2(config)# interface vlan 1144
ASA2(config-if)# igmp forward interface out2
IGMP: Interface out2 disabled router side processing
IGMP: Send v2 general Query on dmz1
IGMP: Received v2 Query on dmz1 from 172.22.44.7
IGMP: Received v2 Report on dmz1 from 172.22.44.44 for 239.33.33.33
IGMP: Updating EXCLUDE group timer for 239.33.33.33
IGMP: Forward v2 Report for group 239.33.33.33 from 172.22.44.44 out interface out2
!
ASA2# show igmp interface dmz1 ! include forwarding
    IGMP forwarding on interface out2
!
! Consequences of enabling ASA2 for Stub Multicast Routing (SMR)

ASA2# show igmp interface out2
out2 is up, line protocol is up
    Internet address is 172.22.9.7/24
    IGMP is disabled on interface
!
ASA2# show conn all
3 in use, 10 most used
IGMP out2 239.33.33.33 NP Identity Ifc 172.22.9.7, idle 0:00:05, bytes 8
IGMP dmz1 224.0.0.1 NP Identity Ifc 172.22.44.7, idle 0:00:16, bytes 8
IGMP dmz1 172.22.44.44 NP Identity Ifc 239.33.33.33, idle 0:00:05, bytes 8
!
ASA2# show mroute 239.33.33.33
Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group,
       C - Connected, L - Local, I - Received Source Specific Host Report,
       P - Pruned, R - RP-bit set, F - Register flag, T - SPT-bit set,
       J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, State
(*, 239.33.33.33), 00:00:29/never, RP 0.0.0.0, flags: DC
    Incoming interface: Null
    RPF nbr: 0.0.0.0
    Immediate Outgoing interface list:
        dmz1, Forward, 00:00:29/never

```

Example 15-12 refers to the scenario in Figure 15-15 and is a continuation in Example 15-11. The router R3 connected to the 'out2' subnet was configured for PIM SM and is aware of the IGMP messages forwarded by ASA2 (who is now operating as an IGMP Proxy Agent). Noteworthy facts in this example are summarized in the following:

- R3 becomes the IGMP Querying Router and the PIM Designated Router in the 'out2' subnet. ASA is still the Querying Router for 'dmz1'. The **show conn all** command demonstrates that R3 has joined the PIM-Routers group (224.0.0.13) and the All-Hosts group (224.0.0.1). The IGMP queries are sent to this last group.
- ASA does not run PIM in the SMR scenario. As a result, R3 sees no PIM neighbor on 'out2' (Nbr Count = 0).
- R3 sees the IGMP messages as sourced from ASA 'out2' interface (172.22.9.7) and not from the original host. R3 sends a triggered PIM Join toward the RP upon receipt of the IGMP Membership Report.
- R3 participates in a PIM-SM topology and therefore the **mroute** to group 239.33.33.33 is created as SM. However, ASA (which is not working as a PIM router,) considers this mroute as a DM entry. This behavior is characterized in the previous example.

Example 15-12 PIM-SM Router Becomes Available on SMR Scenario

```
! R3 is configured for PIM on interface F0/0.1209 (connected to 'out2')
IGMP(0): Send v2 init Query on FastEthernet0/0.1209
PIM(0): Check DR after interface: FastEthernet0/0.1209 came up!
PIM(0): Changing DR for FastEthernet0/0.1209, from 0.0.0.0 to 172.22.9.1 (this
system)
%PIM-5-DRCHG: DR change from neighbor 0.0.0.0 to 172.22.9.1 on interface
FastEthernet0/0.1209
!
R3# show ip igmp interface f0/0.1209 | include router
Current IGMP router version is 2
Multicast designated router (DR) is 172.22.9.1 (this system)
IGMP querying router is 172.22.9.1 (this system)
!
R3# show ip pim interface f0/0.1209
Address          Interface          Ver/   Nbr    Query  DR      DR
                  Mode              Count  Intvl  Prior
172.22.9.1       FastEthernet0/0.1209 v2/SD  0      30     1      172.22.9.1
!
IGMP(0): Send v2 general Query on FastEthernet0/0.1209
IGMP(0): Received v2 Report on FastEthernet0/0.1209 from 172.22.9.7 for
239.33.33.33
IGMP(0): Received Group record for group 239.33.33.33, mode 2 from 172.22.9.7 for
0 sources
IGMP(0): WAVL Insert group: 239.33.33.33 interface: FastEthernet0/0.1209
Successful
```

```

IGMP(0): Switching to EXCLUDE mode for 239.33.33.33 on FastEthernet0/0.1209
IGMP(0): Updating EXCLUDE group timer for 239.33.33.33
IGMP(0): MRT Add/Update FastEthernet0/0.1209 for (*,239.33.33.33) by 0
PIM(0): Check RP 172.22.22.242 into the (*, 239.33.33.33) entry
PIM(0): Building Triggered (*,G) Join/ (S,G,RP-bit) Prune message for 239.33.33.33
PIM(0): Insert (*,239.33.33.33) join in nbr 172.22.13.1's queue
!
R3# show ip igmp groups f0/0.1209
IGMP Connected Group Membership


| Group Address       | Interface            | Uptime   | Expires  | Last Reporter     |
|---------------------|----------------------|----------|----------|-------------------|
| Group Accounted     |                      |          |          |                   |
| <b>239.33.33.33</b> | FastEthernet0/0.1209 | 00:19:35 | 00:02:11 | <b>172.22.9.7</b> |


!
R3# show ip mroute 239.33.33.33 | begin \ (
(*, 239.33.33.33), 00:22:07/00:01:44, RP 172.22.22.242, flags: SJC
Incoming interface: FastEthernet0/0.1213, RPF nbr 172.22.13.1
Outgoing interface list:
FastEthernet0/0.1209, Forward/Sparse-Dense, 00:22:07/00:01:44
!
! ASA's perspective

ASA2# show conn all
5 in use, 10 most used
IGMP out2 239.33.33.33 NP Identity Ifc 172.22.9.7, idle 0:01:17, bytes 8
IGMP dmz1 224.0.0.1 NP Identity Ifc 172.22.44.7, idle 0:01:25, bytes 8
PIM out2 172.22.9.1 NP Identity Ifc 224.0.0.13, idle 0:00:04, bytes 798
IGMP dmz1 172.22.44.44 NP Identity Ifc 239.33.33.33, idle 0:01:17, bytes 8
IGMP out2 172.22.9.1 NP Identity Ifc 224.0.0.1, idle 0:00:05, bytes 88

```

Example 15-13 still builds upon the SMR scenario of Figure 15-15. The source 172.17.41.100 started transmission to multicast group 239.33.33.33, which has a receiver on ASA interface 'dmz1'. R3 (the PIM-SM DR for subnet 'out2') receives the traffic via Sparse Mode, whereas ASA (enabled for SMR) operates in Dense Mode.

Example 15-13 Source Begins Transmission on SMR Scenario

```

! What R3 (PIM-SM speaker) sees after source 172.17.41.100 starts transmission
R3# show ip mroute 239.33.33.33 | begin \ (
(*, 239.33.33.33), 00:03:02/stopped, RP 172.22.22.242, flags: SJC
Incoming interface: FastEthernet0/0.1213, RPF nbr 172.22.13.1
Outgoing interface list:
FastEthernet0/0.1209, Forward/Sparse-Dense, 00:03:02/00:02:05
(172.17.41.100, 239.33.33.33), 00:02:29/00:00:40, flags: JT
Incoming interface: FastEthernet0/0.1213, RPF nbr 172.22.13.1

```

Outgoing interface list:

FastEthernet0/0.1209, Forward/Sparse-Dense, 00:02:29/00:02:05

!

! ASA's perspective

%ASA-6-302015: Built inbound UDP connection 2385 for
out2:172.17.41.100/33000 (172.17.41.100/33000) to
identity:239.33.33.33/123 (239.33.33.33/123)

!

ASA2# **show conn all protocol udp**

7 in use, 11 most used

UDP out2 172.17.41.100:33000 dmz1 239.33.33.33:123, idle 0:00:25, bytes 20, flags -
UDP out2 172.17.41.100:33000 NP Identity Ifc 239.33.33.33:123, idle 0:00:25, bytes
0, flags -

!

ASA2# **show mroute 239.33.33.33 | begin **(

(*, 239.33.33.33), 05:47:57/never, RP 0.0.0.0, flags: DC

Incoming interface: Null

RPF nbr: 0.0.0.0

Immediate Outgoing interface list:

dmz1, Forward, 05:47:57/never

(172.17.41.100, 239.33.33.33), 00:00:17/00:03:12, **flags: DJT**

Incoming interface: out2

RPF nbr: 172.22.9.1**Inherited Outgoing interface list:**

dmz1, Forward, 05:47:57/never

Example 15-14 shows a way to control the groups for which ASA acts as an IGMP Proxy Agent. In the example, only the groups in the range 239.0.0.0/8 are allowed to profit from ASA IGMP forwarding services. For instance, a receiver connected to 'dmz1' sends an IGMP Report to 230.230.230.230, but the request to become part of this group is denied by ASA.

Example 15-14 *Defining an IGMP access-group*

```
! Defining an IGMP access-group and applying it to an interface
access-list IGMP-DMZ1 extended permit ip any 239.0.0.0 255.0.0.0
!
interface Vlan1144
 nameif dmz1
 security-level 50
 ip address 172.22.44.7 255.255.255.0
 igmp access-group IGMP-DMZ1
!
ASA2# show igmp interface dmz1
```

```

dmz1 is up, line protocol is up
  Internet address is 172.22.44.7/24
  IGMP is enabled on interface
  Current IGMP version is 2
  IGMP query interval is 125 seconds
  IGMP querier timeout is 255 seconds
  IGMP max query response time is 10 seconds
  Last member query response interval is 1 seconds
Inbound IGMP access group is: IGMP-DMZ1
  IGMP limit is 500, currently active joins: 0
  Cumulative IGMP activity: 0 joins, 0 leaves
  IGMP querying router is 172.22.44.7 (this system)
!
! ASA behavior after IGMP access-group is applied to interface 'dmz1'

IGMP: Received v2 Report on dmz1 from 172.22.44.44 for 230.230.230.230
IGMP: Group 230.230.230.230 access denied on dmz1

```

ASA Acting as a PIM-SM Router

The most important theoretical concepts pertaining to PIM-SM have been examined in previous subsections. The current section is devoted to document some reference scenarios in which ASA is configured as a PIM-SM router.

Figure 15-16 displays a sample network that contains an IGMP receiver directly connected to an ASA appliance running PIM-SM. R2 is again the RP for this environment, with the particularity that this is manually defined for ASA. (The group-to-RP mapping information was distributed via Auto-RP in the earlier IOS examples). Figure 15-16 also shows that ASA2 and R3 are PIM neighbors.

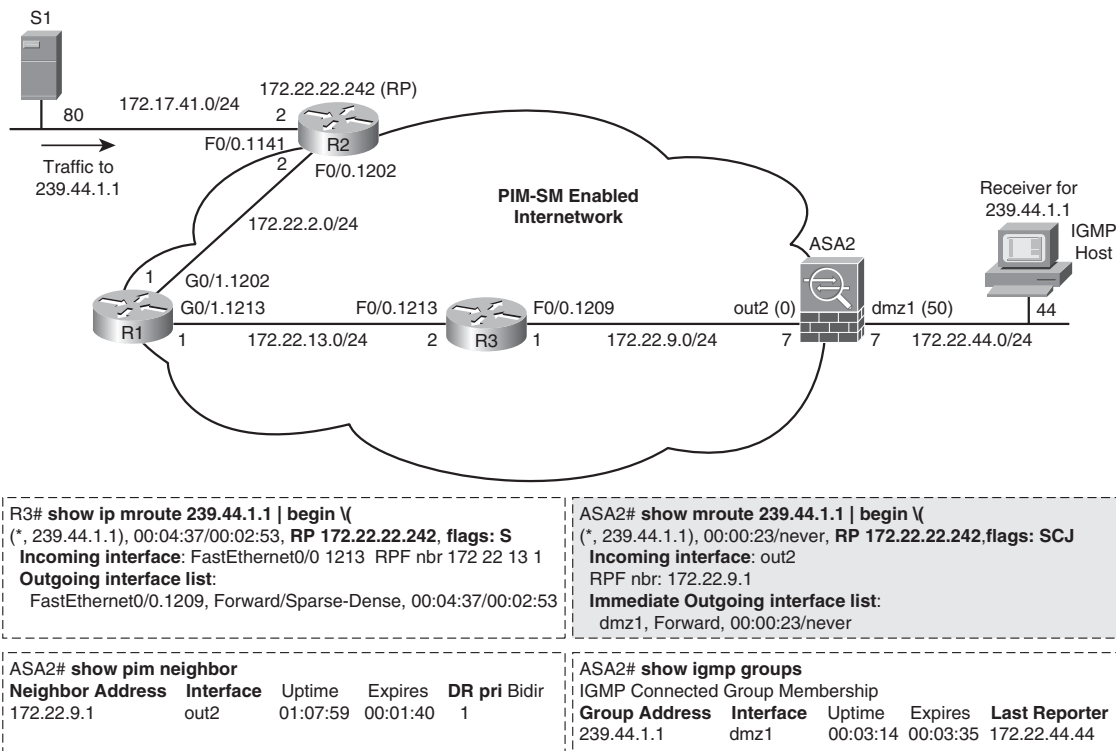


Figure 15-16 Basic PIM-SM Scenario with a Receiver Connected to ASA

Example 15-15 presents details about RP configuration for ASA for the topology drawn in Figure 15-16. Example 15-16 provides extra information for this network by documenting the `mroute` table in ASA after the source 172.17.41.80 starts transmitting to group 239.44.1.1.

Example 15-15 RP Definition in ASA

```
! ASA only supports static RP definition
access-list RP-GROUPS1 standard permit 239.0.0.0 255.0.0.0
pim rp-address 172.22.22.242 RP-GROUPS1
!
```

ASA2# show pim group-map

Group Range	Proto	Client	Groups	RP address	Info
224.0.1.39/32*	DM	static	1	0.0.0.0	
224.0.1.40/32*	DM	static	1	0.0.0.0	

224.0.0.0/24*	L-Localstatic 1	0.0.0.0	
232.0.0.0/8*	SSM config 0	0.0.0.0	
239.0.0.0/8*	SM config 1	172.22.22.242	RPF: V11209,172.22.9.1
224.0.0.0/4*	SM static 0	0.0.0.0	RPF: ,0.0.0.0

Example 15-16 Building (S,G) State in ASA

```

! UDP connection from source to group 239.44.1.1 is seen by ASA
%ASA-6-302015: Built inbound UDP connection 6746 for out2:172.17.41.80/33000
(172.17.41.80/33000) to identity:239.44.1.1/123 (239.44.1.1/123)
!
! The (S,G) entry is created for source 172.17.41.80
ASA2# show mroute 239.44.1.1
Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group,
      C - Connected, L - Local, I - Received Source Specific Host Report,
      P - Pruned, R - RP-bit set, F - Register flag, T - SPT-bit set,
      J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, State
(*, 239.44.1.1), 00:34:52/never, RP 172.22.22.242, flags: SCJ
  Incoming interface: out2
  RPF nbr: 172.22.9.1
  Immediate Outgoing interface list:
    dmz1, Forward, 00:34:52/never
(172.17.41.80, 239.44.1.1), 00:00:25/00:03:04, flags: SJT
  Incoming interface: out2
  RPF nbr: 172.22.9.1
  Inherited Outgoing interface list:
    dmz1, Forward, 00:34:52/never

```

Figure 15-17 extends the reach of the PIM-SM network of Figure 15-16. The PIM-SM Router R7, which has a connected receiver for group 239.99.99.99, is now a PIM-SM neighbor of ASA2 on subnet 172.22.44.0/24 ('dmz1'). The (*,G) entry for this group is shown from the perspective of ASA2. It is instructive to compare the flags in the output of the **show mroute** command with that of Example 15-16.

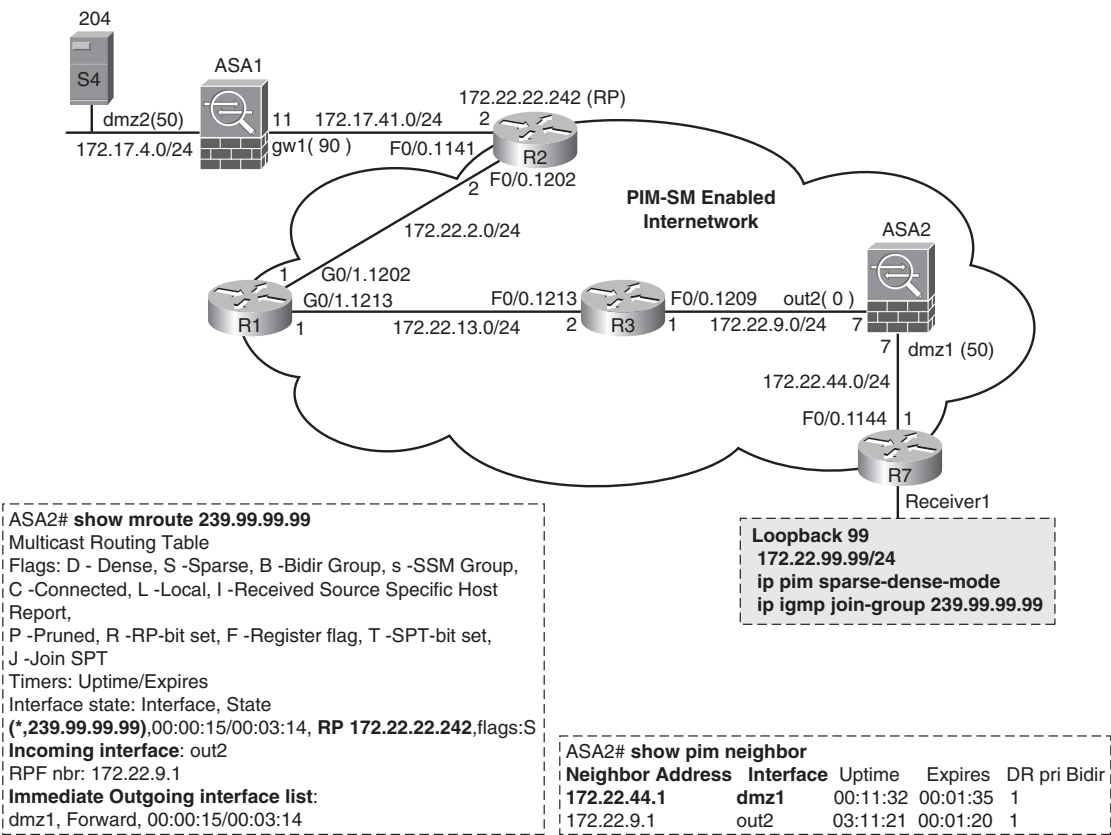


Figure 15-17 ASA as a PIM-SM Router with PIM Neighbors on Two Interfaces

Example 15-17 also relates to the topology of Figure 15-17 and represents a situation in which the source S4 (172.17.4.204), behind ASA1, starts transmitting to group 239.99.99.99. This group already had a receiver connected to R7, as described in Example 15-16. It is interesting to notice the Tunnel 0 interface connecting ASA1 to the RP. (The PIM Register and Register-Stop messages are unicast using this Tunnel interface.) For more details about the theory of source registration, refer to the IOS examples in the section “Multicast Routing with PIM.”

Example 15-17 Source Registration Through ASA with Receiver Joining First

```
! S4 connection to group 239.99.99.99 is seen by ASA1
%ASA-6-302015: Built inbound UDP connection 10456 for
dmz2:172.17.4.204/58000 (172.17.4.204/58000) to
identity:239.99.99.99/123 (239.99.99.99/123)
```

```

!
! Source registration process as seen by ASA1 (acting as first-hop router)
IPv4 PIM: [0] (172.17.4.204,239.99.99.99/32) dmz2 MRIB update (f=20,c=20)
IPv4 PIM: [0] (172.17.4.204,239.99.99.99) Signal present on dmz2
IPv4 PIM: (172.17.4.204,239.99.99.99) Create entry
IPv4 PIM: (172.17.4.204,239.99.99.99) RPF changed from 0.0.0.0/- to
172.17.4.204/dmz2
[ output suppressed ]
IPv4 PIM: (172.17.4.204,239.99.99.99) Start registering to 172.22.22.242
IPv4 PIM: (172.17.4.204,239.99.99.99) Tunnel0 J/P state changed from Null to Join
IPv4 PIM: (172.17.4.204,239.99.99.99) Tunnel0 FWD state change from Prune to
Forward
IPv4 PIM: (172.17.4.204,239.99.99.99) Updating J/P status from Null to Join
IPv4 PIM: (172.17.4.204,239.99.99.99) J/P scheduled in 0.0 secs
IPv4 PIM: [0] (172.17.4.204,239.99.99.99/32) dmz2 MRIB modify NS
IPv4 PIM: (172.17.4.204,239.99.99.99) Set SPT bit
[ output suppressed ]
IPv4 PIM: J/P entry: Join root: 172.17.4.204 group: 239.99.99.99 flags: S
IPv4 PIM: (172.17.4.204,239.99.99.99) gw1 J/P state changed from Null to Join
IPv4 PIM: (172.17.4.204,239.99.99.99) gw1 FWD state change from Prune to Forward
[ output suppressed ]
IPv4 PIM: (172.17.4.204,239.99.99.99) Suppress J/P to connected source
IPv4 PIM: (172.17.4.204,239.99.99.99) Tunnel0 Processing timers
IPv4 PIM: (172.17.4.204,239.99.99.99) Received Register-Stop
IPv4 PIM: (172.17.4.204,239.99.99.99) Stop registering
!
ASA1# show mroute 239.99.99.99 | begin \((
(172.17.4.204, 239.99.99.99), 00:00:04/00:03:25, flags: SFT
Incoming interface: dmz2
RPF nbr: 172.17.4.204
Outgoing interface list:
gw1, Forward, 00:00:04/00:03:25
!
ASA1# show conn all | include PIM
PIM gw1 172.22.22.242 NP Identity Ifc 172.17.41.11, idle 0:00:24, bytes 150
!
ASA1# show pim tunnel
Interface RP Address Source Address
Tunnel0 172.22.22.242 172.17.41.11
!
! R2's perspective (RP)

R2# show ip mroute 239.99.99.99 | begin \((
(*, 239.99.99.99), 00:02:46/stopped, RP 172.22.22.242, flags: S
Incoming interface: Null, RPF nbr 0.0.0.0

```

```

Outgoing interface list:
  FastEthernet0/0.1202, Forward/Sparse-Dense, 00:02:46/00:02:42
(172.17.4.204, 239.99.99.99), 00:00:06/00:03:27, flags: T
Incoming interface: FastEthernet0/0.1141, RPF nbr 172.17.41.11
Outgoing interface list:
  FastEthernet0/0.1202, Forward/Sparse-Dense, 00:00:06/00:03:23

```

Summary of Multicast Forwarding Rules on ASA

Example 15-18 assembles some of the rules that govern multicast forwarding on ASA. The information is further detailed by using the **show asp table classify** and **show asp drop** commands:

- A Class D IP address can never be used as the source address of a packet.
- TCP packets (IP Protocol = 6) destined to multicast group addresses are dropped.
- When the Ethernet multicast MAC address does not correspond to the Layer 3 group address, the packet is dropped and the *Early security checks failed* counter is incremented. This rule takes precedence over any ACL permission. The construction of the Ethernet multicast address from the L3 group address was explained in Figure 15-1.

Example 15-18 Important ASA Rules for Multicast Forwarding (1)

```

! An IP address belonging to Class D cannot be used as source address
in id=0xd7f4fed0, priority=500, domain=permit, deny=true
    hits=0, user_data=0x6, cs_id=0x0, flags=0x0, protocol=0
    src ip=224.0.0.0, mask=240.0.0.0, port=0
    dst ip=0.0.0.0, mask=0.0.0.0, port=0, dscp=0x0
!
! TCP Traffic to a Multicast Group is never allowed
in id=0xd7f51a60, priority=500, domain=permit, deny=true
    hits=0, user_data=0x0, cs_id=0x0, reverse, flags=0x0, protocol=6
    src ip=0.0.0.0, mask=0.0.0.0, port=0
    dst ip=224.0.0.0, mask=240.0.0.0, port=0, dscp=0x0
!
! Packet is dropped if the L2 multicast MAC and the L3 group addresses are inconsistent
ASA2# show asp drop
Frame drop:
    Early security checks failed (security-failed) 1
    FP L2 rule drop (l2_acl) 46

```

Example 15-19 illustrates some other rules pertaining to multicast forwarding on ASA. For all the situations listed in the following, it should be assumed that **multicast-routing** is enabled and that the *Early security checks* have succeeded:

- Typical multicast applications are supposed to use UDP transport. Although there is no **access-list** tied to an interface, UDP multicast packets are automatically allowed through an ASA appliance (even from a lower security-level interface to a higher one). As soon as an interface ACL comes to the scene, UDP traffic intended to traverse an ASA appliance must be explicitly permitted.
- To illustrate the point about UDP traffic, the example initially shows the built-in ASP classify rules that deal with Auto-RP. This assumes that no interface ACL is in place. After an ACL is applied to the interface where Auto-RP UDP packets arrive, explicit permissions become necessary. These new user-defined rules take precedence over the original ones and are also documented in the example.
- The example shows one ACE that explicitly enables UDP to groups 225.0.0.0/8 from sources on the 172.17.0.0/16 range. The corresponding user-defined rule, as it appears in the ASP table (after applying the ACL) are documented (**show asp table classify interface out2**).
- UDP traffic sent to a group that is not permitted by the interface ACL (OUT2 in this case) is dropped (in a similar fashion to what happens with unicast drops).
- ICMP, which is deemed a management protocol, is dealt with in a different manner. When **inspect icmp** is not configured, ICMP traffic to a group is always accepted. The high priority rule shown in the example is in charge of handling *to-the-box* ICMP traffic and takes precedence over interface ACLs (even those that include the **control-plane** keyword in the **access-group** command).

Example 15-19 Important ASA Rules for Multicast Forwarding (2)

```
! Default rules allowing Auto-RP (UDP/496) through ASA
in id=0xd816d9b0, priority=12, domain=autorp, deny=false
    hits=2, user_data=0xd7ec5ad8, cs_id=0x0, flags=0x0, protocol=17
    src ip=0.0.0.0, mask=0.0.0.0, port=0
    dst ip=224.0.1.39, mask=255.255.255.255, port=496, dscp=0x0
in id=0xd816da48, priority=12, domain=autorp, deny=false
    hits=2, user_data=0xd7ec5ad8, cs_id=0x0, flags=0x0, protocol=17
    src ip=0.0.0.0, mask=0.0.0.0, port=0
    dst ip=224.0.1.40, mask=255.255.255.255, port=496, dscp=0x0
!
! User-defined rules to allow Auto-RP (needed if an interface ACL is created)
in id=0xd83a88b8, priority=12, domain=permit, deny=false
    hits=1, user_data=0xd6157110, cs_id=0x0, flags=0x0, protocol=17
    src ip=172.17.43.43, mask=255.255.255.255, port=0
```

```

    dst ip=224.0.1.40, mask=255.255.255.255, port=496, dscp=0x0
in  id=0xd83a8aa0, priority=12, domain=permit, deny=false
    hits=1, user_data=0xd6157160, cs_id=0x0, flags=0x0, protocol=17
    src ip=172.22.22.242, mask=255.255.255.255, port=0
    dst ip=224.0.1.39, mask=255.255.255.255, port=496, dscp=0x0
!
! ACE allowing UDP to groups 225.0.0.0/8 and corresponding rule in the ASP table

access-list OUT2 extended permit udp 172.17.0.0 255.255.0.0 225.0.0.0 255.0.0.0
!
in  id=0xd53924e0, priority=120, domain=permit, deny=false
    hits=0, user_data=0xd6157340, cs_id=0x0, flags=0x0, protocol=17
    src ip=172.17.0.0, mask=255.255.0.0, port=0
    dst ip=225.0.0.0, mask=255.0.0.0, port=0, dscp=0x0
!
! UDP Traffic sent to a group that is not allowed by the interface ACL is
dropped
%ASA-4-106023: Deny udp src out2:172.22.9.220/22000 dst identity:228.22.22.22/123 by
access-group "OUT2" [0x0, 0x0]
!
! ICMP traffic is permitted by default (regardless of ACLs)

in  id=0xd7f4ecf8, priority=120, domain=permit, deny=false
    hits=1, user_data=0x0, cs_id=0x0, reverse, flags=0x0, protocol=1
    src ip=0.0.0.0, mask=0.0.0.0, port=0
    dst ip=0.0.0.0, mask=0.0.0.0, port=0, dscp=0x0

```

Summary

This chapter presented the basic concepts of IPv4 multicast routing.

Important aspects pertaining to the theory of operations of the PIM multicast routing protocol (when configured for Sparse Mode or Dense Mode), and the typical issues that arise when multicast traffic crosses firewalls, were also analyzed.

Further Reading

Developing IP Multicast Networks, Volume I (Beau Williamson)

<http://www.ciscopress.com/bookstore/product.asp?isbn=1587142899>

Cisco Firewalls and IPv6

This chapter covers the following topics:

- Introduction to IPv6
- Overview of IPv6 Addressing
- The IPv6 Header Format
- IPv6 Connectivity Basics
- Handling IOS IPv6 Access Control Lists
- IPv6 support in the Classic IOS Firewall
- IPv6 support in the Zone Policy Firewall
- Handling ASA IPv6 ACLs and Object-Groups
- Stateful Inspection of IPv6 in ASA
- Establishing Connection Limits
- IPv6 and Antispoofing
- IPv6 and Fragmentation

“Change is not made without inconvenience, even from worse to better.” —Richard Hooker

The vast territory of individual firewall features has been carefully explored throughout the previous chapters. During this long (but, hopefully, compensatory) journey, IPv4 has always been assumed as the L3 protocol in place, no matter if the resources under analysis were stateful or stateless in nature. The present chapter builds upon the functionality knowledge acquired up to now as a reference point to explore the IPv6 features currently supported on Cisco Firewalls.

Why dedicate a chapter to IPv6, the new (but yet to be established) version of the Internet Protocol? Well, not exactly because it is considered something fashionable. As

IPv6 adoption gains traction, many organizations might consider at least some kind of experimental deployment.

Note The number 2^{128} is approximately 3.4×10^{38} , which can be expressed as $(3.4 \times 10^4) \times (10^{12}) \times (10^{12}) \times (10 \times 10^9) = 34,000 \times (1 \text{ trillion}) \times (1 \text{ trillion}) \times (10 \text{ billion})$. Assuming a future population of 10 billion people on Earth, how many IPv6 addresses per person would be available?

This chapter is meant, on one hand, to help those security professionals who need to deal with the challenge of building basic IPv6 knowledge, without the time investment of reading a whole book on the topic. On the other hand, if you are already an IPv6 expert, you can still find useful information, in one place, about the specific subject of firewalls.

Introduction to IPv6

The base specification of IPv6 is documented on RFC 2460, but many other technical details of the IPv6 framework are defined on companion RFCs. For example, RFC 4291 covers the IPv6 addressing architecture, and RFC 4443 takes care of ICMPv6, a protocol responsible for many operational tasks in IPv6.

The major changes incorporated in IPv6 are summarized follows:

- **Virtually unlimited addressing capabilities:** IPv6 addresses are 128-bits long, yielding 2^{96} times the size of the IPv4 addressing space. This is a huge number, well beyond human imagination or ability to express quantities using words. And even with this almost unmeasurable space, the Internet Assigned Numbers Authority (IANA) has designed a conservative and organized process for distributing IPv6 addresses to the regional registries (such as the ARIN and the RIPE NCC). Because the allocation is hierarchical (with IANA at the top level), this not only provides visibility of addressing needs, as IPv6 deployments increase, but also renders address aggregation tasks easier. This particular aspect of IPv6 became even more attractive after the IANA's announcement, in February 2011, that the last IPv4 blocks had been allocated to the Regional Internet Registries (RIR).
- **Simplification of the base header format:** To reduce the processing effort in the most common situations of packet routing and forwarding, some fields originally present in the IPv4 header were removed.
- **New encoding model for IP options:** IP options are now carried as purpose-specific extension headers and not as a part of the base header. This enables more efficient forwarding, less strict limits on the length of options, and greater flexibility for defining future options.
- **Native flow labeling capability:** Designed to enable a simpler way for the source of traffic to clearly signal that some packets belong to particular traffic flows, for which special handling is required.

- **Authentication and privacy capabilities:** IPv6 specifies extensions to support authentication, data integrity verification, and confidentiality.

Among the requirements considered during the conception and design phases of IPv6, one of particular significance is the capability to deliver on the promise to provide a global network environment. Ideally, in this new world, complete transparency in the access of users to applications of interest would be ensured, irrespectively of any Layer 3 addressing constraints. In a scenario obeying such boundary conditions, Network Address Translation (NAT) could not be part of the plan.

At this point, many readers might wonder why protocol designers are so ungrateful to NAT, a technique that has provided valuable services all over the Internetworking era. Well, this is not actually the case.

Among the NAT usages discussed in Chapter 8, “Through ASA Using NAT,” the most critical one is that of *mitigating the problem of IPv4 address depletion*. Nevertheless, many security and networking professionals tend to consider *hiding private addresses from the public Internet* (with the aid of PAT) as the most relevant application of NAT. This creates a perception (and even a myth) that securely connecting an organization to the Internet presupposes source address hiding. With that said, it is important to guarantee that you never forget the classic shortcomings introduced by NAT and PAT:

- Given that application protocols are typically not constructed with address translation in mind, time and money need to be invested to adapt such protocols to environments in which NAT is required. On one hand this imposes the burden of developing *fixups* on firewalls and routers. On the other hand, it might severely compromise the timely deployments of applications.
- NAT prevents the full usage of IPsec authentication functionality provided by the AH (Authentication Header) specification.

As discussed with rough calculations previously, address space is not a concern for IPv6, thus eliminating the main motivation for NAT. So, why not profit from the freedom IPv6 provides for transparency and get rid of NAT?

Overview of IPv6 Addressing

The first time I had the chance to type an IPv6 address, I felt just like starting to learn a new foreign language: You need to work hard, from the beginning, to become acquainted with those sounds that do not exist in your native language. It is also necessary to create mental associations between phonemes and the letters that represent them. Otherwise you can never achieve a good pronunciation.

Note At the time of this writing, there was no NAT specification for IPv6 network environments. This might change in the future because of pressure exerted by fundamentalist

customers who cannot conceive of the Internet without address hiding. I do hope that you, as someone who understands the issues pertaining to NAT, do not join this group.

If you are not used to this new IPv6 language, this section provides a quick review of IPv6 address format and types of addresses. The various types of addresses and the processes in which they appear are discussed in the section “*IPv6 Connectivity Basics*.”

IPv6 addresses consist of 128 bits organized as eight fields of four hexadecimal digits each (16 bits per field), separated by colons. The basic format is X1:X2:X3:X4:X5:X6:X7:X8, in which Xi is 16-bits long, but always represented in hexadecimal notation. The basic rules for building IPv6 addresses follow:

- The hexadecimal digits are case-insensitive.
- Leading zeros within a field may be omitted. For instance, 2001:0db8:000A:00BB:0011:0222:000C:0D0D can be written as 2001:db8:A:BB:11:222:C:D0D without any loss of information.
- Successive fields containing only zeros can be represented as a double colon (::). To avoid ambiguity, the double colon is allowed only once in the address. Some examples follow:
 - 2001:db8::5 is equivalent to 2001:db8:0000:0000:0000:0000:0000:0005
 - The *loopback address* is written as ::1 or ::1/128 (The first 127 bits are zero.)
 - The *unspecified address* is composed of 128 zeros and represented as :: or ::/128
- IPv6 always uses prefix-lengths rather than subnet masks to represent the network portion of a given address. For example, the notation 2001:db8::10/64 means that 64 bits specify the subnet within this address. Further, the host address is ‘10’ in the subnet 2001:db8:0:0::/64 (or, simply, 2001:db8::/64).

Figure 16-1 shows the formats of the categories of IPv6 unicast addresses and the method employed to derive modified EUI-64 interface identifiers from the MAC address:

- **Link-local addresses:** Designed for use in a single link, these are employed on tasks such as automatic address configuration and neighbor discovery or in situations when no routers are present. Routers must not forward any packets containing link-local source or destination addresses to other links. These addresses are automatically created on IPv6-enabled interfaces by combining the link-local prefix, FE80::/10, with a sequence of 54 zeros and a 64-bit interface identifier.
- **Global unicast addresses:** Employed for generic IPv6 addressing and conceived with the possibility of aggregation in mind. Defined in RFC 3587.
- **Unique local unicast addresses:** Specified in RFC 4193 and intended for *internal* IPv6 communications, these addresses are not expected to be routable on the global Internet. FC00::/7 is the prefix reserved for this category of addresses, which uses a *pseudo-randomly* generated (40-bits long) Global ID.

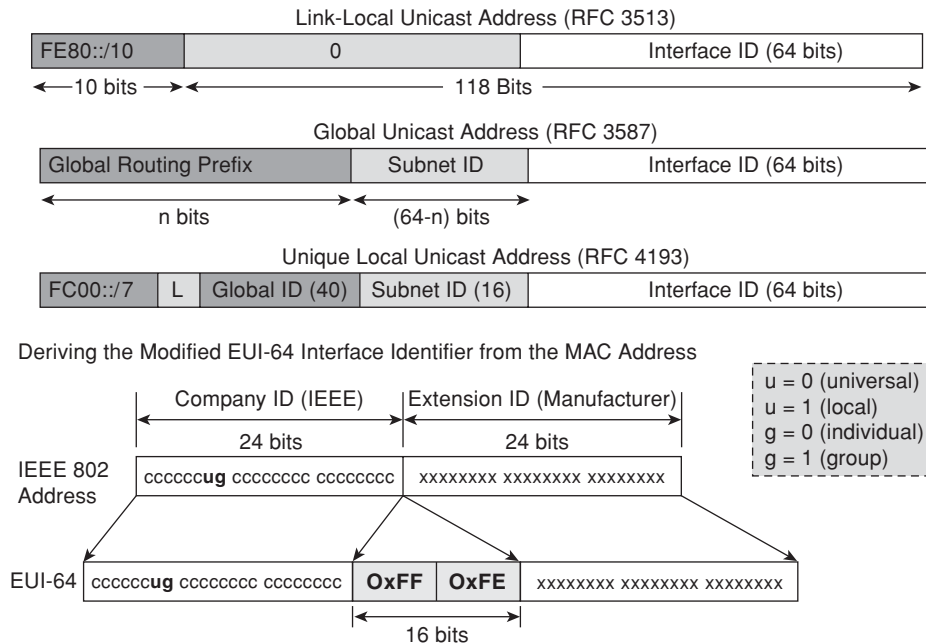


Figure 16-1 Main Types of IPv6 Addresses

- The L bit assumes the value 1 for locally assigned Global IDs (prefix FD00::/8).
- The value 0 for the L (resulting in the prefix FC00::/8) is reserved for a possible new way of allocating the Global ID.
- **Interface identifiers in IPv6 unicast addresses:** Used to identify interfaces on a link. They must be unique in a given subnet but can be reused on multiple interfaces on a single node (as long as they are part of different subnets). All unicast addresses, with the exception of those that start with binary 000, require interface IDs to be built in Modified EUI-64 format.

Note EUI stands for Extended Unique Identifier.

One important characteristic of the IPv6 framework is that broadcast addresses have been completely eliminated (and, whenever necessary, replaced by multicast addresses). For example, tasks such as Layer 2 address resolution and Duplicate Address Detection (DAD) employ multicast.

Figure 16-2 documents the format of the two classes of IPv6 multicast addresses:

- **Permanently assigned multicast addresses:** Defined in RFC 4291, these *well-known* addresses use 112 bits for the Group ID and a Flags field that contains the fixed binary value 0000.

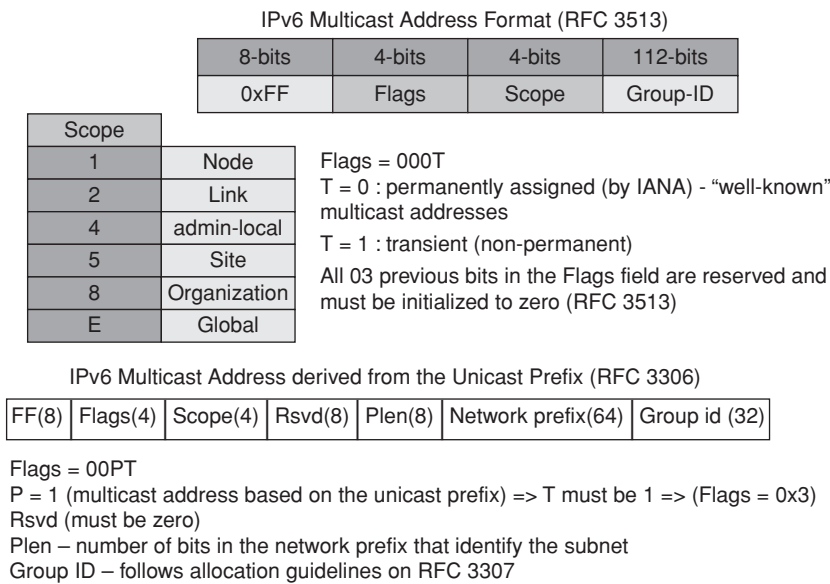


Figure 16-2 *Format of IPv6 Multicast Addresses*

- **Temporarily assigned multicast addresses:** These addresses are identified by the T and P bits being simultaneously set to 1 within the Flags field. RFC 3306 defines a method for dynamically deriving multicast addresses from the IPv6 unicast prefix. In this format, the length of the Group ID is 32 bits and the *plen* parameter determines the number of bits in the *Network Prefix* field that correspond to the subnet portion.

In any of these two addressing formats, multicast addresses are characterized by the FF00::/8 prefix. Also, the *scope* parameter defines the reach of this multicast address or, in other words, the region of the network in which this address is valid. The meaning of the scopes represented in Figure 16-2 follows:

- **node-local:** Spans a single interface on a node and is useful solely for loopback transmission of multicast.
- **link-local:** Multicast addresses whose usage is limited to a single link. Extensively employed for Neighbor Discovery (ND) activities.
- **admin-local:** Addresses with this scope cannot be automatically derived from physical connectivity or nonmulticast-related configuration.
- **site-local:** Constrained to a single site.
- **organization-local:** Span multiple sites that are part of the same organization.

RFC 4291 establishes that IPv6 routers must not forward a multicast packet beyond the scope defined in its destination address. Another important rule is that multicast addresses must not be used as source addresses in IPv6 packets.

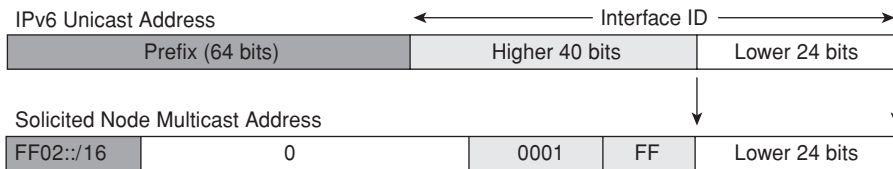
Table 16-1 *Examples of Well-Known Multicast Addresses*

Address	Scope	Meaning
FF01::1	Node-local	All-Nodes
FF02::1	Link-local	All-Nodes
FF01::2	Node-local	All-Routers
FF02::2	Link-local	All-Routers
FF05::2	Site-local	All-Routers
FF05::101	Site-local	All NTP Servers
FF05::1:3	Site-local	All DHCP Servers

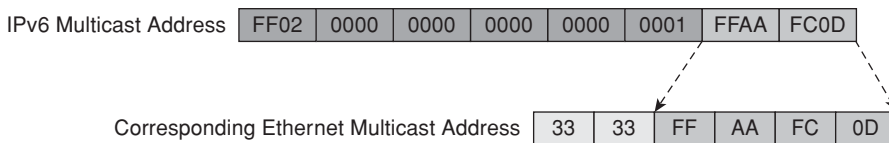
The addresses with format FF0X:: (with X being an HEX digit between 0 and F) are well-known (reserved) multicast addresses. Some important examples of addresses belonging to this group are presented as a reference in Table 16-1. The address FF0X::2 always represents some type of *All-Routers* multicast address. Setting the value of the X digit will determine the scope. A similar reasoning applies to any other reserved address FF0X::Y.

Figure 16-3 illustrates two other important processes related to multicast addresses:

Deriving the Solicited Node Multicast Address from the IPv6 Unicast Address



Deriving Ethernet Multicast Address (L2) from the IPv6 Unicast Address (L3)

**Figure 16-3** *Deriving the Solicited Node and the Ethernet Multicast Addresses*

- **How to obtain the solicited-node address:** This special type of multicast address has a link-local scope and appears in messages used for activities such as Layer 2 address resolution. The solicited-node multicast address of a node is derived by juxtaposing the lower 24 bits of its unicast address to the prefix FF02::1:FF/104.

- **How to obtain the multicast Ethernet address from the IPv6 unicast address:** These MAC addresses are built by adding the prefix 33:33 to the last 32 bits of the IPv6 multicast address.

Note The IPv6 addressing architecture includes a special entity known as an *anycast* address, which is assigned to more than one interface (generally on different nodes). A packet sent to an anycast address is routed to the *nearest* interface configured with that address (where the *distance* is calculated from a routing protocol standpoint). For more information about anycast, refer to RFC 4291.

Note An IPv6 node is required to compute and join the solicited-node multicast address corresponding to each unicast (or anycast) address configured in its interfaces.

IPv6 Header Format

The fields contained in the header of a protocol tell a lot about its operational capabilities and flexibility. The way a packet with such a header is processed by network elements (hosts or routers) can also provide insight about potential protocol vulnerabilities that, if exploited, might lead to security issues. That's why it is important to pay attention to header elements whenever a new protocol is introduced.

Figure 16-4 portrays not only the base IPv6 header, which is 40 bytes long, but also shows sample packets carrying extension headers. If you need to recall the IPv4 header, refer to Chapter 11, "Additional Protection Mechanisms." With a little reflection, you can see that the fields that consume more space in this new header are the source and destination addresses (16 bytes each).

A brief description of the fields that constitute the IPv6 header follows:

- **Version:** Contains the value 6, rather than 4 (which is associated to IPv4).
- **Traffic Class:** Similar to the Type of Service field in IPv4 and used for QoS purposes.
- **Flow Label:** Used to label sequences of packets for which special treatment is being requested. This is a new field that has no IPv4 counterpart.
- **Payload Length:** Describes the length of the IPv6 payload (not including the base header which has a fixed size of 40 bytes).
- **Next Header:** A field that resembles the IPv4 *Protocol Type* field. It typically defines a transport layer packet or indicates the presence of an *extension header*.
- **Hop Limit:** Specifies the number of hops that an IPv6 is allowed to traverse. This is analogous to the role played by the Time-To-Live (TTL) field in IPv4.
- **Source and Destination Addresses:** Well, these are self-explanatory.

IPv6 Base Header (40 Bytes)

Version (4 bits)	Traffic Class (8 bits)	Flow Label (20 bits)	
Payload Length (16 bits)		Next Header (8 bits)	Hop Limit (8 bits)
Source Address (128 bits)			
Destination Address (128 bits)			

IPv6 base Header Next Header = 6 (TCP)	TCP Header + Data		IPv6 Packet with no Extension Header
IPv6 base Header Next Header = 43 (Routing)	Routing Header Next Header = 58 (ICMP)	ICMP Header + Data	IPv6 Packet containing the Routing Extension Header
IPv6 base Header Next Header = 44 (Fragment)	Fragmentation Header Next Header = 17 (UDP)	UDP Header + Data	IPv6 Packet containing the Fragmentation Extension Header

Figure 16-4 IPv6 Base Header and Sample Extension Headers

Chapter 11 presented a basic review of *IPv4 options*, some of the issues associated to their use and how Cisco Firewalls take care of them. If you refer back to Figure 16-4, you can see that there is no equivalent to the *Options* field in the regular IPv6 header.

This happens because IPv6 implements the features corresponding to IP options by means of a set of extra headers named *extension headers*. The length of each extension header is an integer multiple of 8 bytes to guarantee 8-octet alignment for subsequent headers. When a packet contains multiple extension headers, they must appear in the following order:

1. Hop-by-hop options header (next header = 0)
2. Destination options header, when the routing header is present (next header = 60)
3. Routing header (next header = 43)
4. Fragmentation header (next header = 44)
5. Authentication header (next header = 51)
6. Encapsulating security payload header (next header = 50)
7. Mobility header (next header = 135)
8. Destination options header, when there is no routing header
9. Transport layer header (also known as *upper-layer* header)

Note The type 0 Routing Header (RH0) is an extension header that includes a set of intermediate hops through which the packet travels, in a similar manner to what is done with the IPv4 Source Route options. The RH0 might even contain the address of a given hop multiple times, a characteristic that can be exploited for traffic amplification and result in Denial of Service (DoS) situations. To deal with the threat posed by the RH0, RFC 5095 deprecated the use of this header.

Note A complete description of the extension headers is beyond the scope of this book. One notable exception is the fragmentation header, which is detailed in the section “IPv6 and Fragmentation.”

Tip If you need extra information about extension headers, refer to RFC 2460.

IPv6 Connectivity Basics

Having analyzed the IPv6 addressing and header structures, it is now time to employ this knowledge for practical configurations. Before going through the examples, though, review the role of ICMPv6 with respect to IPv6, particularly the fundamental Neighbor Discovery (ND) mechanisms it materializes.

The diagnostic and error reporting services delivered by ICMP within the IPv4 world are still available in IPv6 by means of ICMPv6, a protocol identified by a Next Header value of 58 and whose baseline operations are specified in RFC 4443.

If you, as a security or network administrator, had that classic conditioning to filter out ICMP traffic, regardless of the message types, you need to review these concepts when dealing with IPv6. There are ICMPv6 messages that are indispensable for some link-local processes related to ND within the IPv6 framework.

Figure 16-5 registers two of these new ICMP messages:

- **Router Advertisements (RA messages):** These messages, identified by ICMP type 134, are sent periodically or in response to a Router Solicitation message. An RA is sent to the All-Nodes multicast address with a link-local scope (FF02::1) and includes one or more network prefixes and possibly other data, such as the default router and the MTU on that link. RA messages are used for Stateless Autoconfiguration of hosts, a process that is later shown on Example 16-3.
- **Router Solicitation (RS messages):** These messages, which use ICMP type 133, are generated by hosts at boot time to request immediate sent of RAs by routers, so that they do not need to wait for the periodic RA and can promptly start Autoconfiguration.

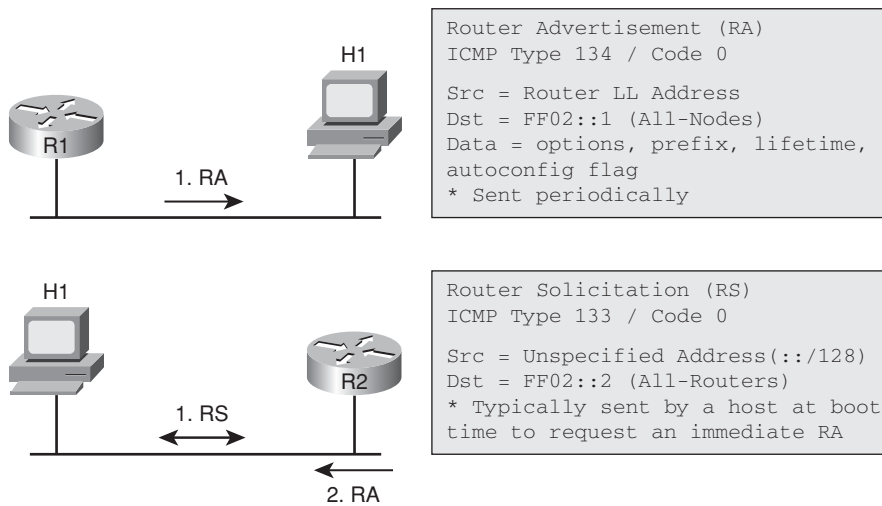


Figure 16-5 Some Usages of ICMPv6 RA and RS Messages

Note Notice on Figure 16-5 that RS messages are sent to the All-Routers link-local address, whereas RAs are directed to the All-Nodes link-local address.

Figure 16-6 presents two other important ICMPv6 ND messages, Neighbor Solicitation (NS) and Neighbor Advertisement (NA), and illustrates two classic situations in which they are employed:

- **Duplicate Address Detection (DAD):** To avoid address conflicts on a link, a node calls the DAD mechanism whenever a new address is configured. For instance, if host H1 wants to configure the unicast address X1, it sends an ICMPv6 Neighbor Solicitation (ICMP Type 135) to the solicited-node multicast address corresponding to X1. If any other host on the link responds, it means that the intended address is already in use and cannot be assigned.
- **Layer 2 Address Resolution (ARP Replacement):** If host H1 wants to send a packet to H2 on a local-link, it first needs to determine the layer 2 address of H2. H1 accomplishes that by sending a NS message to the solicited-node address corresponding to H2. The data portion of the NS contains the Query “what is your L2 address?” H2 then sends a Neighbor Advertisement message (ICMP Type 136) to H1, revealing its Layer 2 address. The hosts can now communicate using IPv6 unicast addresses.

Note This was a just a brief analysis of the main Neighbor Discovery (ND) messages, which are enough for the objectives of this chapter. If you need to learn more about this subject, refer to RFC 2461.

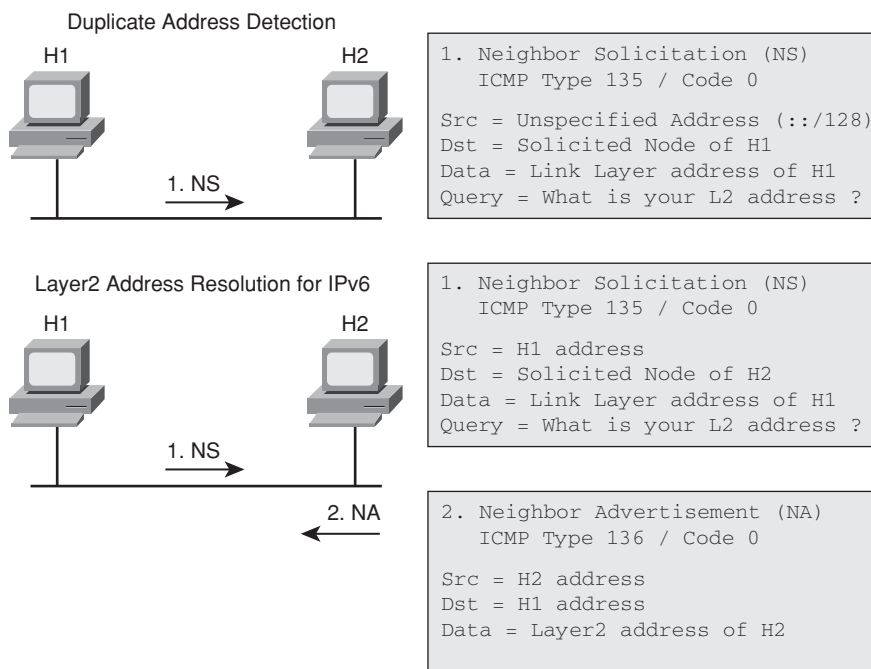


Figure 16-6 *Some Usages of ICMPv6 NS and NA Messages*

The examples presented in this section can certainly contribute for consolidation of the theoretical aspects discussed so far. Figure 16-7 depicts the reference topology for the study of Link-local addresses. For each of the network elements represented in it, a set of addresses has been obtained from the MAC address, following the procedures explained in the section “Overview of IPv6 Addressing.”

Example 16-1 starts by showing how to globally enable IPv6 on a Cisco router and the optimized forwarding achieved by Cisco Express Forwarding (CEF). As soon as the command **ipv6 enable** is applied to an interface, the router invokes the Duplicate Address Detection (DAD) process, to avoid address conflicts, and the IPv6 interface comes up with the Link-local unicast address automatically derived from the MAC address. Because the particular host in this example is a router, it starts sending RA messages sourced from the Link-local address and destined to the All-Nodes multicast group (FF02::1).

The example also registers the data provided by the **show ipv6 interface** command:

- Up to now, this node has only a link-local unicast address (and no global unicast addresses defined).
- The three multicast-groups joined are shown:
 - [FF02::1]: All-Nodes (After all, this router is also a host.)

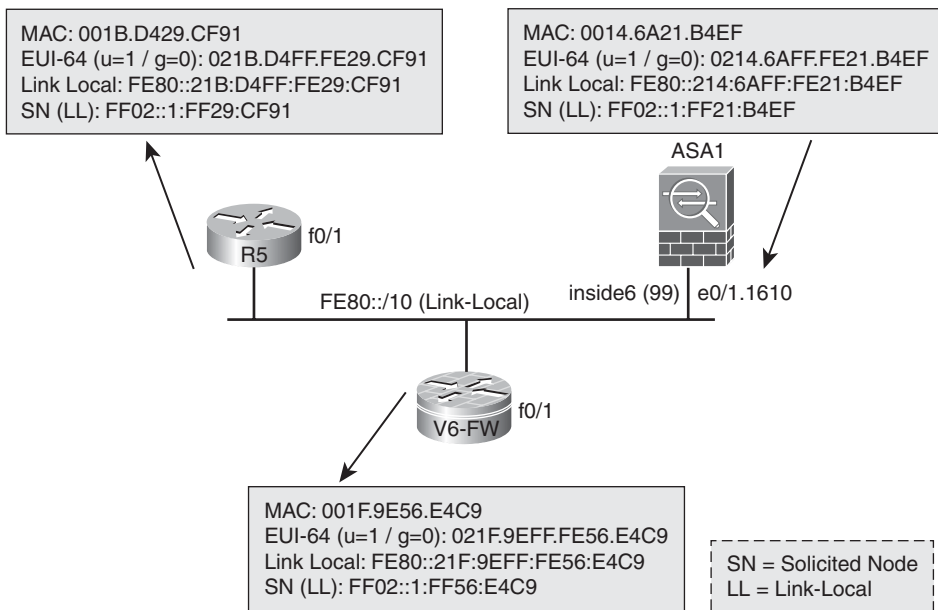


Figure 16-7 Reference Topology for the Study of Link-Local IPv6 Addresses

- [FF02::2]: All-Routers
- [FF02::1:FF56:E4C9]: solicited-node corresponding to the link-local address
- Information about the periodicity of ICMPv6 messages.

Example 16-1 Enabling IPv6 on a Cisco IOS Router

```
! Enabling IPv6 Globally
V6-FW(config)# ipv6 unicast-routing
V6-FW(config)# ipv6 cef
!
! Enabling IPv6 on a particular interface
V6-FW(config)# interface f0/1
V6-FW(config-if)# ipv6 enable
ICMPv6-ND: IPv6 Opr Enabled on FastEthernet0/1
ICMPv6-ND: Allocate ND subblock on FastEthernet0/1 [5]
ICMPv6-ND: L2 came up on FastEthernet0/1
IPv6-Addrmgr-ND: DAD request for FE80::21F:9EFF:FE56:E4C9 on FastEthernet0/1
ICMPv6-ND: Sending NS for FE80::21F:9EFF:FE56:E4C9 on FastEthernet0/1
IPv6-Addrmgr-ND: DAD: FE80::21F:9EFF:FE56:E4C9 is unique.
ICMPv6-ND: Sending NA for FE80::21F:9EFF:FE56:E4C9 on FastEthernet0/1
ICMPv6-ND: L3 came up on FastEthernet0/1
```

```

ICMPv6-ND: Linklocal FE80::21F:9EFF:FE56:E4C9 on FastEthernet0/1, Up
!
! Router Advertisement (RA) is sent to the All-Nodes Multicast Address (Link-local)
ICMPv6-ND: Created RA context for FE80::21F:9EFF:FE56:E4C9
ICMPv6-ND: Request to send RA for FE80::21F:9EFF:FE56:E4C9
ICMPv6-ND: Sending RA from FE80::21F:9EFF:FE56:E4C9 to FF02::1 on FastEthernet0/1
ICMPv6-ND:      MTU = 1500
!
! Determining the MAC address of a given interface

V6-FW# show interface f0/1 | include bia
      Hardware is MV96340 Ethernet, address is 001f.9e56.e4c9 (bia 001f.9e56.e4c9)
!
V6-FW# show ipv6 interface f0/1
FastEthernet0/1 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::21F:9EFF:FE56:E4C9
  No Virtual link-local address(es):
No global unicast address is configured
Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF56:E4C9
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ICMP unreachable are sent
ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds (using 30000)
  ND advertised reachable time is 0 (unspecified)
  ND advertised retransmit interval is 0 (unspecified)
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  ND advertised default router preference is Medium
  Hosts use stateless autoconfig for addresses.

```

Example 16-1 taught that the Link-local address configuration process starts automatically, just after an interface is enabled for IPv6. Example 16-2 shows what happens when a Global Unicast address is configured:

- DAD takes place in the same way as for link-local addresses.
- Given the uniqueness of the address, it is assigned to the interface, as revealed by the **show ipv6 interface** command. This command also shows that the configuration of a global unicast address does not affect the link-local address. An interface enabled for IPv6 might have several addresses at the same time (either static or dynamic). The

solicited-node address (FF02::1:FF00:1) corresponding to the global unicast address also displays.

- Two routes associated to Fast0/1 appear in the IPv6 routing table, one connected (/64) and one local (/128). This last one represents the IPv6 address on the interface.

Example 16-2 *Configuring a Global Unicast Address on IOS*

```
V6-FW(config)# interface f0/1
V6-FW(config-if)# ipv6 address 2001:db8::1/64
IPv6-Admrgr-ND: DAD request for 2001:DB8::1 on FastEthernet0/1
ICMPv6-ND: Sending NS for 2001:DB8::1 on FastEthernet0/1
IPv6-Admrgr-ND: DAD: 2001:DB8::1 is unique.
ICMPv6-ND: Sending NA for 2001:DB8::1 on FastEthernet0/1
!
V6-FW# show ipv6 interface f0/1
FastEthernet0/1 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::21F:9EFF:FE56:E4C9
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8::1, subnet is 2001:DB8::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF00:1
    FF02::1:FF56:E4C9
!
! Initial IPv6 Routing Table

V6-FW# show ipv6 route
IPv6 Routing Table - default - 3 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
        B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
        I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
        D - EIGRP, EX - EIGRP external, NM - NEMO, ND - Neighbor Discovery
        O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
        ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
C    2001:DB8::/64 [0/0]
    via FastEthernet0/1, directly connected
L    2001:DB8::1/128 [0/0]
    via FastEthernet0/1, receive
L    FF00::/8 [0/0]
    via Null0, receive
```

Note The prefix 2001:db8::/32, reserved for documentation purposes (RFC 3849), is used throughout this chapter in all examples that contain global unicast addresses.

Tip The `show ipv6 route` command shows only the global (default) table. If you need to display the link-local route tables on a Cisco IOS Router, use the following command (in which *Null0* and *LI-Null0* are software interfaces):

V6-FW# `show ipv6 route table`

IPv6 Routing - 4 tables

1 tables of global scope, 3 tables of link-local scope

Table id	Scope	Name
12000005	link-local	FastEthernet0/1
12000002	link-local	LI-Null0
12000001	link-local	Null0
0	global	default

If you want to see a specific table, just include the Table id parameter:

HUB1# `show ipv6 route table 12000005 | begin \[`

```
C FE80::/10 [0/0]
    via FastEthernet0/1, directly connected
L FE80::21F:9EFF:FE56:E4C9/128 [0/0]
    via FastEthernet0/1, receive
```

Figure 16-8 brings the reference topology used in Examples 16-3 to 16-6. It also summarizes the global and solicited-node addresses for the network elements represented in this scenario.

Example 16-3 was conceived to illustrate the concepts of Prefix Advertisement and Stateless Autoconfiguration. First, the advertisement of the default ND prefix for this link (2001:db8::/64) is suspended on V6-FW. (This prefix starts being announced via RA messages as soon as V6-FW is configured with the Global address of Example 16-2.) V6-FW is then configured to advertise the new prefix (2001:db8:AA:AA::/64) on the link represented by Fast0/1. Some noteworthy facts in this example follow:

- The typical contents of an RA message are visible.
- The `show ipv6 interface prefix` command provides information about suppressed and announced prefixes on a particular IPv6 interface.

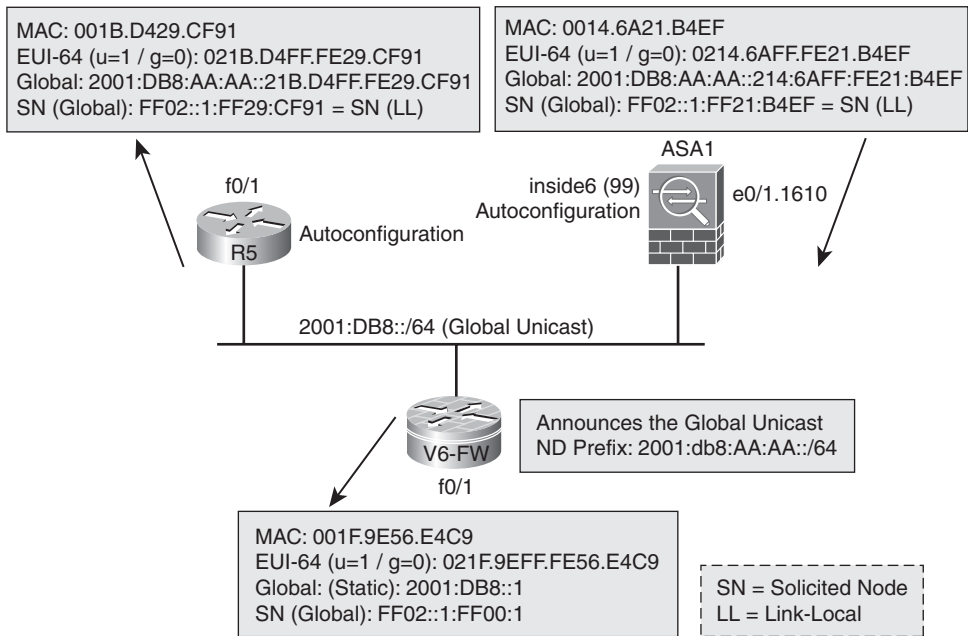


Figure 16-8 Reference Topology for the Analysis of IPv6 Global Unicast Addresses

- When instructed to use stateless autoconfiguration, R5 sends a Router Solicitation message to the All-Routers address (FF02::2) and receives the corresponding RA in the All-Nodes address (FF02::1). This RA includes the new prefix and the associated lifetime. The complete address is obtained by combining the received prefix with the EUI-64 Interface ID.
- R5 does not advertise the received prefix on the link.

Example 16-3 Prefix Advertisement and Stateless Autoconfiguration

```
! Suspending advertisement of the prefix 2001:db8::/64
V6-FW(config)#interface f0/1
V6-FW(config-if)#ipv6 nd prefix 2001:db8::/64 no-advertise
!
! Advertising the new prefix 2001:DB8:AA:AA::/64 on interface F0/1
V6-FW(config-if)# ipv6 nd prefix 2001:DB8:AA:AA::/64
ICMPv6-ND: Request to send RA for FE80::21F:9EFF:FE56:E4C9
ICMPv6-ND: Sending RA from FE80::21F:9EFF:FE56:E4C9 to FF02::1 on FastEthernet0/1
ICMPv6-ND: MTU = 1500
ICMPv6-ND: prefix = 2001:DB8:AA:AA::/64 onlink autoconfig
ICMPv6-ND: 2592000/604800 (valid/preferred)
!
```

```

V6-FW# show ipv6 interface f0/1 prefix
IPv6 Prefix Advertisements FastEthernet0/1
Codes: A - Address, P - Prefix-Advertisement, O - Pool
       U - Per-user prefix, D - Default
       N - Not advertised, C - Calendar
PD default [LA] Valid lifetime 2592000, preferred lifetime 604800
PAN 2001:DB8::/64 [LA] Valid lifetime 2592000, preferred lifetime 604800
PD 2001:DB8:AA:AA::/64 [LA] Valid lifetime 2592000, preferred lifetime 604800
!
! R5 is instructed to use Stateless Autoconfiguration

R5(config)# interface f0/1
R5(config-if)# ipv6 address autoconfig
ICMPv6-ND: Sending RS on FastEthernet0/1
ICMPv6: Sent R-Solicit, Src=FE80::21B:D4FF:FE29:CF91, Dst=FF02::2
ICMPv6: Received R-Advert, Src=FE80::21F:9EFF:FE56:E4C9, Dst=FF02::1
ICMPv6-ND: Received RA from FE80::21F:9EFF:FE56:E4C9 on FastEthernet0/1
IPv6-Address: Prefix Information change for 2001:DB8:AA:AA::/64, 0x0 -> 0xE0
IPv6-Address: Adding prefix 2001:DB8:AA:AA::/64 to FastEthernet0/1
IPv6-Address: Address 2001:DB8:AA:AA:21B:D4FF:FE29:CF91 configured on
FastEthernet0/1
IPv6-Addrmgr-ND: DAD request for 2001:DB8:AA:AA:21B:D4FF:FE29:CF91 on
FastEthernet0/1
ICMPv6-ND: Sending NS for 2001:DB8:AA:AA:21B:D4FF:FE29:CF91 on FastEthernet0/1
ICMPv6: Sent N-Solicit, Src=::, Dst=FF02::1:FF29:CF91
ICMPv6-ND: Autoconfiguring 2001:DB8:AA:AA:21B:D4FF:FE29:CF91 on FastEthernet0/1
IPv6-Addrmgr-ND: DAD: 2001:DB8:AA:AA:21B:D4FF:FE29:CF91 is unique.
!
! Results on R5

R5# show ipv6 interface f0/1
FastEthernet0/1 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::21B:D4FF:FE29:CF91
  No Virtual link-local address(es):
  Stateless address autoconfig enabled
  Global unicast address(es):
    2001:DB8:AA:AA:21B:D4FF:FE29:CF91, subnet is 2001:DB8:AA:AA::/64 [EUI/CAL/PRE]
      valid lifetime 2591818 preferred lifetime 604618
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF29:CF91
!
R5# show ipv6 interface f0/1 prefix
IPv6 Prefix Advertisements FastEthernet0/1

```

Codes: A - Address, P - Prefix-Advertisement, O - Pool
 U - Per-user prefix, D - Default
N - Not advertised, C - Calendar

PD default [LA] Valid lifetime 2592000, preferred lifetime 604800
 AN 2001:DB8:AA:AA::/64 [LA] Valid lifetime 2592000, preferred lifetime 604800

Example 16-4 is the ASA counterpart of Example 16-1. The Link-local address appears just after enabling IPv6 on the interface. For the interface to work properly on ASA, you need to assign a logical name to it using the **nameif** command. (This is identical to ASA behavior for IPv4.)

Example 16-4 Enabling IPv6 on ASA

```
ASA1(config)# interface e0/1.1610
ASA1(config-subif)# ipv6 enable
%ASA-6-302020: Built outbound ICMP connection for faddr ff02::1:ff21:b4ef/0
gaddr ::/0 laddr ::/0
ICMPv6-ND: Sending NS for fe80::214:6aff:fe21:b4ef on inside6
ICMPv6-ND: DAD: fe80::214:6aff:fe21:b4ef is unique on inside6

%ASA-6-302020: Built outbound ICMP connection for faddr ff02::1/0
gaddr fe80::214:6aff:fe21:b4ef/0 laddr fe80::214:6aff:fe21:b4ef/0.
ICMPv6-ND: Sending NA for fe80::214:6aff:fe21:b4ef on inside6
!
ASA1# show ipv6 interface inside6
inside6 is up, line protocol is up
  IPv6 is enabled, link-local address is fe80::214:6aff:fe21:b4ef
  No global unicast address is configured
  Joined group address(es):
    ff02::1
    ff02::2
    ff02::1:ff21:b4ef
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds
  ND advertised reachable time is 0 milliseconds
  ND advertised retransmit interval is 1000 milliseconds
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
!
ASA1# show interface inside6 | include MAC
      MAC address 0014.6a21.b4ef, MTU 1500
```


Example 16-5 relates to the network scenario of Figure 16-8 and documents three methods for obtaining a Global Unicast address for an ASA IPv6-enabled interface:

- **Using Stateless Autoconfiguration:** The prefix received is announced by the V6-FW router, as in Example 16-3.
- **Static Prefix and EUI-64 Interface ID:** In this case, the **interface-level** command **ipv6 address eui-64** instructs ASA to use the static prefix 2001:db8::/64 with a dynamically generated EUI-64 ID.
- **Manually configured prefix and static host address:** This is completely analogous to the procedure for IOS on Example 16-2. One remarkable difference between IOS and ASA is that the latter shows all the IPv6 routes (link-local and global) as components of a single routing table.

Example 16-5 *Configuring a Global Unicast Address on ASA*

```
! First Method : Stateless Autoconfiguration
ASA1(config)# interface e0/1.1610
ASA1(config-subif)# ipv6 address autoconfig
ICMPv6-ND: Sending RS on inside6
ICMPv6-ND: Received RA from fe80::21f:9eff:fe56:e4c9 on inside6
ICMPv6-ND: Sending NS for 2001:db8:aa:aa:214:6aff:fe21:b4ef on inside6
ICMPv6-ND: Autoconfiguring 2001:db8:aa:aa:214:6aff:fe21:b4ef on inside6
ICMPv6-ND: DAD: 2001:db8:aa:aa:214:6aff:fe21:b4ef is unique on inside6.
ICMPv6-ND: Sending NA for 2001:db8:aa:aa:214:6aff:fe21:b4ef on inside6
!
ASA1# show ipv6 interface inside6
inside6 is up, line protocol is up
  IPv6 is enabled, link-local address is fe80::214:6aff:fe21:b4ef
Global unicast address(es):
    2001:db8:aa:aa:214:6aff:fe21:b4ef, subnet is 2001:db8:aa:aa::/64 [AUTOCONFIG]
    valid lifetime 2591845 preferred lifetime 604645
Joined group address(es):
  ff02::1
  ff02::2
  ff02::1:ff21:b4ef
!
! Second Method: Manually defined Prefix + EUI-64 host address

ASA1(config)# interface e0/1.1610
ASA1(config-subif)# ipv6 address 2001:db8::/64 eui-64
ICMPv6-ND: Adding prefix 2001:db8::/64 to inside6
ICMPv6-ND: Sending NS for fe80::214:6aff:fe21:b4ef on inside6
ICMPv6-ND: DAD: fe80::214:6aff:fe21:b4ef is unique on inside6.
ICMPv6-ND: Sending NA for fe80::214:6aff:fe21:b4ef on inside6
```

```

ICMPv6-ND: Sending NS for 2001:db8::214:6aff:fe21:b4ef on inside6
ICMPv6-ND: DAD: 2001:db8::214:6aff:fe21:b4ef is unique on inside6.
ICMPv6-ND: Sending NA for 2001:db8::214:6aff:fe21:b4ef on inside6
!
ASA1# show ipv6 interface inside6
inside6 is up, line protocol is up
    IPv6 is enabled, link-local address is fe80::214:6aff:fe21:b4ef
Global unicast address(es):
    2001:db8::214:6aff:fe21:b4ef, subnet is 2001:db8::/64
Joined group address(es):
    ff02::1
    ff02::2
    ff02::1:ff21:b4ef
!
! Third Method: Manually configured prefix + Static Host Address

ASA1(config)# interface e0/1.1610
ASA1(config-subif)# ipv6 address 2001:db8::11/64
ICMPv6-ND: Adding prefix 2001:db8::/64 to inside6
ICMPv6-ND: Sending NS for 2001:db8::11 on inside6
ICMPv6-ND: DAD: 2001:db8::11 is unique on inside6.
ICMPv6-ND: Sending NA for 2001:db8::11 on inside6
!
ASA1# show ipv6 interface inside6
inside6 is up, line protocol is up
    IPv6 is enabled, link-local address is fe80::214:6aff:fe21:b4ef
Global unicast address(es):
    2001:db8::11, subnet is 2001:db8::/64
Joined group address(es):
    ff02::1
    ff02::2
ff02::1:ff00:11
    ff02::1:ff21:b4ef
!
! ASA displays all the IPv6 routes (link-local and global) in the same table

ASA1# show ipv6 route
IPv6 Routing Table - 4 entries
Codes: C - Connected, L - Local, S - Static
L   2001:db8::11/128 [0/0]
    via ::, inside6
C   2001:db8::/64 [0/0]
    via ::, inside6
L   fe80::/10 [0/0]
    via ::, inside6

```

```
L    ff00::/8 [0/0]
    via ::, inside6
```

Example 16-6 is based on the topology of Figure 16-8 and illustrates L2 address resolution between IOS and ASA. This example complements the description previously presented in Figure 16-6.

Example 16-6 *Resolving L2 Addresses*

```
R5# ping 2001:db8::11 source 2001:db8::5 repeat 1
Packet sent with a source address of 2001:DB8::5
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
ICMPv6-ND: DELETE -> INCMP: 2001:DB8::11
ICMPv6-ND: Sending NS for 2001:DB8::11 on FastEthernet0/1
ICMPv6: Sent N-Solicit, Src=2001:DB8::5, Dst=FF02::1:FF00:11
ICMPv6-ND: Resolving next hop 2001:DB8::11 on interface FastEthernet0/1
ICMPv6: Sent echo request, Src=2001:DB8::5, Dst=2001:DB8::11
ICMPv6: Received N-Advert, Src=2001:DB8::11, Dst=2001:DB8::5
ICMPv6-ND: Received NA for 2001:DB8::11 on FastEthernet0/1 from 2001:DB8::11
ICMPv6-ND: Neighbour 2001:DB8::11 on FastEthernet0/1 : LLA 0014.6a21.b4ef
ICMPv6-ND: INCMP -> REACH: 2001:DB8::11
ICMPv6: Received echo reply, Src=2001:DB8::11, Dst=2001:DB8::5
!
! ASA1's perspective
ICMPv6: Received ICMPv6 packet from 2001:db8::5, type 135
ICMPv6-ND: Received NS for 2001:db8::11 on inside6 from 2001:db8::5
ICMPv6-ND: DELETE -> INCMP: 2001:db8::5
ICMPv6-ND: INCMP -> STALE: 2001:db8::5
ICMPv6-ND: Sending NA for 2001:db8::11 on inside6
ICMPv6-ND: STALE -> DELAY: 2001:db8::5
ICMPv6: Received ICMPv6 packet from 2001:db8::5, type 128
ICMPv6: Received echo request from 2001:db8::5
ICMPv6: Sending echo reply to 2001:db8::5
ICMPv6-ND: DELAY -> PROBE: 2001:db8::5
ICMPv6-ND: PROBE -> REACH: 2001:db8::5
!
! Resultant neighbor tables on R5 and ASA1

R5# show ipv6 neighbor f0/1
IPv6 Address                               Age Link-layer Addr State Interface
FE80::214:6AFF:FE21:B4EF                   2 0014.6a21.b4ef STALE Fa0/1
2001:DB8::11                               2 0014.6a21.b4ef STALE Fa0/1
!
ASA1# show ipv6 neighbor inside6
```

IPv6 Address	Age	Link-layer Addr	State	Interface
2001:db8::5	2	001b.d429.cf91	STALE	inside6
fe80::21b:d4ff:fe29:cf91	2	001b.d429.cf91	STALE	inside6

Example 16-7 is bound to the scenario portrayed in Figure 16-9 and was conceived to show how static routes are created and displayed on both ASA and IOS. The procedures are similar to those of IPv4, with the distinction that IPv6 always uses the *prefix-length* rather than the *network mask*, for defining the subnet portion of an address. Despite the fact that all destinations in Figure 16-9 are /64 networks, the prefix lengths configured in the example are, in most cases, less specific.

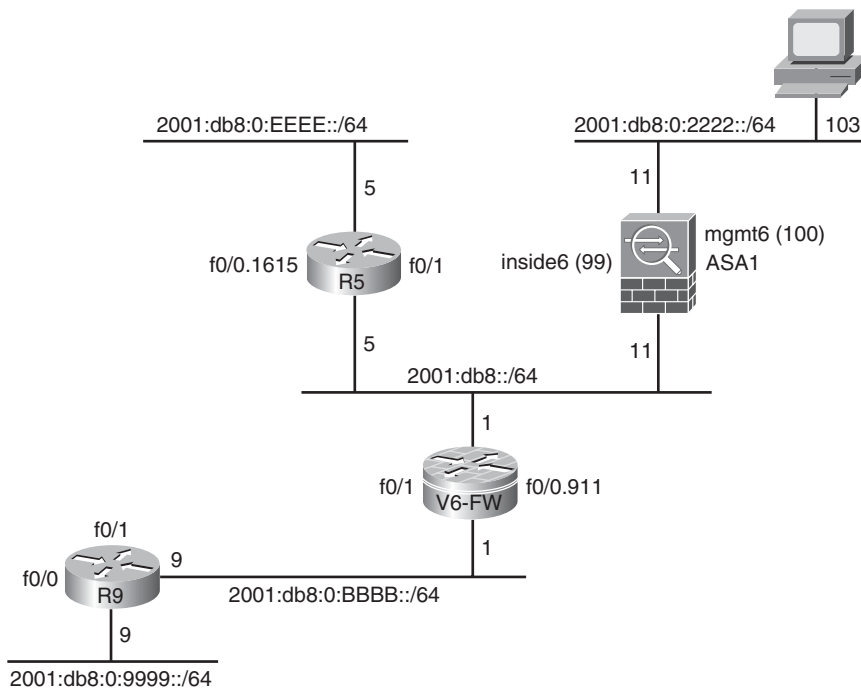


Figure 16-9 Reference Topology for IPv6 Static Routing

Example 16-7 Configuring Static Routes

```
! Adding a default route on R9
R9(config)#ipv6 route ::/0 2001:db8:0:bbb::1
!
R9# sh ipv6 route static
IPv6 Routing Table - 4 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
```

```

    U - Per-user Static route, M - MIPv6
    I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
    O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
    ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
    D - EIGRP, EX - EIGRP external
S  ::/0 [1/0]
    via FE80::21F:9EFF:FE56:E4C8, FastEthernet0/1
    via 2001:DB8:0:BBBB::1
!
! /64 route pointing to ASA1 and /32 route pointing to V6-FW

R5(config)# ipv6 route 2001:db8:0:2222::/64 2001:db8::11
R5(config)# ipv6 route 2001:db8::/32 2001:db8::1
!
! /64 route pointing to R5 and /48 route pointing to V6-FW

ASA1(config)# ipv6 route inside6 2001:db8:0:eeee::/64 2001:db8::5
ASA1(config)# ipv6 route inside6 2001:db8::/48 2001:db8::1
!
! Some routing and forwarding information on V6-FW

V6-FW# show ipv6 route static
IPv6 Routing Table - default - 5 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
        B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
        I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
        D - EIGRP, EX - EIGRP external, NM - NEMO, ND - Neighbor Discovery
        O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
        ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
S  2001:DB8:0:2222::/64 [1/0]
    via 2001:DB8::11
S  2001:DB8:0:9999::/64 [1/0]
    via 2001:DB8:0:BBBB::9
S  2001:DB8:0:EEEE::/64 [1/0]
    via 2001:DB8::5
!
V6-FW# show ipv6 route 2001:db8:0:2222::/64
Routing entry for 2001:DB8:0:2222::/64
  Known via "static", distance 1, metric 0
  Route count is 1/1, share count 0
  Routing paths:
    2001:DB8::11
      Last updated 00:01:11 ago
!
! Obtaining Forwarding Information for a specific IPv6 destination

```

```
V6-FW# show ipv6 cef 2001:db8:0:2222::103
2001:DB8:0:2222::/64
    nexthop 2001:DB8::11 FastEthernet0/1
```

Tip IOS Release 12.4(20)T introduced a new way to define virtual routing and forwarding instances (VRF) that enable virtualization of IPv6 routing tables:

```
V6-FW(config)# vrf definition VRF6
```

```
V6-FW(config-vrf)# rd 65006:6006
```

```
V6-FW(config-vrf)# address-family ?
```

```
  ipv4  Address family
```

```
  ipv6  Address family
```

The new **interface-level** command to bind an interface to a VRF is **vrf forwarding**. A VRF defined in this way might simultaneously contain IPv4 and IPv6 address-families. For a description of VRF, refer to Chapter 6, “Virtualization in the Firewall World..

Note At the time of this writing, IOS already supported several IPv6 routing protocols (such as RIP, OSPF, and EIGRP), whereas ASA implemented only IPv6 static routes. The basic principles are similar to those presented in Chapter 5, “Firewalls in the Network Topology,” but configuration of IPv6 routing protocols is well beyond the scope of this book.

Following this familiarization with the basic IPv6 connectivity aspects, it is time to adapt a useful tool (largely employed throughout the book) for use within IPv6 environments. Example 16-8 proposes a custom Netflow record aimed to provide insight about multiple fields within the IPv6 base header and on eventual upper-layer headers. It also presents the **flow exporter** and **flow monitor** configurations. One noticeable detail in this Netflow infrastructure is that the export functions are still carried out using IPv4.

Example 16-8 *Reference Netflow Structure for Use Within This Chapter*

```
! Custom Flow Record contains IPv6 parameters
flow record FLEXRECORD6
  match ipv6 traffic-class
  match ipv6 protocol
  match ipv6 source address
  match ipv6 destination address
  match transport source-port
  match transport destination-port
  match interface input
  collect routing next-hop address ipv6
```

```

collect ipv6 next-header
collect ipv6 hop-limit
collect ipv6 payload-length
collect ipv6 extension map
collect ipv6 fragmentation flags
collect ipv6 fragmentation offset
collect ipv6 fragmentation id
collect transport tcp flags
collect interface output
collect counter bytes
collect counter packets
!
! Flow export still uses IPv4
flow exporter FLEXNETFLOW
description *** Exporting to Cisco MARS
destination 192.168.1.114
source FastEthernet0/0.1102
transport udp 2055
!
! Defining the Flow Monitor (to be later bound to an interface)
flow monitor FLEX6
record FLEXRECORD6
exporter FLEXNETFLOW

```

Example 16-9 applies the **flow monitor** defined in Example 16-8 to provide visibility of multicast packets that arrive at V6-FW's interface Fast0/1 in the topology of Figure 16-9. The flow cache shown contains a Router Advertisement message sent by ASA1 (using its link-local address, previously registered in Figure 16-7).

Example 16-10 shows some examples of flow aggregation according to specific fields within the IPv6 (or upper-layer) headers.

Example 16-9 *Netflow in Action*

```

! Applying the Flow Monitor only to Multicast Packets
interface FastEthernet0/1
  ipv6 flow monitor FLEX6 multicast input
!
! Displaying the flow cache

V6-FW# show flow monitor FLEX6 cache

```

Cache type:	Normal
Cache size:	4096
Current entries:	1
High Watermark:	4

```

Flows added:                                115
Flows aged:                                114
- Active timeout      ( 1800 secs)          0
- Inactive timeout    (   15 secs)        114
- Event aged                          0
- Watermark aged      0
- Emergency aged      0

```

```
IPv6 SOURCE ADDRESS:      FE80::214:6AFF:FE21:B4EF
```

```
IPv6 DESTINATION ADDRESS: FF02::1
```

```
TRNS SOURCE PORT:          0
```

```
TRNS DESTINATION PORT:     34304
```

```
INTERFACE INPUT:          Fa0/1
```

```
IP PROTOCOL:               58
```

```
IP TOS:                     0xE0
```

```
ipv6 flow label:           0
```

```
ipv6 next header:         58
```

```
ipv6 payload length:       64
```

```
ipv6 extension map:        0x00000000
```

```
ipv6 fragmentation id:     0
```

```
icmp ipv6 type:           134
```

```
icmp ipv6 code:          0
```

```
tcp flags:                  0x00
```

```
interface output:          Null
```

```
counter bytes:              104
```

```
counter packets:         1
```

```
ip ttl:                     254
```

```
ip fragmentation offset:    0
```

```
ip fragmentation flags:     0x00
```

Example 16-10 Netflow Aggregation

```

! Sample aggregation of information within the flow cache
V6-FW# show flow monitor FLEX6 cache aggregate transport icmp ipv6 type transport
icmp ipv6 code
Processed 2 flows
Aggregated to 2 flows

```

ICMP IPV6 TYPE	ICMP IPV6 CODE	flows	bytes	pkts
128	0	1	49000	49
134	0	1	104	1

```

!
```



```

! Another aggregation example (02 sources sending pings to FF02::2)

V6-FW# show flow monitor FLEX6 cache aggregate ipv6 source address transport icmp
ipv6 type transport icmp ipv6 code
Processed 3 flows
Aggregated to 3 flows
IPV6 SOURCE ADDRESS: 2001:DB8::5
ICMP IPV6 TYPE: 128
ICMP IPV6 CODE: 0
counter flows: 1
counter bytes: 86000
counter packets: 86

IPV6 SOURCE ADDRESS: FE80::214:6AFF:FE21:B4EF
ICMP IPV6 TYPE: 128
ICMP IPV6 CODE: 0
counter flows: 1
counter bytes: 5358600
counter packets: 4122

IPV6 SOURCE ADDRESS: FE80::21B:D4FF:FE29:CF91
ICMP IPV6 TYPE: 134
ICMP IPV6 CODE: 0
counter flows: 1
counter bytes: 104
counter packets: 1

```

Example 16-11 shows a sample output of the **show ipv6 traffic** command, a useful way to view traffic distribution in IPv6. This command is equally available for IOS and ASA.

Example 16-11 Viewing IPv6 Traffic Statistics

```

V6-FW# show ipv6 traffic
IPv6 statistics:
  Rcvd: 22799 total, 21568 local destination
        0 source-routed, 0 truncated
        0 format errors, 0 hop count exceeded
        0 bad header, 0 unknown option, 0 bad source
        0 unknown protocol, 0 not a router
        0 fragments, 0 total reassembled
        0 reassembly timeouts, 0 reassembly failures
  Sent: 21136 generated, 0 forwarded
        0 fragmented into 0 fragments, 0 failed
        40 encapsulation failed, 7 no route, 0 too big

```

```

0 RPF drops, 0 RPF suppressed drops
Mcast: 19320 received, 842 sent
ICMP statistics:
  Rcvd: 21568 input, 0 checksum errors, 0 too short
        0 unknown info type, 0 unknown error type
        unreachable: 0 routing, 0 admin, 0 neighbor, 0 address, 0 port
        parameter: 0 error, 0 header, 0 option
        0 hopcount expired, 0 reassembly timeout, 0 too big
        18030 echo request, 0 echo reply
        0 group query, 0 group report, 0 group reduce
        6 router solicit, 1261 router advert, 0 redirects
        1108 neighbor solicit, 1163 neighbor advert
  Sent: 21136 output, 0 rate-limited
        unreachable: 7 routing, 0 admin, 0 neighbor, 3 address, 0 port
        parameter: 0 error, 0 header, 0 option
        0 hopcount expired, 0 reassembly timeout, 0 too big
        0 echo request, 18030 echo reply
        0 group query, 0 group report, 0 group reduce
        0 router solicit, 721 router advert, 0 redirects
        1231 neighbor solicit, 1110 neighbor advert
UDP statistics:
  Rcvd: 0 input, 0 checksum errors, 0 length errors
        0 no port, 0 dropped
  Sent: 0 output
TCP statistics:
  Rcvd: 0 input, 0 checksum errors
  Sent: 0 output, 0 retransmitted

```

Handling IOS IPv6 Access Control Lists

As studied before, Access Control Lists (ACL) are an important resource not only for packet filtering but also for tasks such as traffic classification and route filtering. This section covers the options supported by IOS IPv6 ACLs.

Example 16-12 highlights the parameters frequently used when defining IOS IPv6 ACLs. They are represented by the well-known **permit**, **deny**, **remark**, and **sequence** keywords. The example also shows the available protocol keywords displayed when the **permit** option is configured under an IPv6 ACL.

Example 16-12 Basic Information About IPv6 ACLs

```

V6-FW(config)# ipv6 access-list V6-ACL1
V6-FW(config-ipv6-acl)#?

```

```
IPv6 Access List configuration commands:
default      Set a command to its defaults
deny         Specify packets to reject
evaluate     Evaluate an access list
exit         Exit from access-list configuration mode
no           Negate a command or set its defaults
permit       Specify packets to forward
remark       Access list entry comment
sequence     Sequence number for this entry
<cr>

!
! Displaying the 'permit' options (the main protocols are highlighted)
V6-FW(config-ipv6-acl)# permit?
<0-255>      An IPv6 protocol number
X:X:X:X:X/<0-128> IPv6 source prefix x:x::y/<z>
ahp          Authentication Header Protocol
any          Any source prefix
esp          Encapsulation Security Payload
host         A single source host
icmp         Internet Control Message Protocol
ipv6         Any IPv6
pcp          Payload Compression Protocol
sctp         Streams Control Transmission Protocol
tcp          Transmission Control Protocol
udp          User Datagram Protocol
```

Example 16-13 is tightly coupled with the schematics shown in Figure 16-10. The purpose is to emphasize that IPv6 ACL construction continues to employ the *from source to destination* logic that characterizes IPv4 access lists.

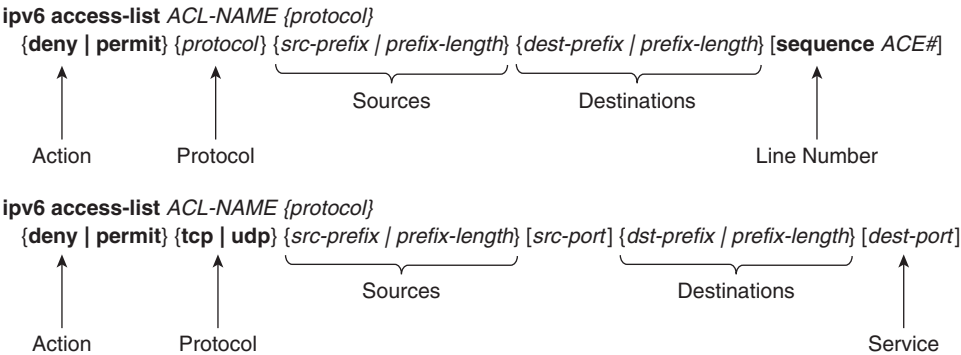


Figure 16-10 ACL Construction Logic

Example 16-13 *ACL Construction Logic*

```

! IPv6 ACLs employ the same source/destination logic as IPv4 ACLs
V6-FW(config-ipv6-acl)# permit ipv6?
X:X:X:X::X/<0-128> IPv6 source prefix x:x::y/<z>
any                Any source prefix
host               A single source host
V6-FW(config-ipv6-acl)# permit ipv6 any?
X:X:X:X::X/<0-128> IPv6 destination prefix x:x::y/<z>
any                Any destination prefix
host               A single destination host

```

Example 16-14 shows the IOS Options for IPv6 Base Header filtering and highlights the extension headers for which specific **permit** (or **deny**) actions are currently supported.

Example 16-14 *Filtering Options for IPv6 Packets*

```

! Highlighting main extension headers that can be filtered with IPv6 ACLs
V6-FW(config-ipv6-acl)# permit ipv6 any any?
auth                Match on authentication header
dest-option        Destination Option header (all types)
dest-option-type     Destination Option header with type
dscp                 Match packets with given dscp value
flow-label           Flow label
fragments           Check non-initial fragments
log                  Log matches against this entry
log-input             Log matches against this entry, including input
mobility            Mobility header (all types)
mobility-type        Mobility header with type
reflect              Create reflexive access list entry
routing             Routing header (all types)
routing-type         Routing header with type
sequence             Sequence number for this entry
time-range           Specify a time-range
<cr>

```

Figure 16-11 represents the reference network topology used for the ACL examples with-in this section.

Example 16-15 registers some basic tasks involved in ACL construction:

- A name for the ACL is defined as a parameter of the **ipv6 access-list** command.
- The filtering policy is based on **deny** and **permit** statements under the named ACL.

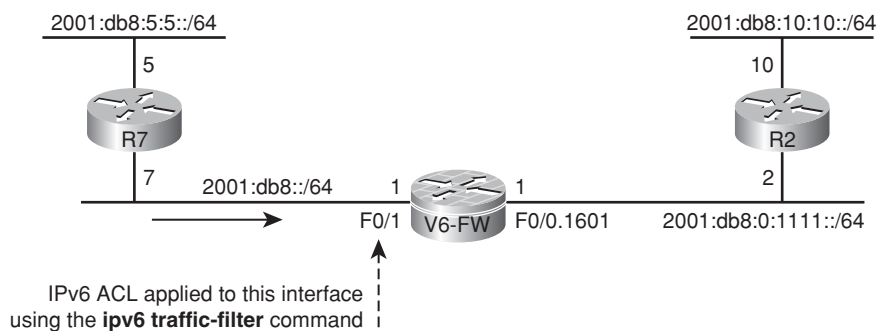


Figure 16-11 Reference Topology for the Study of IOS IPv6 ACLs

- The ACL is assigned to an interface with the **ipv6 traffic-filter** command. In this particular example, the ACL named V6-ACL2 filters incoming packets on interface Fast0/1 (router V6-FW in the topology of Figure 16-11).
- The **show access-list** command displays not only the individual ACEs but also the pertinent sequence numbers (either statically defined or automatically generated).

Example 16-15 Sample IOS ACL (1)

```
! Configuring a basic IOS IPv6 Access-list
ipv6 access-list V6-ACL2
  deny ipv6 any any flow-label 100 log-input
  deny ipv6 any any log-input routing
  sequence 1000 permit ipv6 any any
!
! Applying the IPv6 ACL to an interface
interface FastEthernet0/1
  ipv6 traffic-filter V6-ACL2 in
!
! Verifying that the ACL has been assigned to the interface
V6-FW# show ipv6 interface f0/1 | include list
  Inbound access list V6-ACL2
!
! Displaying the IPv6 ACL
V6-FW# show access-list V6-ACL2
IPv6 access list V6-ACL2
  deny ipv6 any any flow-label 100 log-input sequence 10
  deny ipv6 any any log-input routing sequence 20
  permit ipv6 any any (112 matches) sequence 1000
```

Example 16-16 shows typical editing tasks of IOS IPv6 ACLs. The starting point is the V6-ACL2 of Example 16-15:

- Using the **sequence** keyword to include ACEs in existent ACLs.
- The effect of including an ACE without specifying the **sequence** parameter and how to solve the issue.
- A possible way of adding a **remark** in a specific position. This remark does not appear in the **show access-list** command. It is visible only within the **show running-config** output.

Example 16-16 *Editing an IOS IPv6 ACL*

```
! Including entries (ACEs) in an existent IPv6 ACL
V6-FW(config)# ipv6 access-list V6-ACL2
V6-FW(config-ipv6-acl)# sequence 30 deny ipv6 any any undetermined-transport
!
! Including (by mistake) an entry without specifying the sequence number
V6-FW(config-ipv6-acl)# deny tcp any any
V6-FW(config-ipv6-acl)# do show access-list V6-ACL2
IPv6 access list V6-ACL2
    deny ipv6 any any flow-label 100 log-input sequence 10
    deny ipv6 any any log-input routing sequence 20
    deny ipv6 any any log undetermined-transport sequence 30
    permit ipv6 any any (250 matches) sequence 1000
    deny tcp any any sequence 1010
!
! Removing the undesired Access Control Entry
V6-FW(config-ipv6-acl)# no sequence 1010
V6-FW(config-ipv6-acl)# do show access-list V6-ACL2
IPv6 access list V6-ACL2
    deny ipv6 any any flow-label 100 log-input sequence 10
    deny ipv6 any any log-input routing sequence 20
    deny ipv6 any any log undetermined-transport sequence 30
    permit ipv6 any any (250 matches) sequence 1000

! Adding a remark in a specific position (using the 'sequence' keyword)

V6-FW(config)# ipv6 access-list V6-ACL2
V6-FW(config-ipv6-acl)# sequence 9 remark ** denying specific value of the Flow Label
!
! The updated access-list as it appears in the running-config
ipv6 access-list V6-ACL2
    sequence 9 remark ** denying specific value of the Flow Label
```

```
sequence 10 deny ipv6 any any flow-label 100 log-input
deny ipv6 any any log-input routing
deny ipv6 any any log undetermined-transport
sequence 1000 permit ipv6 any any
```

Example 16-17 was conceived to show the IOS V6-ACL2 of Example 16-16 in action:

- A packet containing the *Routing Header* (sent by R7 in Figure 16-11) was dropped by ACE number 20. R7 receives a destination unreachable/administratively prohibited ICMPv6 message (type 1/code 1). Because of the presence of the **log-input** option in this ACE, the source MAC address of the blocked packet is shown in the Syslog message.
- ACE number 10 blocks a TCP packet sourced from the address 2001:db8::7 (R7) and containing the value '100' in its flow-label header field.
- ACE number 30 discards some IPv6 packets that carry undetermined L4 protocols.

Example 16-17 IOS IPv6 ACLs in action (1)

```
! Generating IPv6 packets containing the Routing Header
R7# traceroute ipv6
Target IPv6 address: 2001:db8:0:1111::2
Source address: 2001:db8::7
Insert source routing header? [no]: yes
Nexthop address: 2001:db8::1
Nexthop address: <Enter>
Numeric display? [no]: <Enter>
Timeout in seconds [3]: <Enter>
Probe count [3]: <Enter>
Minimum Time to Live [1]: <Enter>
Maximum Time to Live [30]: <Enter>
Priority [0]: <Enter>
Port Number [33434]: <Enter>
Type escape sequence to abort.
Tracing the route to 2001:DB8:0:1111::2
 1 2001:DB8::1 !A !A !A

! ICMPv6 type1/code1 message (destination unreachable/administratively prohibited)
R7# ICMPv6: Received ICMPv6 packet from 2001:DB8::1, type 1
ICMPv6: Received ICMP unreachable code 1 from 2001:DB8::1

! What is seen on the IOS router
%IPV6_ACL-6-ACCESSLOGP: list V6-ACL2/20 denied udp 2001:DB8::7(53335)
(FastEthernet0/1 000f.3461.9620) -> 2001:DB8::1(33435), 1 packet
```

```

!
V6-FW# show access-list V6-ACL2 | include sequence 20
      deny ipv6 any any log-input routing (3 matches) sequence 20

! Entry #10 matches a packet whose flow-label has the value 100

%IPV6_ACL-6-ACCESSLOGP: list V6-ACL2/10 denied tcp 2001:DB8::7(20005)
(FastEthernet0/1 000f.3461.9620) -> 2001:DB8:0:1111::8(80), 1 packet
!
! Entry #30 matches undetermined L4 protocols

%IPV6_ACL-6-ACCESSLOGNP: list V6-ACL2/30 denied 1 2001:DB8::7 ->
2001:DB8:0:1111::8, 1 packet
%IPV6_ACL-6-ACCESSLOGNP: list V6-ACL2/30 denied 78 2001:DB8::7 ->
2001:DB8:0:1111::8, 1 packet

```

Note The `clear access-list counters ACL_N` command clears the hit counts for each ACE in the access control list called `ACL_N`.

Example 16-18 shows the initial configuration of an ACL called V6-ACL3, which is intended to illustrate blocking based on source/destination combinations as well as the impact of an explicit **deny any** statement as the last ACE in the access-list. After adding this ACL to the V6-FW's Fast0/1 interface on Figure 16-11, it is possible to see the following:

- In the first case, a TCP/80 packet with source address not explicitly allowed to access the server on 2001:db8:0:1111::8 is denied.
- In the second case, a valid source host tries to reach the nonallowed destination 2001:db8:10:10::10 and therefore is blocked.
- The last part of the example is the most instructive because it underlines an important aspect of IPv6 operations. Vital ICMPv6 messages (types 134, 135 and 136) are blocked by the explicit **deny any** statement in ACE #500. To overcome this problem, some new **permit icmp** statements are added before ACE #500 to cope with Neighbor Discovery tasks.

Example 16-18 IOS IPv6 ACLs in Action (2)

```

! New ACL focusing on source/destination combination
ipv6 access-list V6-ACL3
  permit ipv6 2001:DB8::/64 2001:DB8:0:1111::/64
  sequence 500 deny ipv6 any any log

! Source address not allowed

```



```
%IPV6_ACL-6-ACCESSLOGP: list V6-ACL3/500 denied tcp 2001:DB8:5:5::5(20000) ->
2001:DB8:0:1111::8(80), 1 packet
!
! Destination address not allowed

%IPV6_ACL-6-ACCESSLOGP: list V6-ACL3/500 denied tcp 2001:DB8::7(20004) ->
2001:DB8:10:10::10(80), 1 packet
!
! Some ICMPv6 messages blocked by the explicit deny on list V6-ACL3 (ACE #500)

%IPV6_ACL-6-ACCESSLOGDP: list V6-ACL3/500 denied icmpv6 FE80::214:6AFF:FE21:B4EF
-> FF02::1 (134/0), 4 packets
%IPV6_ACL-6-ACCESSLOGDP: list V6-ACL3/500 denied icmpv6 2001:DB8::5 ->
FF02::1:FF00:1 (135/0), 30 packets
%IPV6_ACL-6-ACCESSLOGDP: list V6-ACL3/500 denied icmpv6 2001:DB8::5 ->
FE80::21F:9EFF:FE56:E4C9 (136/0), 1 packet
!
! Adding permit entries for the 4 basic ICMPv6 ND messages (before the explicit
deny)
V6-FW# show access-list V6-ACL3
IPv6 access list V6-ACL3
    permit ipv6 2001:DB8::/64 2001:DB8:0:1111::/64 sequence 10
    permit icmp any any nd-na (4 matches) sequence 491
    permit icmp any any nd-ns (4 matches) sequence 492
    permit icmp any any router-advertisement (16 matches) sequence 493
    permit icmp any any router-solicitation sequence 494
    deny ipv6 any any log (5 matches) sequence 500
```

Note IPv6 ACLs constructed as a sequence of ACEs that end on a **permit** statement have an **implicit deny** as their last entry (not visible with the **show access-list** command). There are two other **implicit permit** lines (also not visible), before the **implicit deny**, aimed at enabling the Neighbor Advertisement (NA) and Neighbor Solicitation (NS) messages:

```
permit icmp any any nd-na
```

```
permit icmp any any nd-ns
```

Despite these **permit** entries, NA and NS messages were blocked in Example 16-18 because the explicit **deny** takes precedence over the implicit entries. Further, it is convenient to underline that the other two classic ND messages (RS and RA) are not defined as **implicit permit** entries.

IPv6 Support in the Classic IOS Firewall

Chapter 9, “Classic IOS Firewall Overview,” presented a detailed analysis of the Classic IOS Firewall, which was originally referred to as Context Based Access Control (CBAC). Chapter 12, “Application Inspection,” explored the application-level inspection capabilities available for this IOS Firewall approach. This section briefly covers the CBAC inspection resources for IPv6 environments, while reiterating that the Zone-based Policy Firewall (ZFW) is the recommended IOS firewall implementation model going forward.

Example 16-19 relates to the scenario shown in Figure 16-12. A policy called CBAC-IPv6 is defined to promote generic L4 inspection for UDP and TCP. Stateless ICMP filtering rules (represented by ACLs OUTBOUND and INBOUND) enable echo requests and replies through the IOS Firewall (between networks 2001:db8::/64 and 2001:db8:0:1111::/64).

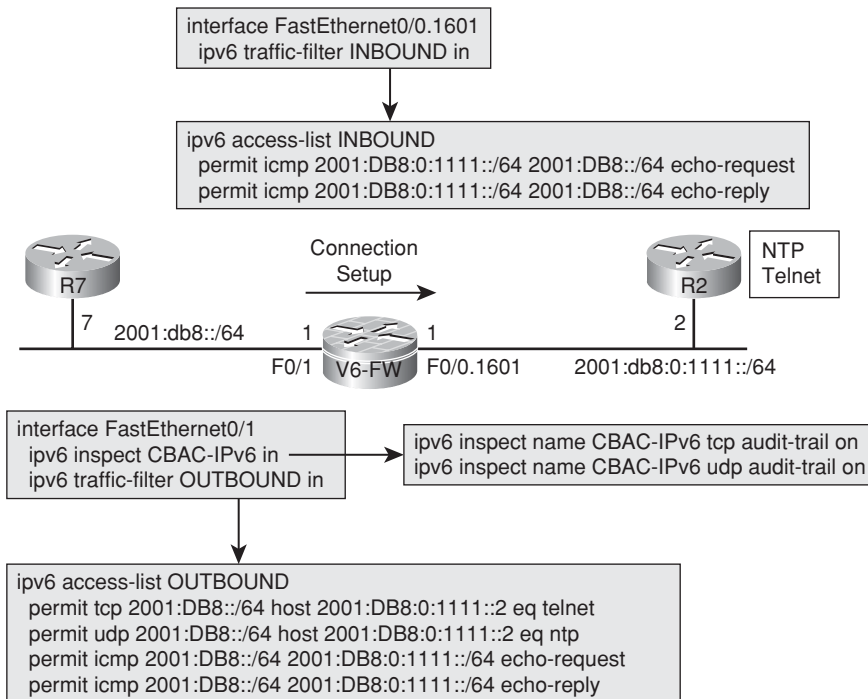


Figure 16-12 Reference Topology for CBAC Inspection of TCP and UDP

Example 16-19 CBAC Policy for Generic Inspection of UDP and TCP

```

! CBAC inspection options for IPv6
V6-FW(config)# ipv6 inspect name CBAC-IPv6?
ftp      File Transfer Protocol
  
```

```

icmp  ICMP Protocol
tcp    Transmission Control Protocol
udp    User Datagram Protocol
!
! UDP Session (NTP) creation and termination

FIREWALL ipv6_insp_init_sis - OBJ_CREATE: create sis 4B6B1060
FIREWALL ipv6_insp_init_sis: - Pak 497651C8 sis 4B6B1060
initiator_addr (2001:DB8::5:123) responder_addr
(2001:DB8:0:1111::2:123)
FIREWALL OBJ-CREATE: sid 48A425DC acl INBOUND Prot: udp
  Src 2001:DB8:0:1111::2 Port [123]
  Dst 2001:DB8::5 Port [123]
!
FIREWALL ipv6_insp_delete_sis - OBJ_DELETE: delete sis 4B6B1060
%IPV6_FW-6-SESS_AUDIT_TRAIL: udp session initiator (2001:DB8::5:123)
sent 48 bytes — responder (2001:DB8:0:1111::2:123) sent 48 bytes
FIREWALL OBJ-DELETE: sid 48A425DC on acl INBOUND
  Src 2001:DB8:0:1111::2 Port [123:123]
  Dst 2001:DB8::5 Port [123:123]
!
! R7 telnets to R2 through the Classic IOS Firewall
R7# telnet 2001:db8:0:1111::2
Trying 2001:DB8:0:1111::2 ... Open
User Access Verification
Password: ****
R2> show tcp brief
TCB          Local Address          Foreign Address          (state)
64F8DFC8    2001:DB8:0:1111::2.23    2001:DB8::7.43480       ESTAB
!
! TCP session creation and audit-trail (Telnet from R7 to R2)
FIREWALL ipv6_insp_init_sis - OBJ_CREATE: create sis 4B6B0EB0
FIREWALL ipv6_insp_init_sis: - Pak 497651C8 sis 4B6B0EB0
initiator_addr (2001:DB8::7:43480) responder_addr
(2001:DB8:0:1111::2:23)
FIREWALL OBJ-CREATE: sid 48A42574 acl INBOUND Prot: tcp
  Src 2001:DB8:0:1111::2 Port [23]
  Dst 2001:DB8::7 Port [43480]
FIREWALL OBJ_CREATE: create host entry 48A19594 addr 2001:DB8:0:1111::2 bucket 170
FIREWALL OBJ_DELETE: delete host entry 48A19594 addr 2001:DB8:0:1111::2
%IPV6_FW-6-SESS_AUDIT_TRAIL: tcp session initiator
(2001:DB8::7:43480) sent 66 bytes — responder
(2001:DB8:0:1111::2:23) sent 270 bytes
!

```

```
! Displaying information about the CBAC sessions
```

```
V6-FW# show ipv6 inspect session
```

```
Established Sessions
```

```
Session 4B6B0EB0 (2001:DB8::7:43480)=>(2001:DB8:0:1111::2:23) tcp SIS_OPEN
```

```
V6-FW# show ipv6 inspect session detail
```

```
Established Sessions
```

```
Session 4B6B0EB0 (2001:DB8::7:43480)=>(2001:DB8:0:1111::2:23) tcp SIS_OPEN
```

```
Created 00:00:33, Last heard 00:00:17
```

```
Bytes sent (initiator:responder) [66:264]
```

```
In SID 2001:DB8:0:1111::2[23:23]=>2001:DB8::7[43480:43480] on ACL_INBOUND
```

```
(36 matches)
```

Examples 16-20 and 16-21 employ the network topology and policy schematics represented in Figure 16-13. Example 16-20 deals exclusively with ICMP, as described here:

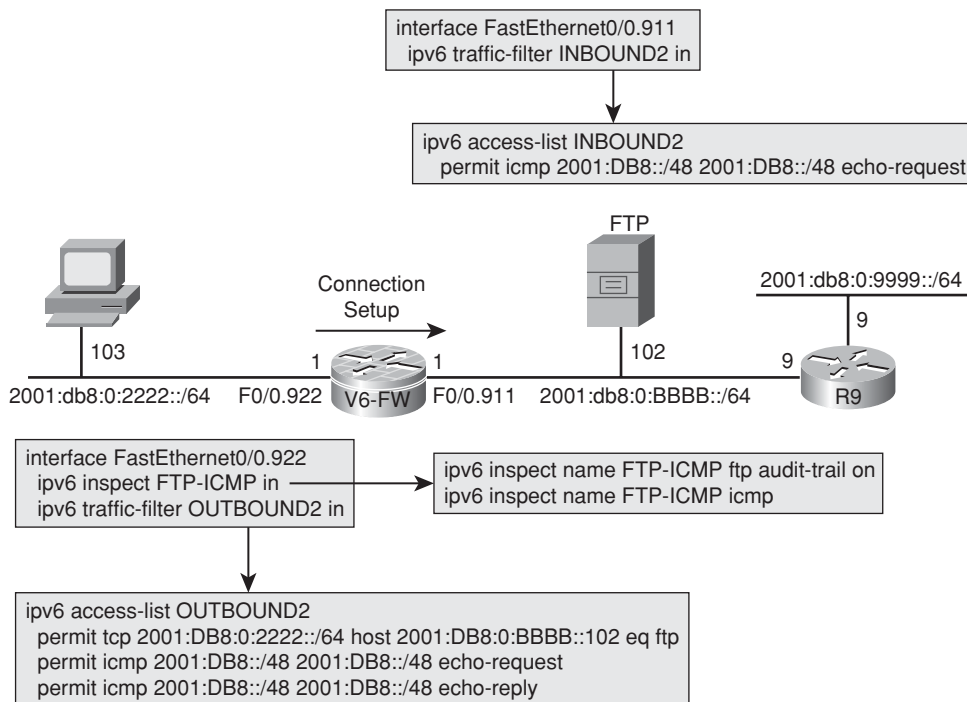


Figure 16-13 Network Topology for CBAC Inspection of ICMP and FTP

- A ping session in the outbound direction is inspected by CBAC. The firewall sees each echo-request and the correspondent echo-reply packet.

- A second ping session (now in the inbound direction) is treated in a stateless manner. For every echo-request registered in the ACL called INBOUND2, an echo-reply is seen on ACL OUTBOUND2.
- The example also shows an outbound **traceroute** session from the Windows host on 2001:db8:0:2222::3 toward destination 2001:db8:0:9999::9. The stateful inspection of ICMP dynamically creates permissions for return traffic on ACL INBOUND2. The message types automatically allowed back (in response to an echo request) follow:
 - **Type 129:** Echo Reply
 - **Type 1:** Destination Unreachable
 - **Type 2:** Packet too big
 - **Type 3:** Time Exceeded

Example 16-20 CBAC in Action for ICMP Inspection

```
! Outbound ping from 2001:db8:0:2222::103 to destination 2001:db8:0:bbbb::102
C:\Documents and Settings\Administrator.AD1>ping 2001:db8:0:bbbb::102 -l 900 -n 2
Pinging 2001:db8:0:bbbb::102 from 2001:db8:0:2222::103 with 900 bytes of data:
Ping statistics for 2001:db8:0:bbbb::102:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),

! What the firewall sees (for each echo-request packet sent)
FIREWALL* sis 4B6B0EB0 L4 inspect result: PASS packet 4949EC64
(2001:DB8:0:2222::103:0) (2001:DB8:0:BBBB::102:0) bytes 908 icmp
FIREWALL* sis 4B6B0EB0 L4 inspect result: PASS packet 4949EC64
(2001:DB8:0:BBBB::102:0) (2001:DB8:0:2222::103:0) bytes 908 icmp
!
V6-FW# show access-list OUTBOUND2 | include icmp
    permit icmp 2001:DB8::/48 2001:DB8::/48 echo-request (2 matches) sequence 20
    permit icmp 2001:DB8::/48 2001:DB8::/48 echo-reply sequence 30
!
! Inbound ping session (50 packets) - no inspection

R9# ping 2001:db8:0:2222::103 source 2001:db8:0:9999::9 repeat 50
Sending 50, 100-byte ICMP Echos to 2001:DB8:0:2222::103, timeout is 2 seconds:
Packet sent with a source address of 2001:DB8:0:9999::9
Success rate is 100 percent (50/50), round-trip min/avg/max = 0/3/80 ms
!
V6-FW# show access-list INBOUND2 | include icmp
    permit icmp 2001:DB8::/48 2001:DB8::/48 echo-request (50 matches) sequence 10
V6-FW# show access-list OUTBOUND2 | include icmp
    permit icmp 2001:DB8::/48 2001:DB8::/48 echo-request (2 matches) sequence 20
    permit icmp 2001:DB8::/48 2001:DB8::/48 echo-reply (50 matches) sequence 30
```

```

! Traceroute session from Windows host 2001:db8:0:2222::103 to
2001:db8:0:9999::9
C:\Documents and Settings\Administrator.AD1>tracert 2001:db8:0:9999::9
Tracing route to 2001:db8:0:9999::9 over a maximum of 30 hops
    1      2 ms      <1 ms      <1 ms    2001:db8:0:2222::1
    2      1 ms      1 ms      1 ms    2001:db8:0:9999::9
Trace complete.
!
V6-FW# show ipv6 inspect session detail
Established Sessions
Session 4B6B0EB0 (2001:DB8:0:2222::103:0)=>(2001:DB8:0:9999::9:0) icmp SIS_OPEN
Created 00:00:03, Last heard 00:00:03
Destinations: 1
    Dest addr [2001:DB8:0:9999::9]
Bytes sent (initiator:responder) [192:192]
In SID 2001:DB8:0:9999::9[0:0]=>2001:DB8:0:2222::103[129:129] on ACL INBOUND2
(3 matches)
In SID 2001:DB8:0:9999::9[0:0]=>2001:DB8:0:2222::103[2:2] on ACL INBOUND2
In SID 2001:DB8:0:9999::9[0:0]=>2001:DB8:0:2222::103[1:1] on ACL INBOUND2
In SID 2001:DB8:0:9999::9[0:0]=>2001:DB8:0:2222::103[3:3] on ACL INBOUND2

```

Example 16-21 shows the Classic IOS Firewall application awareness for a typical FTP session from client 2001:db8:0:2222::103 to server 2001:db8:0:bbbb::102. In the example, you can see that the client issues the EPRT command, which is the IPv6 version of the PORT command for active FTP. (Likewise, the EPSV command plays the role of the PASV command in IPv6 passive FTP sessions). RFC 2428 defines the FTP extensions for IPv6 support.

Example 16-21 CBAC in Action for FTP Inspection

```

! Initial connection and user authentication (command session is created)
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 1
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Client: USER user1~~
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 1
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Server: 331 Password required for user1~~
FIREWALL* FUNC: ipv6_insp_watch_ftp_login_rsp — State 1
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 1
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Client: PASS xxxxxxxx
FIREWALL* FUNC: ipv6_insp_watch_ftp_pass_cmd — sis 4B6B0EB0 PASS
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 2
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0

```

```

FTP-Server: 230 Login OK~~
FIREWALL* FUNC: ipv6_insp_watch_ftp_login_rsp — State 2
FIREWALL* FUNC: ipv6_insp_watch_ftp_login_rsp — sis 4B6B0EB0, User authenticated
!
! user1 lists the files (first FTP-data session is created)
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Client: EPRT |2|2001:db8:0:2222::103|2857|~~
FIREWALL* FUNC: ipv6_insp_watch_ftp_eprt_cmd — delimiter = |
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Server: 200 PORT command successful~~
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Client: NLST~~
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Server: 150 Opening ASCII mode data connection for NLIST (39 bytes).~~
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Server: 226 Transfer complete (0.235 KB/s).~~
%IPV6_FW-6-SESS_AUDIT_TRAIL: ftp-data session initiator (2001:DB8:0:BBBB::102:20)
sent 39 bytes — responder (2001:DB8:0:2222::103:2857) sent 0 bytes
!
! user1 changes the transfer mode to binary ('bin' command)
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Client: TYPE I~~
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Server: 200 Type set to I.~~
!
! user1 downloads the file 'Reg1.exe' (second FTP-data session is created)
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Client: EPRT |2|2001:db8:0:2222::103|2858|~~
FIREWALL* FUNC: ipv6_insp_watch_ftp_eprt_cmd — delimiter = |
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Server: 200 PORT command successful~~
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Client: RETR Reg1.exe~~
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3

```

```

FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Server: 150 Opening BINARY mode data connection for Reg1.exe (477 bytes).~~
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Server: 226 Transfer complete (0.926 KB/s).~~
%IPV6_FW-6-SESS_AUDIT_TRAIL: ftp-data session initiator (2001:DB8:0:BBBB::102:20)
sent 477 bytes — responder (2001:DB8:0:2222::103:2858) sent 0 bytes
!
! user1 finishes the session and the control connection is terminated
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Client: QUIT~~
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — State 3
FIREWALL* FUNC: ipv6_insp_ftp_process_control_data — FTP sis 4B6B0EB0
FTP-Server: 221 Good-Bye~~
%IPV6_FW-6-SESS_AUDIT_TRAIL: ftp session initiator (2001:DB8:0:2222::103:2853)
sent 131 bytes — responder (2001:DB8:0:BBBB::102:21) sent 378 bytes

```

In summary, the Classic IOS Firewall for IPv6 supports the following features:

- Generic stateful inspection of TCP, UDP, and ICMP being carried over IPv6.
- Stateful inspection of FTP.
- Port-to-application mapping (PAM) with the aid of the **ipv6 port-map** command. If you need to review the PAM operation, refer to Chapter 12.
- Inspection of fragmented IPv6 packets: This topic is illustrated in the “IPv6 and Fragmentation” section.
- Inspection of tunneled packets: CBAC for IPv6 supports inspection of tunneled packets terminating at the same router in which CBAC is running. A brief discussion of IPv6 tunneling techniques, and guidelines for firewall placement in this type of environment, are presented in Chapter 17, “Firewall Interactions.”

The Classic IOS Firewall uses the traffic class, flow label, payload length, next header, hop limit and source and destination addresses header fields for inspection purposes.

IPv6 Support in the Zone Policy Firewall

A natural follow-on to the analysis of the Classic IOS Firewall is the study of the Zone-based Policy Firewall (ZFW), the Cisco recommended approach for IOS-based Firewall implementations. This section brings a set of examples covering ZFW policy construction for IPv6. ZFW IPv6 policies employ the same logic that characterizes their IPv4 counterparts.

This section was built using IOS version 15.1(2)T, the initial release of ZFW for IPv6. This is an incipient implementation of ZFW, which mainly focuses on replacing the existent CBAC-based deployments. Nonetheless, it is already useful for illustrating that Cisco is committed with the evolution of this important IOS software module.

Example 16-22 shows the **type inspect** parameter-maps used within this section. The keyword **global** was introduced in version 15.1(1)T, therefore allowing the configuration of the **log** options within the **parameter-map**.

Example 16-22 *Zone Policy Firewall and parameter-maps*

```
! Default settings for the 'global' parameter-map
V6-FW# show parameter-map type inspect global
  alert on
  sessions maximum 2147483647
  waas disabled
  l2-transparent dhcp-passthrough disabled
  log dropped-packets disabled
  log summary disabled
  max-incomplete low 2147483647
  max-incomplete high 2147483647
  one-minute low 2147483647
  one-minute high 2147483647
!
! Changing a setting of the 'global' parameter-map
parameter-map type inspect global
  log dropped-packets enable
!
! Custom parameter-map to be used with specific Zone policies
parameter-map type inspect TRACKING
  audit-trail on
```

Figure 16-14 portrays an IPv6 environment for which the security zones INSIDE and DMZ have been defined. Packets flowing from the INSIDE to the DMZ zone are subject to the zone-pair policy OUTBOUND1, which relies on the **service-policy** named POLICY1.

Example 16-23 relates to Figure 16-14 and registers the behavior of an ICMP session through the IPv6-enabled ZFW. In the example, R5 (INSIDE host) pings host R2, which is located on the DMZ. The perspective of each element participating in the scenario is also captured.

Example 16-23 *ICMP Connection Through the Zone Policy Firewall*

```
! R5 (2001:db8::5) sends a PING to R2
R5# ping ipv6 2001:db8:0:1111::2 repeat 1
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/4/4 ms
```

```

ICMPv6: Sent echo request, Src=2001:DB8::5, Dst=2001:DB8:0:1111::2
ICMPv6: Received echo reply, Src=2001:DB8:0:1111::2, Dst=2001:DB8::5
!
! R2's perspective

ICMPv6: Received echo request, Src=2001:DB8::5, Dst=2001:DB8:0:1111::2
ICMPv6: Sent echo reply, Src=2001:DB8:0:1111::2, Dst=2001:DB8::5
!
! What is seen on the Zone Policy Firewall

%IPV6_FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:GENERIC-
V6):Start icmpv6 session: initiator ([2001:DB8::5]:0) — responder
([2001:DB8:0:1111::2]:0)
FIREWALL* sis 4A65BB00: ICMPv6 Echo pkt [2001:DB8::5] => [2001:DB8:0:1111::2]
FIREWALL* sis 4A65BB00: ICMPv6 Echo Reply pkt [2001:DB8:0:1111::2] =>
[2001:DB8::5]

```

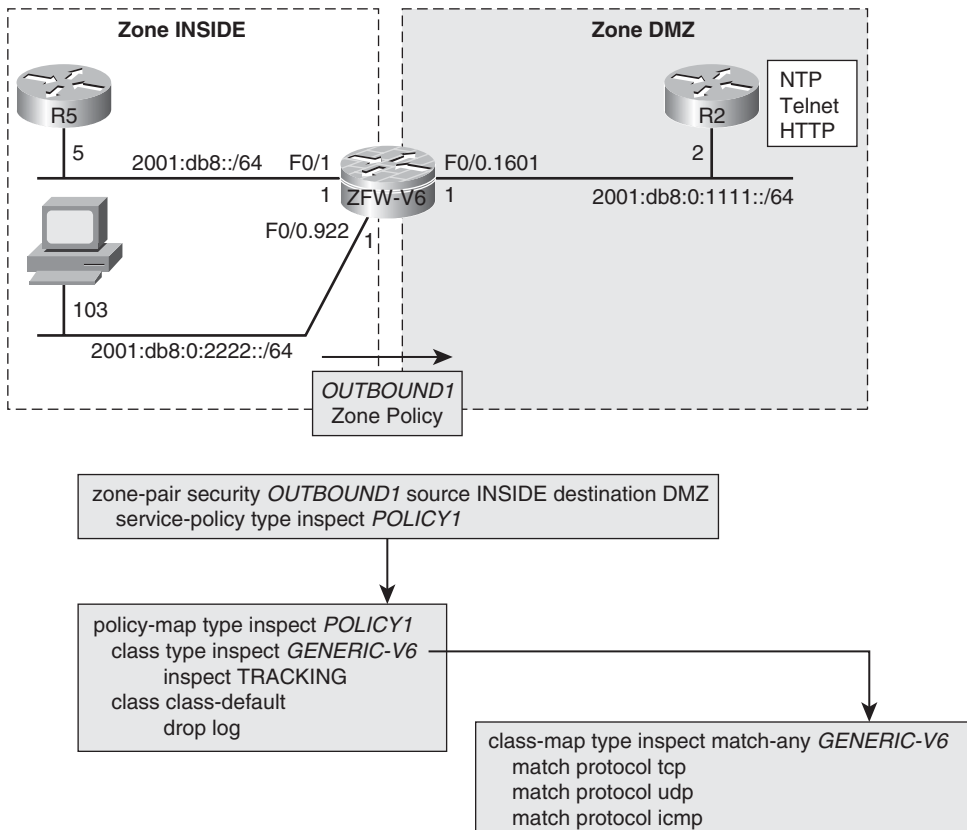


Figure 16-14 Zone Policy Firewall - Simple IPv6 Policy

Example 16-24 uses NTP to illustrate the generic inspection of UDP in the ZFW. In the example, the NTP client on R5 connects to the NTP server on R2.

Example 16-24 *UDP Connection Through the Zone Firewall*

```
! R5 (NTP Client) sends a packet to R2 (NTP Server) and session is created
FIREWALL* sis 49FA6440: Session Created
FIREWALL* sis 49FA6440: IPv6 address extention Created
FIREWALL* sis 49FA6440: Pak 497651C8 init_addr ([2001:DB8::5]:123)
resp_addr ([2001:DB8:0:1111::2]:123)
FIREWALL* sis 49FA6440: FO cls 0x489C3100 clsgrp 0x20000000, target
0xA0000000, FO 0x4A91F6C0, alert = 1, audit_trail = 1, L7 = Unknown-
17, PAMID = 0

%IPv6_FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:GENERIC-V6):Start udp
session: initiator ([2001:DB8::5]:123) — responder ([2001:DB8:0:1111::2]:123)

! Displaying established sessions
ZFW-V6# show policy-map type inspect zone-pair OUTBOUND1 sessions | include Session
Number of Established Sessions = 1
Established Sessions
Session 49FA6440 [2001:DB8::5]:123=>[2001:DB8:0:1111::2]:123 udp SIS_OPEN
!
! Connection termination

FIREWALL sis 49FA6440: Deleting sis half_open 0
FIREWALL sis 49FA6440: Deleting sis
FIREWALL sis 49FA6440: session on temporary delete list
%IPv6_FW-6-SESS_AUDIT_TRAIL: (target:class)-(OUTBOUND1:GENERIC-
V6):Stop udp session initiator ([2001:DB8::5]:123) sent 48 bytes —
responder ([2001:DB8:0:1111::2]:123) sent 48 bytes
```

Example 16-25 relies on Telnet to show how ZFW deals with generic TCP inspection. It is instructive to observe that the firewall is aware of parameters such as TCP flags (besides source and destination ports).

Example 16-26 shows another situation in which TCP inspection takes place through the ZFW, but now using HTTP. In the example, the web browser on 2001:db8:0:2222::103 opens a web page on R2. URLs that use IPv6 addresses (instead of hostnames) must be written between brackets to avoid confusion with the port number over which HTTP is running.

Example 16-25 *Telnet Connection Through the Zone Firewall*

```

! Telnet from R5 (2001:db8::5) to R2
R5# telnet 2001:db8:0:1111::2
Trying 2001:DB8:0:1111::2 ... Open

Reserved port 39579 in Transport Port Agent for TCP IP type 20
TCP: sending SYN, seq 2111465098, ack 0
TCP0: Connection to 2001:DB8:0:1111::2:23, advertising MSS 516
TCP0: state was CLOSED -> SYNSENT [39579 -> 2001:DB8:0:1111::2(23)]
TCP0: state was SYNSENT -> ESTAB [39579 -> 2001:DB8:0:1111::2(23)]
TCP: tcb 476F3F0C connection to 2001:DB8:0:1111::2:23, peer MSS 516, MSS is 516
TCB476F3F0C connected to 2001:DB8:0:1111::2.23
User Access Verification
Password:*****
R2>
!
! R2 receives TCP SYN packet and responds with SYN ACK packet
Reserved port 0 in Transport Port Agent for TCP IP type 0
TCP0: state was LISTEN -> SYNRCVD [23 -> 2001:DB8::5(39579)]
TCP: tcb 64F9F6D0 connection to 2001:DB8::5:39579, peer MSS 516, MSS is 516
TCP: sending SYN, seq 3417206967, ack 2111465099
TCP0: Connection to 2001:DB8::5:39579, advertising MSS 516
TCP0: state was SYNRCVD -> ESTAB [23 -> 2001:DB8::5(39579)]
R2# show tcp brief
TCB          Local Address          Foreign Address          (state)
64F9F6D0     2001:DB8:0:1111::2.23   2001:DB8::5.39579       ESTAB
!
! ZFW's perspective

%IPV6_FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:GENERIC-V6):Start tcp
session: initiator ([2001:DB8::5]:39579) — responder ([2001:DB8:0:1111::2]:23)

FIREWALL* sis 49FAC640: pak 497651C8 SIS_CLOSED/LISTEN TCP SYN SEQ 2111465098
LEN 0 ([2001:DB8::5]:39579) => ([2001:DB8:0:1111::2]:23)
FIREWALL* sis 49FAC640: Max window size changed from 0 to 4128
FIREWALL* sis 49FAC640: pak 4949EC64 SIS_OPENING/SYNSENT TCP SYN ACK 2111465099
SEQ 3417206967 LEN 0 ([2001:DB8::5]:39579) <= ([2001:DB8:0:1111::2]:23)
FIREWALL* sis 49FAC640: Max window size changed from 0 to 4128
FIREWALL* sis 49FAC640: pak 497651C8 SIS_OPENING/SYNRCVD TCP ACK
3417206968 SEQ 2111465099 LEN 0([2001:DB8::5]:39579) =>
([2001:DB8:0:1111::2]:23)
!
ZFW-V6# show policy-map type inspect zone-pair OUTBOUND1 sessions | include Session
Number of Established Sessions = 1

```

```

Established Sessions
    Session 49FAC640 [2001:DB8::5]:39579=>[2001:DB8:0:1111::2]:23
tcp SIS_OPEN/TCP_ESTAB

```

Example 16-26 HTTP Connection Through the Zone Firewall

```

! Browser on 2001:db8:0:2222::103 opens a web page on R2 ([2001:DB8:0:1111::2])
FIREWALL* sis 4A659B80: Session Created
FIREWALL* sis 4A659B80: IPv6 address extension Created
FIREWALL* sis 4A659B80: Pak 4949EC64 init_addr ([2001:DB8:0:2222::103]:2287)
resp_addr ([2001:DB8:0:1111::2]:80)
FIREWALL* sis 4A659B80: FO cls 0x489C3100 clsgrp 0x20000000, target
0xA0000000, FO 0x4A91F6C0, alert = 1, audit_trail = 1, L7 = Unknown-
17, PAMID = 0
FIREWALL* sis 4A659B80: create host entry 4A9F6D40 addr
[2001:DB8:0:1111::2] bucket 150 (vrf 0:0) fwfo 0x489BB080
%IPv6_FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND1:GENERIC-
V6):Start tcp
session: initiator ([2001:DB8:0:2222::103]:2287) — responder
([2001:DB8:0:1111::2]:80)

```

Figure 16-15 assembles the topology and policy schematics for the analysis of a ZFW policy that includes an IPv6 ACL. The ACL under the policy-map POLICY2 is defined in Example 16-27. The example also demonstrates some packet drops that result from non-conformance with the ACL rules:

- 2001:db8:0:6666::5 is not an acceptable source for an echo request.
- HTTPS to R2 is not allowed from any source.

Example 16-27 Zone Policy Firewall and IPv6 ACL

```

! Sample IPv6 ACL (defines one of the match conditions under class JOINT-V6)
ipv6 access-list V6-ACL1
 permit icmp 2001:DB8::/64 any echo-request
 permit tcp any any eq www
 permit tcp any any eq telnet
 permit udp any host 2001:DB8:0:1111::2 eq ntp
!
! R5 sends a ping to R2 from an address out of the 2001:db8::/64 network

R5# ping ipv6 2001:db8:0:1111::2 source 2001:db8:0:6666::5 repeat 1
Sending 1, 100-byte ICMP Echos to 2001:DB8:0:1111::2, timeout is 2 seconds:
Packet sent with a source address of 2001:DB8:0:6666::5
ICMPv6: Sent echo request, Src=2001:DB8:0:6666::5, Dst=2001:DB8:0:1111::2.

```

```

Success rate is 0 percent (0/1)
!
! Echo-request is dropped, for it does not match the ACL condition of class
JOINT-V6
%FW-6-DROP_PKT: Dropping icmpv6 session [2001:DB8:0:6666::5]:0
[2001:DB8:0:1111::2]:0
on zone-pair OUTBOUND2 class class-default due to DROP action found in policy-map
with ip ident 0
!
! HTTPS to R2 is blocked (ACL allows HTTP instead of HTTPS)
%FW-6-DROP_PKT: Dropping tcp session [2001:DB8:0:2222::103]:2456

```

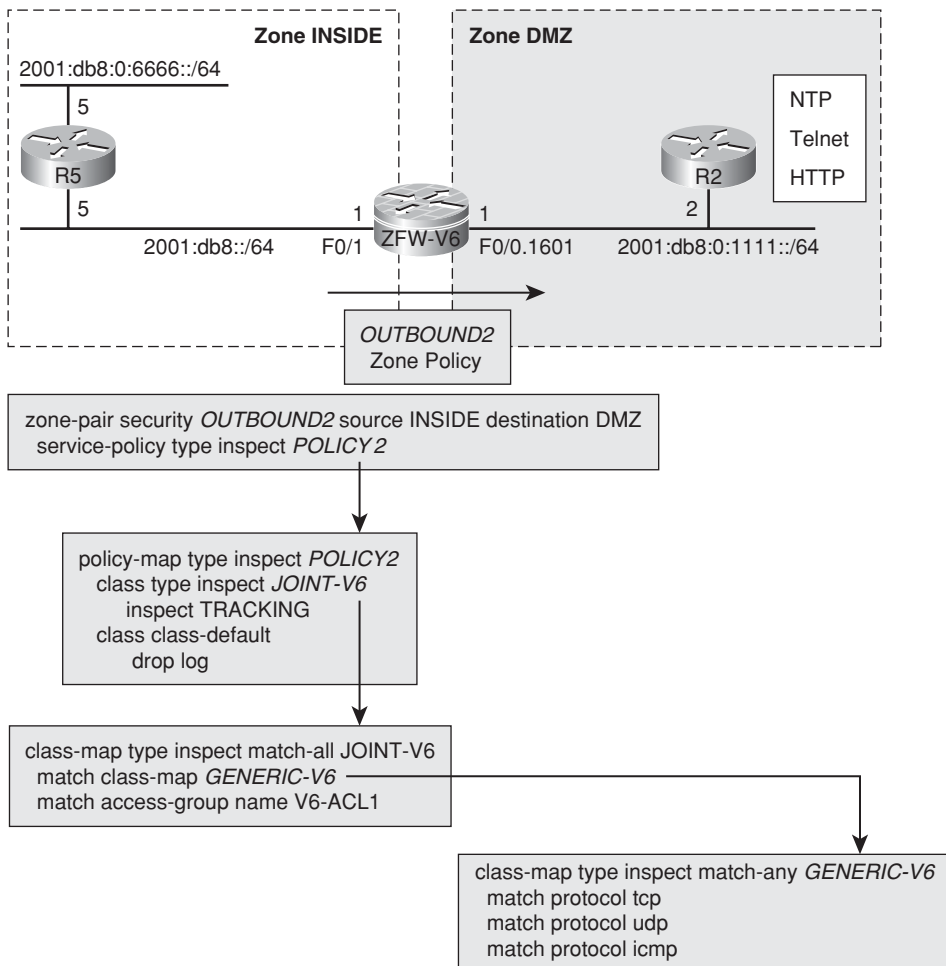


Figure 16-15 Zone Policy Firewall and IPv6 ACL

```
[2001:DB8:0:1111::2]:443 on zone-pair OUTBOUND2 class class-default
due to DROP action found in policy-map with ip ident 0
!
! Displaying the IPv6 ACL that is part of the ZFW policy definition

ZFW-V6# show access-list V6-ACL1
IPv6 access list V6-ACL1
    permit icmp 2001:DB8::/64 any echo-request (1 match) sequence 10
    permit tcp any any eq www (1 match) sequence 20
    permit tcp any any eq telnet sequence 30
    permit udp any host 2001:DB8:0:1111::2 eq ntp (16 matches) sequence 40
!
! Class class-default in action

ZFW-V6# show policy-map type inspect zone-pair OUTBOUND2 | begin class-default
Class-map: class-default (match-any)
    Match: any
    Drop
        4 packets, 132 bytes
```

Figure 16-16 depicts the topology used for the study of FTPv6 sessions through the ZFW. Given that FTP uses dynamically negotiated secondary channels for data transport, generic TCP inspection is not enough. The client commands are the same as those documented in Example 16-21. The difference is that, in this new case, there is no **debug** turned on. The visibility of FTP data sessions is provided by the **audit-trail** feature.

Example 16-28 *FTP over IPv6 Through the Zone Policy Firewall*

```
C:\Documents and Settings\Administrator.AD1>ftp 2001:db8:0:bbbb::102
Connected to 2001:db8:0:bbbb::102.
220 Xlight FTP Server 3.5 ready...
User (2001:db8:0:bbbb::102:(none)): user1
331 Password required for user1
Password:*****
230 Login OK
ftp> bin
200 Type set to I.
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for NLIST (39 bytes).
exe1.doc
Reg1.exe
226 Transfer complete (0.235 KB/s).
ftp: 39 bytes received in 0.00Seconds 39000.00Kbytes/sec.
ftp> get Reg1.exe
```

```

200 PORT command successful
150 Opening BINARY mode data connection for Reg1.exe (477 bytes).
226 Transfer complete (1.559 KB/s).
ftp: 477 bytes received in 0.06Seconds 7.69Kbytes/sec.
ftp> quit
221 Good-Bye
C:\Documents and Settings\Administrator.AD1>

! FTP session from the ZFW's standpoint

%IPV6_FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND3:V6-
FTP):Start ftp session: initiator ([2001:DB8:0:2222::103]:2510) —
responder ([2001:DB8:0:BBBB::102]:21)
%IPV6_FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND3:V6-
FTP):Start ftp-data session: initiator ([2001:DB8:0:BBBB::102]:20) —
responder ([2001:DB8:0:2222::103]:2512)
%IPV6_FW-6-SESS_AUDIT_TRAIL: (target:class)-(OUTBOUND3:V6-FTP):Stop
ftp-data sessioninitiator ([2001:DB8:0:BBBB::102]:20) sent 39 bytes
— responder ([2001:DB8:0:2222::103]:2512) sent 0 bytes
%IPV6_FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(OUTBOUND3:V6-
FTP):Start ftp-data session: initiator ([2001:DB8:0:BBBB::102]:20) —

```

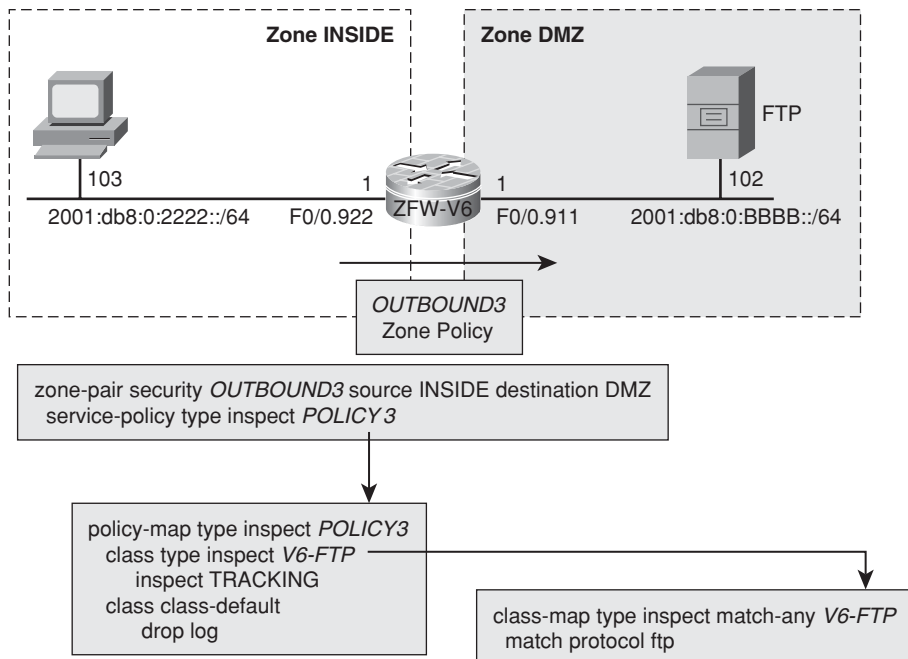


Figure 16-16 Zone Policy Firewall and FTP over IPv6


```

responder ([2001:DB8:0:2222::103]:2513)
%IPV6_FW-6-SESS_AUDIT_TRAIL: (target:class)-(OUTBOUND3:V6-FTP):Stop
ftp-data session initiator ([2001:DB8:0:BBBB::102]:20) sent 477 bytes
— responder ([2001:DB8:0:2222::103]:2513) sent 0 bytes
%IPV6_FW-6-SESS_AUDIT_TRAIL: (target:class)-(OUTBOUND3:V6-FTP):Stop
ftp session
initiator ([2001:DB8:0:2222::103]:2510) sent 147 bytes — responder
([2001:DB8:0:BBBB::102]:21) sent 418 bytes

```

Note The `show policy-map type inspect zone-pair` commands were used throughout this chapter to facilitate comparison with the IPv4 Examples presented within previous chapters. With the advent of IOS releases 15.X, a set of shorter commands became available for verifying ZFW operation:

V6-FW# show policy-firewall?

```

config      Show policy-firewall config
mib         Policy Firewall MIB specific show commands
session     Show policy-firewall session details
stats       Show policy-firewall stats
summary-log show ZBFW summary logs

```

The reader is naturally encouraged to play with these new options.

Handling ASA IPv6 ACLs and Object-Groups

This section summarizes ACLs and **object-groups** that can be used in IPv6 environments protected by ASA firewalls. The reference topology for all the examples in this section is depicted in Figure 16-17.

Example 16-29 contains sample network and service **object-groups** for use within IPv6 scenarios. Overall, these IPv6-based **object-groups** are close to their IPv4 equivalents. One important exception to be aware of is that ICMPv6 messages need to be defined under enhanced object-groups. ICMP-specific service **object-groups** deal only with the original ICMP protocol. Some of these **object-groups** will be later used in Example 16-31 with the goal of simplifying ACL configuration.

Example 16-29 Sample ASA Object-Groups

```

! Sample Network Object-Groups
object-group network DOC-NETS
  description * IPv6 Documentation Prefix *
  network-object 2001:db8::/32

```

```

!
object-group network INSIDE-NETS
  network-object 2001:db8::/64
  network-object 2001:db8:0:2222::/64
!
object-group network MGMT-HOSTS1
  network-object host 2001:db8:0:2222::103
!
! Sample TCP Service-Group
object-group service TCP1 tcp
  description * TCP-based Management Protocols *
  port-object eq www
  port-object eq https
  port-object range ssh telnet
!
! Sample Enhanced Service-group for ICMPv6

object-group service V6-ICMP
  description * ICMP messages needed for traceroute *
  service-object icmp6 time-exceeded
  service-object icmp6 unreachable

```

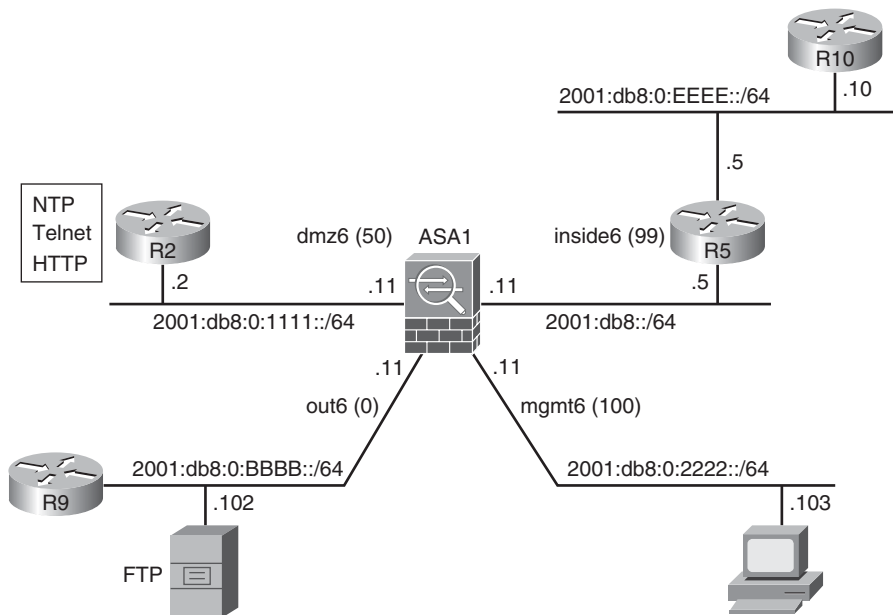


Figure 16-17 Reference Topology for ASA-Related Examples

Example 16-30 assembles basic information about ASA IPv6 ACLs. One advantage of ASA ACLs when compared to IOS is that the former support **object-groups**. The source to destination construction logic is identical to that employed by IOS.

Example 16-30 Basic Information About IPv6 ACLs

```
ASA1(config)# ipv6 access-list V6-ACL1 permit?
configure mode commands/options:
  <0-255>      Enter protocol number (0 - 255)
  ah
  eigrp
  esp
  gre
  icmp
  icmp6
  igmp
  igrp
  ip
  ipinip
  ipsec
  nos
  object-group Specify a service or protocol object-group after this keyword
  ospf
  pcp
  pim
  pptp
  snp
  tcp
  udp
  !
ASA1(config)# ipv6 access-list OUT6 permit ip any any?
configure mode commands/options:
  inactive      Keyword for disabling an ACL element
  log           Keyword for enabling log option on this ACL element
  time-range    Keyword for attaching time-range option to this ACL element
  <cr>
```

Note At the time of this writing, ASA did not support filtering based on a specific extension header. However, it is important to register that IPv6 packets carrying the routing header are always blocked, without the need of any further configuration:

```
ASA1# show asp drop | include header
Unsupported IPV6 header (unsupport-ipv6-hdr)
```

10

Example 16-31 relates to the scenario represented in Figure 16-17 and demonstrates how **object-groups** are used to simplify ACL definition. The example also teaches how to tie an ACL to a particular ASA interface and suggests a way of displaying ACEs with non-null hit counts.

One point that deserves special attention in Example 16-31 is how the ICMPv6 enhanced object-group is referenced by the ACL. Enhanced object-groups do not use protocol keywords after the **permit | deny** statement. Please compare to the TCP service-group and, if you need extra details about this topic, revisit Chapter 7, “Through ASA Without NAT.”

Example 16-31 *Sample IPv6 ACL That Employs Object-Groups*

```

ipv6 access-list OUT6 permit object-group V6-ICMP object-group DOC-NETS object-
group INSIDE-NETS
ipv6 access-list OUT6 permit tcp 2001:db8:0:bbbb::/64 object-group INSIDE-NETS eq
telnet
ipv6 access-list OUT6 permit tcp 2001:db8:0:bbbb::/64 2001:db8:0:1111::/64 object-
group TCP1
!
! Binding the IPv6 ACL to interface out6
access-group OUT6 in interface out6
!
! Exhibiting the IPv6 ACL and the ACEs with non-null hit counts

ASA1# show access-list OUT6 | exclude =0
ipv6 access-list OUT6; 9 elements; name hash: 0xca9e993e
ipv6 access-list OUT6 line 1 permit object-group V6-ICMP object-group DOC-NETS
object-group INSIDE-NETS 0x28c7d082
ipv6 access-list OUT6 line 2 permit tcp 2001:db8:0:bbbb::/64 object-group INSIDE-
NETS eq telnet 0x52781f09
ipv6 access-list OUT6 line 3 permit tcp 2001:db8:0:bbbb::/64 2001:db8:0:1111::/64
object-group TCP1 0x87aaab4a
    ipv6 access-list OUT6 line 3 permit tcp 2001:db8:0:bbbb::/64
2001:db8:0:1111::/64
eq www (hitcnt=4) 0xa6c51e3e
    ipv6 access-list OUT6 line 3 permit tcp 2001:db8:0:bbbb::/64
2001:db8:0:1111::/64
eq https (hitcnt=4) 0x542b2c49

```

Example 16-32 shows how to start a **packet-tracer** simulation using IPv6 data. In this particular case, an ICMP echo request from 2001:db8:0:bbbb::9 2001:db8:0:2222::103 was denied by the ACL called OUT6. As shown on previous chapters, **packet-tracer** is a useful tool to understand how ASA software modules deal with specific traffic flows. At this point, you are highly encouraged to start applying this powerful resource to better explore other IPv6 scenarios.

Example 16-32 *Sample Packet-tracer Simulation Revealing an ACL-Based Drop*

```
ASA1# packet-tracer input out6 icmp 2001:db8:0:bbbb::9 128 0 1
2001:db8:0:2222::103
```

Phase: 1

Type: FLOW-LOOKUP

Subtype:

```
%ASA-4-106023: Deny icmp src out6:2001:db8:0:bbbb::9 dst
mgmt6:2001:db8:0:2222::103
```

```
(type 128, code 0) by access-group "OUT6" [0x0, 0x0]
```

Result: ALLOW

Config:

Additional Information:

Found no matching flow, creating a new flow

Phase: 2

Type: ROUTE-LOOKUP

Subtype: input

Result: ALLOW

Config:

Additional Information:

```
in 2001:db8:0:2222:: ffff:ffff:ffff:ffff:: mgmt6
```

Phase: 3

Type: ACCESS-LIST

Subtype:

Result: DROP

Config:

Implicit Rule

Additional Information:

Result:

input-interface: out6

input-status: up

input-line-status: up

output-interface: mgmt6

output-status: up

output-line-status: up

Action: drop

Drop-reason: (acl-drop) Flow is denied by configured rule

Stateful Inspection of IPv6 in ASA

Figure 16-18 shows the base Adaptive Security Device Manager (ASDM) screen from which you can start defining IPv6 access rules. The screenshot also underlines the following facts:

- In ASDM, you can display any combination of IPv6 and IPv4 access rules. In this particular case, the choice was to display only IPv6-related rules.

- The implicit rules based on the concept of security-levels are equally valid for IPv6.

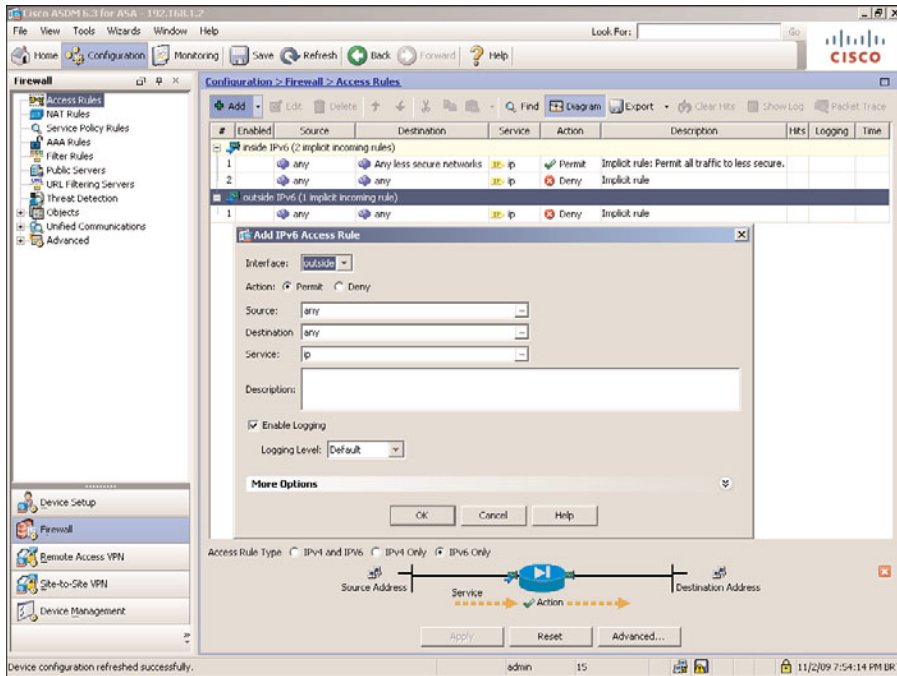


Figure 16-18 ASDM and IPv6

Example 16-33 was conceived to illustrate basic connections through ASA:

- An NTP session between R5 (client) and R2 (server) registers the behavior of generic UDP inspection.
- An IOS traceroute departing from R5 and targeted at R9 not only shows the pertinent UDP and ICMP connections but also characterizes that the ASA does not display itself as a hop for IPv6. Chapter 7 demonstrated how to change this setting for IPv4. (But there is no correspondent method in IPv6 yet.)
- A telnet from R5 to R9 provides insight about generic TCP inspection. TCP Sequence Number randomization is supported for IPv6. Disabling this feature for a given TCP application is possible (either globally or on a per-interface basis). However, given that **class-maps** that include IPv6 ACLs are not yet available for traffic classification, you cannot disable this feature for specific source or destination addresses.

Example 16-33 *Basic Connections Through ASA*

```

! NTP connection from 2001:db8::5 (R5) to 2001:db8:0:1111::2 (R2)
%ASA-6-302015: Built outbound UDP connection 23145 for
dmz6:2001:db8:0:1111::2/123 (2001:db8:0:1111::2/123) to
inside6:2001:db8::5/123 (2001:db8::5/123)
ASA1# show conn protocol udp
4 in use, 35 most used
UDP dmz6 2001:db8:0:1111::2:123 inside6 2001:db8::5:123,idle 0:00:38,bytes 192,
flags -

! IOS traceroute from 2001:db8::5 (R5) to 2001:db8:0:9999::9 (R9)

R5# traceroute 2001:db8:0:9999::9
Tracing the route to 2001:DB8:0:9999::9
 1 2001:DB8:0:9999::9 4 msec 0 msec 4 msec
ICMPv6: Received Unreachable code 4, Src=2001:DB8:0:9999::9, Dst=2001:DB8::5
!
%ASA-6-302015: Built outbound UDP connection 23263 for
out6:2001:db8:0:9999::9/33434 (2001:db8:0:9999::9/33434) to
inside6:2001:db8::5/53239 (2001:db8::5/53239)
%ASA-6-302020: Built inbound ICMP connection for faddr
2001:db8:0:9999::9/0 gaddr 2001:db8::5/0 laddr 2001:db8::5/0

! Telnet from 2001:db8::5 (R5) to 2001:db8:0:9999::9 (R9)

ASA1# %ASA-6-302013: Built outbound TCP connection 23196 for
out6:2001:db8:0:9999::9/23 (2001:db8:0:9999::9/23) to
inside6:2001:db8::5/17549 (2001:db8::5/17549)

ASA1# show conn detail protocol tcp
7 in use, 35 most used
Flags: A - awaiting inside ACK to SYN, a - awaiting outside ACK to SYN,
      B - initial SYN from outside, b - TCP state-bypass or nailed, C - CTIQBE
media,
      D - DNS, d - dump, E - outside back connection, F - outside FIN, f -
inside FIN,
      G - group, g - MGCP, H - H.323, h - H.225.0, I - inbound data,
      i - incomplete, J - GTP, j - GTP data, K - GTP t3-response
      k - Skinny media, M - SMTP data, m - SIP media, n - GUP
      O - outbound data, P - inside back connection, p - Phone-proxy TFTP con-
nection,
      q - SQL*Net data, R - outside acknowledged FIN,
      R - UDP SUNRPC, r - inside acknowledged FIN, S - awaiting inside SYN,
      s - awaiting outside SYN, T - SIP, t - SIP transient, U - up,
      V - VPN orphan, W - WAAS,
      X - inspected by service module
TCP out6:2001:db8:0:9999::9/23 inside6:2001:db8::5/17549,
      flags UTO, idle 14s, uptime 23s, timeout 1h0m, bytes 306

```

Example 16-34 focuses on characterizing ASA's application awareness for protocols such as HTTP and FTP.

- If the **http inspect** command is configured, ASA log messages display information about HTTP URLs accessed inside each connection.
- ASA understands the particularities of FTP operation. This specific example shows an active FTP session from 2001:db8:0:2222::103 to server 2001:db8:0:bbbb::102 in which the user called *user1* retrieved the file *Reg1.exe*.

The term “awareness” means that ASA understands the protocol nature. The difference, when compared to Chapter 12, is that the use of this application knowledge for filtering purposes is not yet supported under IPv6.

Example 16-34 ASA Application Awareness

```
! HTTP connection highlighting ASA's URL visibility
%ASA-6-302013: Built inbound TCP connection 102991 for
out6:2001:db8:0:bbbb::102/1265 (2001:db8:0:bbbb::102/1265) to
dmz6:2001:db8:0:1111::2/80 (2001:db8:0:1111::2/80)
%ASA-5-304001: 2001:db8:0:bbbb::102 Accessed URL 2001:db8:0:1111::2:
http://[2001:db8:0:1111::2]/
!
! User retrieves a file on 2001:db8:0:bbbb::102 using Active FTP

%ASA-6-302013: Built outbound TCP connection 44398 for
out6:2001:db8:0:bbbb::102/21 (2001:db8:0:bbbb::102/21) to
mgmt6:2001:db8:0:2222::103/3111
(2001:db8:0:2222::103/3111)
%ASA-6-302013: Built outbound TCP connection 44407 for
out6:2001:db8:0:bbbb::102/20 (2001:db8:0:bbbb::102/20) to
mgmt6:2001:db8:0:2222::103/3113 (2001:db8:0:2222::103/3113)
%ASA-6-302014: Teardown TCP connection 44407 for
out6:2001:db8:0:bbbb::102/20 to mgmt6:2001:db8:0:2222::103/3113
duration 0:00:00 bytes 39 TCP FINs
%ASA-6-302013: Built outbound TCP connection 44409 for
out6:2001:db8:0:bbbb::102/20 (2001:db8:0:bbbb::102/20) to
mgmt6:2001:db8:0:2222::103/3114 (2001:db8:0:2222::103/3114)
%ASA-6-303002: FTP connection from mgmt6:2001:db8:0:2222::103/3111 to
out6:2001:db8:0:bbbb::102/21, user user1 Retrieved file Reg1.exe
%ASA-6-302014: Teardown TCP connection 44409 for out6:2001:db8:0:bbbb::102/20 to
mgmt6:2001:db8:0:2222::103/3114
duration 0:00:00 bytes 477 TCP FINs
%ASA-6-302014: Teardown TCP connection 44398 for out6:2001:db8:0:bbbb::102/21 to
mgmt6:2001:db8:0:2222::103/3111
duration 0:00:26 bytes 509 TCP FINs
```


Establishing Connection Limits

This section is aimed to review the topic of establishing connection limits for stateful inspection activities performed by Cisco Firewalls. As previously discussed for IPv4, this might be an important resource for combating Denial of Service (DoS) attacks.

Although ASA mechanisms deserve a specific subsection with more detailed analysis, IOS resources are simple to configure and are summarized here:

- The Classic IOS Firewall uses the options **max-incomplete {high | low}** and **one-minute {high | low}** with the **ipv6 inspect** command, in a totally analogous fashion to that demonstrated for the **ip inspect** command within Chapter 9.
- The Zone Policy Firewall specifies the limits within **type inspect** parameter-maps, such as those presented in Example 16-22. The practical behavior of these limits for the ZFW was presented for IPv4 in Chapter 10, “IOS Zone Policy Firewall Overview.”

Setting an Upper Bound for Connections Through ASA

Example 16-35 illustrates the configuration of an upper bound in ASA for connections belonging to a certain traffic class. In this particular case, the acceptable limit of simultaneous Telnet connections arriving on interface *out6* of ASA1 was set to 500 (refer to Figure 16-17). Other details explored in this example follow:

- TCP Sequence Number randomization is disabled for the class TELNET6.
- The default **threat-detection** rates for drops related to the **conn-limit** parameter are changed so that the feature behavior could be demonstrated. The point to emphasize here is that this functionality works exactly in the same way for IPv4 and IPv6. If you need more details about this topic, review Chapter 11.

Example 16-36 not only registers the procedure for limiting the number of simultaneous incomplete TCP connections for a given client but also documents what the firewall sees when the acceptable thresholds are crossed. It is convenient to observe that the **show service-policy** command provides information on a per-class basis and reveals the addresses (either IPv4 or IPv6) that are exceeding the established limits.

Example 16-35 *Setting Connection Limits for a Given Class*

```
! Setting a limit for telnet connections arriving on interface 'out6'
class-map TELNET6
  match port tcp eq telnet
!
policy-map POLICY1
  class TELNET6
    set connection conn-max 500 random-sequence-number disable
!
service-policy POLICY1 interface out6
```

```

!
! What happens when the connections limit is exceeded
%ASA-3-201011: Connection limit exceeded 500/500 for
input packet from 2001:db8:0:bbbb::3cb/20627 to 2001:db8::7/23 on interface out6
!
! Changing default threat-detection rates for drops based on the conn-limit param-
eter
threat-detection rate conn-limit-drop rate-interval 600 average-rate 30 burst-rate
60
threat-detection rate conn-limit-drop rate-interval 3600 average-rate 25 burst-
rate 50
!
%ASA-4-733100: [ Connection limit] drop rate-1 exceeded. Current burst rate is 22
per second, max configured rate is 60; Current average rate is 42 per second,
max configured rate is 30; Cumulative total count is 25320
%ASA-4-733100: [ Connection limit] drop rate-2 exceeded. Current burst rate is
53 per second, max configured rate is 50; Current average rate is 12 per second,
max configured rate is 25; Cumulative total count is 45236
!
ASA1# show service-policy interface out6 set connection detail | include conn
Set connection policy: conn-max 500 random-sequence-number disable
current conns 280, drop 52160
!
ASA1# show asp drop | include conn
Connection limit reached (conn-limit)

```

52160

Example 16-36 Limiting Embryonic Connections on a Per-Client Basis

```

policy-map POLICY1
class TELNET6
set connection per-client-embryonic-max 10
!
%ASA-6-201012: Per-client embryonic connection limit exceeded 10/10 for
input packet from 2001:db8:0:bbbb::1f9/20117 to 2001:db8::7/23 on
interface out6
!
ASA1# show service-policy interface out6 set connection detail
Interface out6:
Service-policy: POLICY1
Class-map: TELNET6
Set connection policy: per-client-embryonic-max 10
drop 0

```

Per client		Embryonic	Total
out6	2001:db8:0:bbbb::1f6	10	11
out6	2001:db8:0:bbbb::1f9	10	10
out6	2001:db8:0:bbbb::1fa	10	10

IPv6 and Antispoofing

The evil associated to IP spoofing and the main techniques available to mitigate it were presented in Chapter 11. For IPv6, the considerations are identical and no further theoretical discussion is necessary. Although this section covers the uRPF feature for both ASA and IOS, the task to configure antispoofing with the help of ACLs is left as an exercise for you.

Antispoofing with uRPF on ASA

Example 16-37 shows that the command to enable IPv6 uRPF check on ASA is exactly the same one used for IPv4. This means that, once configured, uRPF works for all source addresses seen on that interface, no matter if the arriving packets use IPv4 or IPv6. In this example, R2 attempts to start a Telnet session using the source address 2001:db8:0:9999::5. ASA's concern here is that this source address should be reachable via interface *out6* and not through *inside6* (refer to Figure 16-17).

Example 16-37 uRPF in Action on ASA

```
! Enabling uRPF for a given interface on ASA
ASA1(config)# ip verify reverse-path interface inside6
!
! R5 telnets to R2 (2001:db8:0:1111::2) from 2001:db8:0:9999::5 (loopback6)
R5# telnet 2001:db8:0:1111::2 /source-interface loopback6
Trying 2001:DB8:0:1111::2 ...
% Connection refused by remote host
!
! What is seen on ASA

%ASA-1-106021: Deny TCP reverse path check from 2001:db8:0:9999::5 to
2001:db8:0:1111::2 on interface inside6
!
ASA1# show asp drop | include rpf
Reverse-path verify failed (rpf-violated) 1
!
ASA1# show ip verify statistics | include inside6
interface inside6: 1 unicast rpf drops
```

Antispoofing with uRPF on IOS

Example 16-38 examines a situation in which Strict uRPF has been configured for the IOS router V6-FW, in the topology shown in Figure 16-9. IPv6 destinations belonging to network 2001:db8:0:9999::/64 are reachable, from V6-FW's standpoint, using the next

hop 2001:db8:0:bbbb::9 on interface F0/0.911. When a packet with source address 2001:db8:0:9999::5 (sent by R5) arrives on interface F0/1, the uRPF feature comes into play and drops it. The example reinforces the fact that uRPF on IOS relies on the CEF table. (Example 16-1 demonstrates how to globally enable CEF for IPv6 environments.)

Example 16-38 uRPF in Action on IOS

```

! R5 uses a source IPv6 address that belongs to a network connected to R9
R5# ping 2001:db8:0:1111::2 source 2001:db8:0:9999::5 repeat 2
Packet sent with a source address of 2001:DB8:0:9999::5
Success rate is 0 percent (0/2)
!
! The V6-FW IOS router is configured for Strict uRPF checking
interface FastEthernet0/1
  ipv6 verify unicast source reachable-via rx
!
! Routing and forwarding information available on V6-FW

V6-FW# show ipv6 route 2001:db8:0:9999::/64
Routing entry for 2001:DB8:0:9999::/64
  Known via "static", distance 1, metric 0
  Route count is 1/1, share count 0
  Routing paths:
    2001:DB8:0:BBBB::9
      Last updated 1w5d ago
!
V6-FW# show ipv6 cef 2001:db8:0:9999::/64
2001:DB8:0:9999::/64
  nexthop 2001:DB8:0:BBBB::9 FastEthernet0/0.911
!
! Viewing uRPF-related packet drops

V6-FW# IPv6-CEF-Drop: Packet from 2001:DB8:0:9999::5 (Fa0/1) to
2001:DB8:0:1111::2, Verify Unicast Reverse-Path feature
!
V6-FW# show ipv6 traffic | include RPF
      2 RPF drops, 0 RPF suppressed drops
!
V6-FW# show ipv6 interface f0/1 | include verif
IPv6 verify source reachable-via rx
  0 verification drop(s) (process), 2 (CEF)
  0 suppressed verification drop(s) (process), 0 (CEF)

```

Note Loose uRPF is also available for IPv6 on IOS. You just need to replace the `reachable-via rx` parameter by the `reachable-via any` option in the `ipv6 verify unicast source` command. The expected behavior was discussed for IPv4 in Chapter 11.

IPv6 and Fragmentation

Fragmentation handling is one of the areas in which major changes took place when IPv6 was designed. For instance, IPv6 fragmentation is performed only by source nodes and not by intermediate network elements such as routers. IPv6 endpoints typically use the Path MTU Discovery (PMTUD) mechanism to determine the adequate packet size, for transmission along a certain path, so that they can avoid fragmentation. Any packet exceeding the PMTUD calculated value needs to be fragmented by the source.

Besides that change, fragmentation control information was removed from the Base Header and placed on a specific extension header, identified by the value 44 within the Next Header field (of the Base Header).

Figure 16-19 depicts the format of the IPv6 Fragmentation Header and shows a sample 3000-bytes long ICMPv6 packet. This ICMP packet was subject to fragmentation while traveling through a network whose links were configured with 1500-bytes MTU. The fields that constitute the Fragmentation header are explained here:

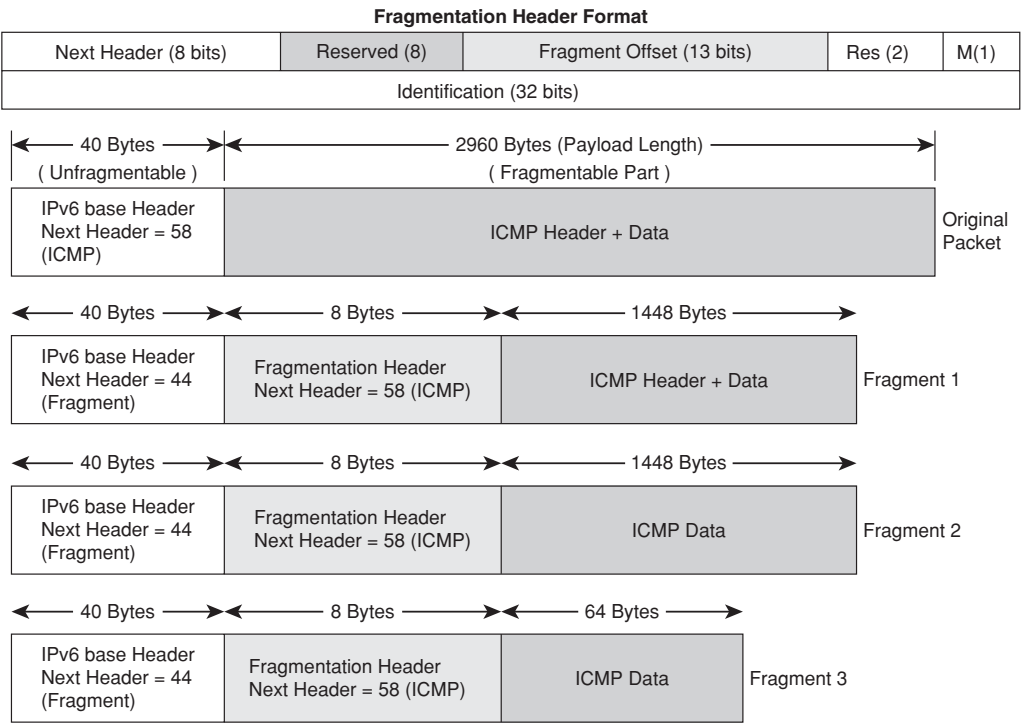


Figure 16-19 *IPv6 Fragmentation Header*

- **Next Header:** Header type of the Fragmentable Part of the original packet. In this example $NH = 58$, because the original packet was ICMPv6.
- **Reserved:** Reserved for future use. Initialized to zero before for transmission and simply ignored when the packet is received.
- **Fragment Offset:** This is a 13-bit unsigned integer. The offset is calculated in *8-octet units*, relative to the start of the Fragmentable Part of the original packet. For instance, with an MTU of 1500 bytes and an IPv6 Base Header of 40 bytes, the amount of bytes allowed to be carried in the data portion of the first fragment is 1456 (and not 1460 because this number is not divisible by 8). This yields a payload length of 1456 for the first fragment and a fragment offset of 182 ($1456 = 8 \times 182$).
- **Res:** Also reserved for future use. Set to zero for transmission; ignored on reception.
- **M bit:** $M = 1$ indicates that there are more fragments and $M = 0$ states that this is the last fragment of the original packet.
- **Identification:** For every packet that requires fragmentation, the source node needs to generate an Identification value. Based on the Identification field, the destination host can determine which incoming fragments belong to a given datagram and buffer all of them until the last fragment is received.

Figure 16-19 divides the header of the original IPv6 packet in *Unfragmentable* and *Fragmentable* parts. According to RFC 2460, the *Unfragmentable Part* is composed of the IPv6 header, plus any extension headers that must be processed by nodes along the path from source to destination, which means all headers up to and including the routing header (if present). The extension headers deemed as unfragmentable are those assuming numbers 1 to 3 (in terms of order of appearance in a packet that carries multiple headers), as documented in the section “The IPv6 Header Format.”

Figure 16-20 complements the theory just summarized by showing an actual ICMPv6 packet that was fragmented according to the definitions of Figure 16-19. Helpful data is also found in Example 16-39, which illustrates the fragmentation process between routers R9 and R10 (acting as IPv6 hosts), in the topology of Figure 16-17.

Original Packet Ethernet Packet: 3014 bytes Dest Addr: 0014.6A21.B4EF, Source Addr: 000F.3461.9621 Protocol: 0x86DD IPv6 Version: 0x6, Traffic_Class: 0x0, Flow_Label: 0x000000, Payload_Length: 2960 Next_Header: 58 , Hop_Limit: 64 Source: 2001:DB8:0:BBBB::9 Dest: 2001:DB8:0:EEEE::10 ICMPv6 Type: 128, Code: 0 (Echo Request) Checksum: 0x5D1D (OK) Identifier: 10E2, Sequence: 0000 Echo Data: [...]	Fragment 2 Ethernet Packet: 1510 bytes Dest Addr: 0014.6A21.B4EF, Source Addr: 000F.3461.9621 Protocol: 0x86DD IPv6 Version: 0x6, Traffic_Class: 0x0, Flow_Label: 0x000000, Payload_Length: 1456 Next_Header: 44 , Hop_Limit: 64 Source: 2001:DB8:0:BBBB::9 Dest: 2001:DB8:0:EEEE::10 IPv6 Option: 44 (Fragmentation Option) Next_Header: 58, Frag Offset: 181 (1448 bytes), Mflag: 1 (More) Identification: 20001 Fragment Data: Echo Data: [Part 2 – suppressed]
Fragment 1 Ethernet Packet: 1510 bytes Dest Addr: 0014.6A21.B4EF, Source Addr: 000F.3461.9621 Protocol: 0x86DD IPv6 Version: 0x6, Traffic_Class: 0x0, Flow_Label: 0x000000, Payload_Length: 1456 Next_Header: 44 , Hop_Limit: 64 Source: 2001:DB8:0:BBBB::9 Dest: 2001:DB8:0:EEEE::10 IPv6 Option: 44 (Fragmentation Option) Next_Header: 58, Frag Offset: 0 (0 bytes), Mflag: 1 (More) Identification: 20001 Fragment Data: ICMPv6 Type: 128, Code: 0 (Echo Request) Checksum: 0x5D1D (OK) Identifier: 10E2, Sequence: 0000 Echo Data: [Part 1 – suppressed]	Fragment 3 Ethernet Packet: 126 bytes Dest Addr: 0014.6A21.B4EF, Source Addr: 000F.3461.9621 Protocol: 0x86DD IPv6 Version: 0x6, Traffic_Class: 0x0, Flow_Label: 0x000000, Payload_Length: 72 Next_Header: 44 , Hop_Limit: 64 Source: 2001:DB8:0:BBBB::9 Dest: 2001:DB8:0:EEEE::10 IPv6 Option: 44 (Fragmentation Option) Next_Header: 58, Frag Offset: 362 (2896 bytes), Mflag: 0 (Last) Identification: 20001 Fragment Data: Echo Data: [Part 3 – suppressed]]

Figure 16-20 Sample IPv6 Packet and Associated Fragments

Note If you need extra information about the PMTUD process for IPv6, please refer to RFC 1981, “Path MTU Discovery for IP version 6.”

Example 16-39 relates to the topology portrayed in Figure 16-17 and was conceived to further illustrate IPv6 fragmentation behavior. R9 (acting as the source host) generates a 3000-bytes long ICMPv6 echo request packet destined to R10 (2001:db8:0:EEEE::10). The Layer 3 hops between source and destination are ASA1 and R5 and, as per RFC 2460 definitions are not involved in fragmentation or in reassembly. They just forward the fragments as individual (and self-contained) packets. Relevant aspects of this example are explained here:

- ASA knows that the three fragments are part of the same packet and shows only one hit count in the interface ACL named OUT6. A packet capture was configured on ASA interface *out6* to provide more insight about the forwarded fragments. A good exercise is to compare this capture with the details presented for the ICMPv6 packet of same length detailed in Figures 16-19 and 16-20.

- R5 was configured with an inbound ACL in its interface connecting to ASA1. This ACL uses the **fragments** keyword, which is aimed to detect noninitial fragments. (ACE #10 in ACL2 shows 02 hit counts and ACE #20 just one match.. It is convenient to emphasize that this access-list was configured with the purpose of providing visibility and not with the goal of filtering noninitial fragments. That is the reason for the **permit** statements in all of its ACEs (as discussed in Chapter 4, “Learn the Tools. Know the Firewall”).
- The perspective of R10, the router acting as the end host, is also shown. The lengths that appear in this debug are in the format X+14, where X is the IPv6 Packet length (base header + payload length of each fragment), whereas the 14 bytes correspond to the Ethernet header. The last two lines in the output refer to the echo reply sent by R10 (over Ethernet 3/0 and using R5 as gateway) after reassembly takes place.

Example 16-39 IPv6 Fragmentation Illustrated

```

R9# ping ipv6
Target IPv6 address: 2001:db8:0:eeee::10
Repeat count [5]: 1
Datagram size [100]: 3000
Timeout in seconds [2]: <Enter>
Extended commands? [no]: yes
Source address or interface: 2001:db8:0:bbbb::9
UDP protocol? [no]: <Enter>
Verbose? [no]: yes
Precedence [0]: <Enter>
DSCP [0]: <Enter>
Include hop by hop option? [no]: <Enter>
Include destination option? [no]: <Enter>
Sweep range of sizes? [no]: <Enter>
Type escape sequence to abort.
Sending 1, 3000-byte ICMP Echos to 2001:DB8:0:EEEE::10, timeout is 2 seconds:
Packet sent with a source address of 2001:DB8:0:BBBB::9
Reply to request 0 (16 ms)
Success rate is 100 percent (1/1), round-trip min/avg/max = 16/16/16 ms

! Fragmented packet from ASA's standpoint

%ASA-6-106100: access-list OUT6 permitted icmp out6/2001:db8:0:bbbb::9(128) ->
inside6/2001:db8:0:eeee::10(0) hit-cnt 1 first hit [0x824f1819, 0xe30120cc]
!
ASA1# show capture CAPTURE2 detail
3 packets captured

```



```

1: 23:22:33.627576 000f.3461.9621 0014.6a21.b4ef 0x8100 1514:
802.1Q vlan#911 P0 2001:db8:0:bbbb::9 > 2001:db8:0:eeee::10: frag
(0x000000bd:0!1448) icmp6: echo request
(len 1456, hlim 64)
2: 23:22:33.627576 000f.3461.9621 0014.6a21.b4ef 0x8100 1514:
802.1Q vlan#911 P0 2001:db8:0:bbbb::9 > 2001:db8:0:eeee::10: frag
(0x000000bd:1448!1448)
(len 1456, hlim 64)
3: 23:22:33.627576 000f.3461.9621 0014.6a21.b4ef 0x8100 130:
802.1Q vlan#911 P0 2001:db8:0:bbbb::9 > 2001:db8:0:eeee::10: frag
(0x000000bd:2896!64) (len 72, hlim 64)
3 packets shown

! R5's perspective (intermediate router with interface ACL configured)

%IPV6_ACL-6-ACCESSLOGDP: list ACL2/20 permitted icmpv6
2001:DB8:0:BBBB::9 -> 2001:DB8:0:EEEE::10 (128/0), 1 packet
%IPV6_ACL-6-ACCESSLOGDP: list ACL2/10 permitted icmpv6
2001:DB8:0:BBBB::9 -> 2001:DB8:0:EEEE::10 (0/0), 1 packet
%IPV6_ACL-6-ACCESSLOGDP: list ACL2/10 permitted icmpv6
2001:DB8:0:BBBB::9 -> 2001:DB8:0:EEEE::10 (0/0), 1 packet
!
R5# show access-list ACL2
IPv6 access list ACL2
    permit ipv6 host 2001:DB8:0:BBBB::9 host 2001:DB8:0:EEEE::10 log fragments
(2 matches) sequence 10
    permit icmp host 2001:DB8:0:BBBB::9 host 2001:DB8:0:EEEE::10 echo-request log
(1 match) sequence 20
!
! R10's perspective (end host)

IPv6: source 2001:DB8:0:BBBB::9 (Ethernet3/0)
    dest 2001:DB8:0:EEEE::10
    traffic class 0, flow 0x0, len 1496+14, prot 44, hops 63, forward to ulp
IPv6: source 2001:DB8:0:BBBB::9 (Ethernet3/0)
    dest 2001:DB8:0:EEEE::10
    traffic class 0, flow 0x0, len 1496+14, prot 44, hops 63, forward to ulp
IPv6: source 2001:DB8:0:BBBB::9 (Ethernet3/0)
    dest 2001:DB8:0:EEEE::10
    traffic class 0, flow 0x0, len 112+14, prot 44, hops 63, forward to ulp
IPv6: nexthop 2001:DB8:0:EEEE::5,
IPv6: Sending on Ethernet3/0

```

After becoming acquainted with the IPv6 fragmentation process, it is now time to revisit the mechanism of Virtual Fragment Reassembly (VFR), available on both ASA and IOS. The related theory behind VFR was analyzed in Chapter 11 for IPv4.

Virtual Fragment Reassembly on ASA

Example 16-40 was designed to show VFR in action for ASA. In the first situation, only two fragments per packet are allowed. Any packet divided in more than two fragments is detected and dropped. The second part of the example portrays a situation in which the amount of packets in the reassembly queue exceeded the tolerable limit of 200.

Example 16-40 ASA Virtual Fragment Reassembly in Action

```
! Allowing a maximum of 02 packets per Fragment Set
ASA1# show fragment out6
Interface: out6
    Size: 200, Chain: 2, Timeout: 5, Reassembly: virtual
    Queue: 0, Assembled: 0, Fail: 3, Overflow: 0
%ASA-4-209005: Discard IP fragment set with more than 2 elements:
src = 2001:db8:0:bbbb::9, dest = 2001:db8:0:eeee::10, proto = IPv6-ICMP, id =
558759936

! Too many fragments requiring reassembly

%ASA-4-209003: Fragment database limit of 200 exceeded: src = 2001:db8:0:bbbb::9,
dest = 2001:db8:0:eeee::10, proto = IPv6-ICMP, id = 2119237632

ASA1# show fragment out6
Interface: out6
    Size: 200, Chain: 24, Timeout: 30, Reassembly: virtual
    Queue: 200, Assembled: 0, Fail: 1664, Overflow: 3455
```

Virtual Fragment Reassembly on IOS

Example 16-41 shows how to enable IPv6 VFR on IOS and displays the initial situation of the reassembly table (no change in any parameters and no fragments in the queue). The example also registers the occurrence of two events related to VFR:

- An IPv6 packet composed of more than 02 fragments was dropped. The acceptable number of fragments per packet was set with the **max-fragments** parameter.
- The number of concurrent reassembly sessions, defined with the **max-reassemblies** parameter, was exceeded.

Example 16-41 *IOS Virtual Fragment Reassembly in Action*

```

interface FastEthernet0/0.911
  ipv6 virtual-reassembly in
!
V6-FW# show ipv6 virtual-reassembly f0/0.911
%Interface FastEthernet0/0.911 [in]
  IPv6 configured concurrent reassemblies (max-reassemblies): 64
  IPv6 configured fragments per reassembly (max-fragments): 16
  IPv6 configured reassembly timeout (timeout): 3 seconds
  IPv6 configured drop fragments: OFF

  IPv6 current reassembly count:0
  IPv6 current fragment count:0
  IPv6 total reassembly count:0
  IPv6 total reassembly timeout count:0
!
! Establishing a limit for the number of fragments per packet

interface FastEthernet0/0.911
  ipv6 virtual-reassembly in max-fragments 2
!
%IPV6_VFR-4-TOO_MANY_FRAGMENTS: Too many fragments per datagram (more than 2) -
sent by 2001:DB8:0:BBBB::3F2, destined to 2001:DB8:0:EEEE::10
!
! Limiting the number of simultaneous datagrams being reassembled

interface FastEthernet0/0.911
  ipv6 virtual-reassembly in max-reassemblies 10 timeout 8
!
%IPV6_VFR-4-FRAG_TABLE_OVERFLOW: the fragment table has reached its
maximum threshold 10
!
V6-FW# show ipv6 virtual-reassembly | include current
  IPv6 configured concurrent reassemblies (max-reassemblies): 10
  IPv6 current reassembly count:10
  IPv6 current fragment count:22

```

Summary

After briefly reviewing IPv6 connectivity topics such as address categories and Neighbor Discovery, this chapter examines the main IPv6 security resources currently available on Cisco firewalls:

- The ACL filtering resources on IOS and ASA-based platforms
- Stateful Inspection of IPv6 in the Classic IOS Firewall and its evolution toward the Zone-based Policy Firewall approach
- ASA IPv6 Stateful Inspection resources
- Antispoofing with uRPF technique
- Theoretical review of IPv6 Fragmentation and their application to practical scenarios

Chapter 17, “Firewall Interactions,” presents some design considerations for firewall placement in IPv4 to IPv6 transition environments. This analysis complements the topics explored in the current chapter.

Although IPv6 is not widely deployed yet, Cisco has made a commitment to adapt its products to fully support this protocol. You are encouraged to stay tuned and quickly become aware of new feature developments on Cisco platforms.

Further Reading

Deploying IPv6 Networks (Ciprian Popoviciu, Eric Levy-Anegnoli, Patrick Grossetete)

<http://www.ciscopress.com/bookstore/product.asp?isbn=1587052105>

IPv6 Security (Scott Hogg, Eryc Vyncke)

<http://www.ciscopress.com/bookstore/product.asp?isbn=1587055945>

This page intentionally left blank

Firewall Interactions

This chapter covers the following topics:

- Firewalls and Intrusion Prevention Systems
- Firewalls and Quality of Service
- Firewall and Private VLANs
- Firewalls and Server Load Balancing
- Firewalls and Virtual Machines
- Firewalls and IPv6 Tunneling Mechanisms
- Firewalls and IPSec VPNs
- Firewalls and SSL VPNs
- Firewalls and MPLS Networks
- The Borderless Networks vision

“As far as we can discern, the sole purpose of human existence is to kindle a light in the darkness of mere being.”—Carl Jung

This final chapter provides some information about the typical interactions of firewall functionality with other features (or systems) that might add value to the overall security practice.

In some cases, the definition of *interaction* has more to do with the challenges that should be taken into account when deploying firewalls in some specific environments. This is what happens, for example, in the “Firewalls and IPv6 Tunneling Mechanisms” section.

Even though the list of interactions is by no means complete, they hopefully can provide some helpful information about the conceptual level of security and eventually motivate you to search for additional combinations.

Firewalls and Intrusion Prevention Systems

There is not much questioning about the capability of intrusion prevention systems (IPS) to complement the security functionality provided by stateful firewalls. Notwithstanding, there is controversy about the placement of IPS devices for the firewall (the classic *before or after the firewall* reference, as shown in Figure 17-1). There have been many philosophical (and passionate) discussions about this topic, and some people even say that the outside IPS “*protects*” the firewall or limits the connection requests that arrive at the firewall (in their particular environment).

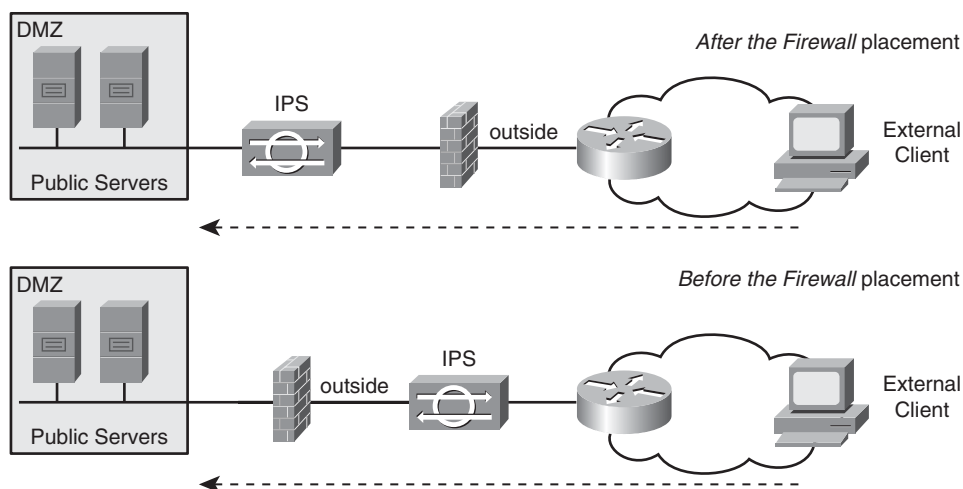


Figure 17-1 Basic Options for IPS Placement

Reflecting for a while about common network devices and their scope of operation within the TCP/IP layered model, you can conclude that job complexity (and consequent impact on resources such as CPU) increases in the following order: Layer 2 switch > router > stateful firewall > IPS.

An IPS concentrates most of its analysis at the network, transport, and application layers, possibly maintaining state (*stateful pattern matching*) and performing anomaly detection, while still having to forward packets (at the data link layer). These many types of tasks that are part of the typical job of IPS systems end up meaning that their highest achieved performance at a given point in time is approximately 4 degrees of magnitude lower than that of the largest router or a Layer 2 switch and roughly 10 times lower than the largest stateful firewall. The latter comparison is even more favorable to firewalls if the metrics of connections per second (CPS) and packets per second (PPS), rather than mere throughput, are taken into account (as previously discussed in Chapter 2, “Cisco Firewall Families Overview”).

Another way to look at this is to consider that if it were possible to build an IPS with the same performance of the largest available router, the cost per protected Gbps on the IPS would certainly be larger than the cost per forwarded Gbps on the router side.

So, where does this perception of higher performance of an IPS, when compared to a stateful firewall, come from? Is the outside IPS the most natural placement?

First, in a significant part of the organizations, the IPS solution has been purchased (and deployed) on a later stage than the firewall in place. This phase shift between the IPS and firewall projects may have led to a situation in which the IPS was selected from the latest offers in the industry, whereas the firewall (performance-wise) was a bit outdated.

Second, the parameters used for the analysis were probably limited to raw throughput, therefore overlooking that IPS solutions behave as stateful elements most of the time. And as you already know, CPS is a key attribute for any stateful device.

A metaphor that helps explain the ideal firewall-IPS interaction involves a check-in and boarding process in an airport:

- A passenger first goes to the airline ticket counter and presents one credential that with the purchased ticket will *open the door* to the boarding area. This access control scheme corresponds to the firewall role.
- In the second stage, those that have been granted with basic access (boarding pass) are supposed to undergo more *detailed inspection* (such as X-ray screening) before they can reach the boarding gate. This second layer of inspection, which is centered only on those that were initially allowed, suggests an analogy with the IPS job. If you do not agree with this illustration, try to picture the scenario in which everybody in an airport must initially be subject to the screening, regardless of being actual passengers. Would this be a self-imposed Denial of Service (DoS)?

In summary, the IPS after the firewall design, as shown in Figure 17-2, can be described as follows:

- The firewall selects acceptable traffic according to combinations of IP addresses and ports and keeps track of connections using a state table.
- The IPS inspects only the traffic forwarded by the firewall and can do a more focused analysis, avoiding the waste of CPU resources to block packets that did not comply with simpler rules. For example, you do need to employ sophisticated IPS mechanisms to block a packet that was sent by an invalid source or to a service port that should not be visible externally. The firewall achieves that protection with much less effort.
- In such a placement, firewalls typically block a lot of packets before they can even be directed to the IPS. This contributes to the reduction of events being shown on the IPS management console, enabling the administrator to spend more time on the events that matter.

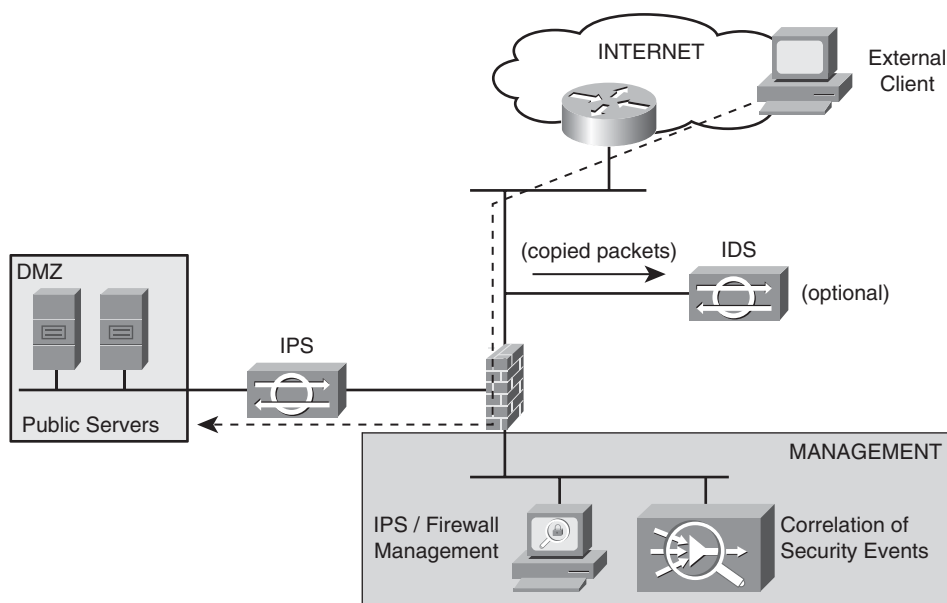


Figure 17-2 *The Ideal Combination of Firewalls and Dedicated IPS Devices*

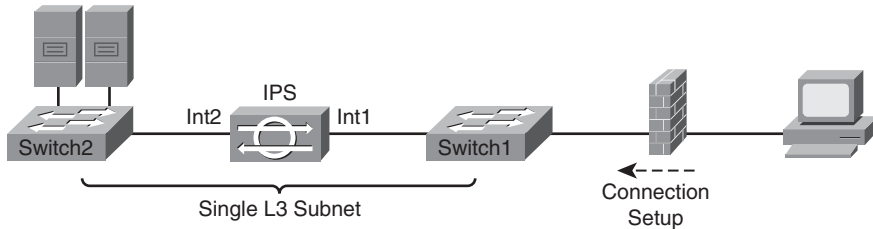
- An intrusion detection system (IDS) device can be optionally deployed *before the firewall*, in promiscuous mode, to provide additional monitoring capabilities. This might provide insight about attackers gathering information about potential targets that do not necessarily reside in the internal area protected by the firewall.
- In this type of environment, solutions that can correlate security events generated by firewalls, IPS, and IDS solutions add great value to security operations.

Figure 17-3 shows more details about the underlying Layer 2 topology for the inline deployment of a dedicated IPS appliance:

- The IPS can be used with two distinct physical interfaces connected to two different switches. The firewall and the protected servers are on different VLANs.
- The IPS can be deployed in a mode sometimes referred to as *inline in-a-stick*. The IPS connects to the switch using a single physical interface (operating in 802.1Q trunk mode) and bridges traffic between a pair of VLANs. The protected servers and the firewall reside on two distinct VLANs.

Note Cisco IPS appliance models that support multiple monitoring interfaces can be deployed in hybrid model: one interface in Promiscuous mode and others in Inline mode. The IPS and IDS appliances shown in Figure 17-2 can be the same physical device.

Inline IPS using 02 physical interfaces



Inline IPS using a VLAN-pair

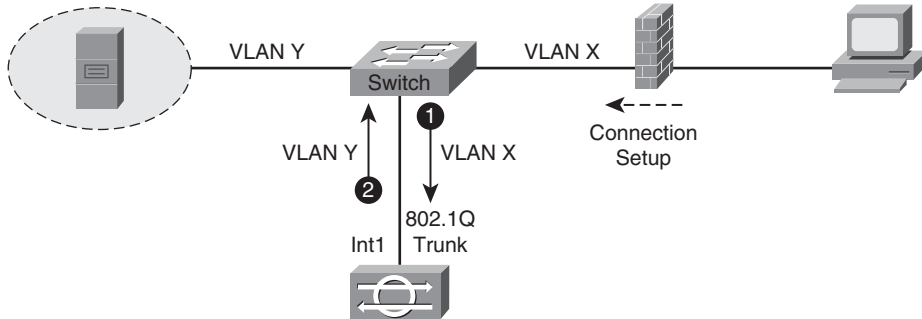
**Figure 17-3** More Details About External IPS Placement

Figure 17-4 reveals how an ASA appliance (ASA5520 in this particular example) sees an installed Advanced Inspection and Prevention Security Services Module (AIP-SSM) (SSM-20 in this case). ASA has an internal backplane connection to the IPS services module.

Figure 17-5 illustrates the two logical deployments associated with an AIP-SSM module installed in an ASA appliance:

- **Inline mode:** The IPS is in the traffic path for all the packets sent by ASA to it. Traffic can be dropped either by the firewall or IPS policy checks. In the example, an access-list is used to avoid sending encrypted traffic (HTTPS and IPsec ESP) to the IPS. This choice keeps the IPS from receiving traffic for which it could not provide value-added inspection services, therefore saving precious CPU cycles.
- **Promiscuous mode:** A copy of the traffic selected by ASA is sent to the IPS module. In this mode, the IPS can send TCP resets to terminate connections not compliant with the configured rules or instruct the ASA to shun undesirable traffic.

Note Cisco ISR G2 routers support the enhanced network module for IPS (NME-IPS), which uses the same inspection code and signature available for dedicated IPS appliances (Cisco 42xx series) and AIP-SSM cards. The interaction between the Cisco IOS router and the NME is similar to that presented for ASA and its associated AIP module.

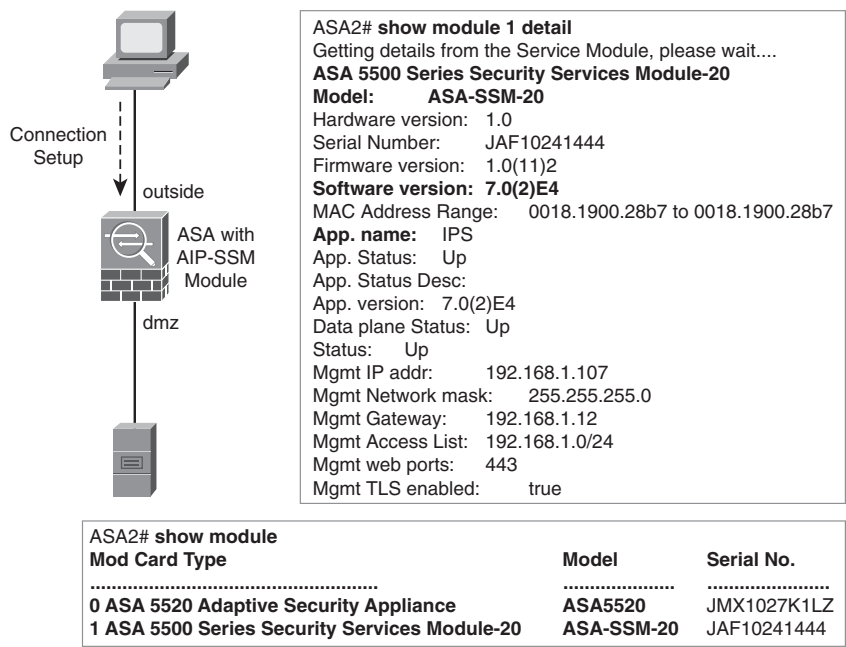


Figure 17-4 ASA with AIP-SSM Module

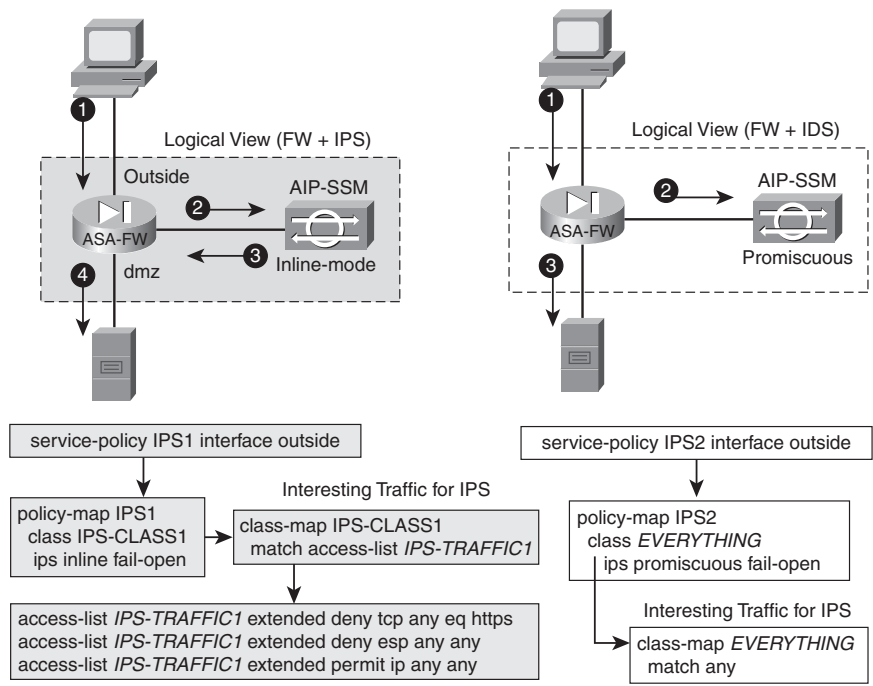


Figure 17-5 ASA Designs Using AIP-SSM Module

Note Another option for adding IPS functionality to Cisco router-based firewalls is the Cisco IOS IPS solution. This is an inline, software-based IPS implementation that can be easily integrated with either CBAC or ZFW to provide a more complete security solution for branch offices.

Firewalls and Quality of Service

Quality of service (QoS) functionality is concerned with offering different service levels for network-based applications that have disparate requirements for parameters such as bandwidth, delay, jitter, and packet drop probability. The building blocks for Diffserv, the most widely deployed QoS model, are shown in Figure 17-6.

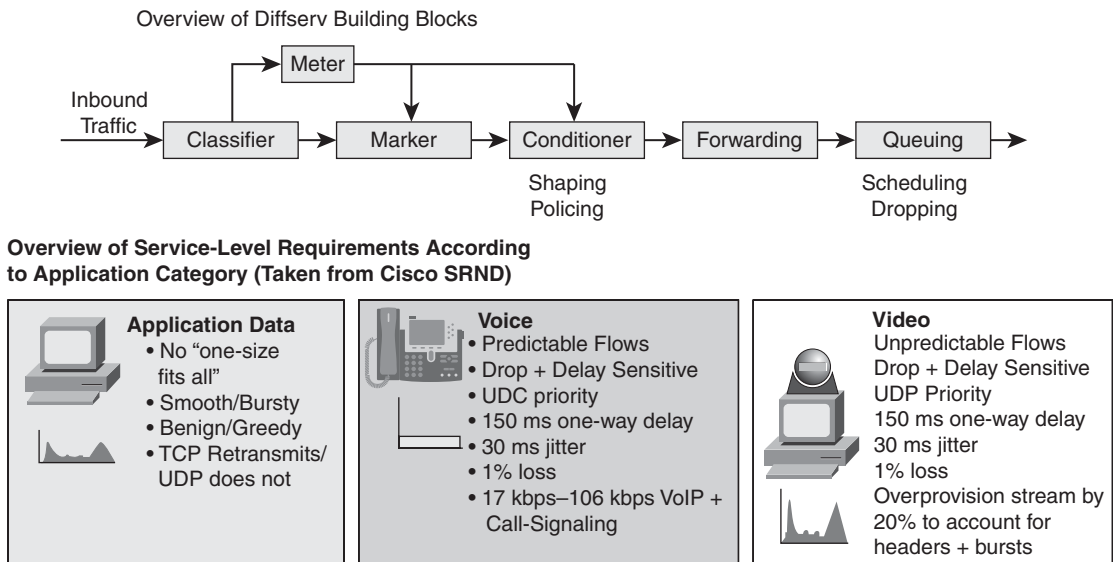


Figure 17-6 Diffserv Building Blocks and Service-Level Requirements per Application Category

QoS has been historically associated with the conception, design, and implementation of convergent networks that can support the (harmonic) transport of data, voice, and video applications. These networks are a business fact (and not just a technical trend), and the evolution of the QoS feature set on switches and routers has been key for this success. A summary of the service-level requirements on a per application category basis is shown in Figure 17-6. (This model was obtained from the Cisco SRND design documents.)

Detailing QoS theory is way beyond the scope of this book. Including this possible firewall interaction in the last chapter raises awareness about the broad set of QoS mechanisms available on Cisco IOS routers and Cisco Catalyst and Nexus switches.

If you take advantage of the wealth of features and performance level available on Cisco routers and switches in an arrangement such as the one in Figure 17-7, it is possible, for instance

- To control the volume of traffic arriving on a firewall interface. This helps ensure that the bps and pps performance parameters of the firewall are not exceeded. If you compare this approach with the idea of using the IPS to limit traffic directed to the firewall, the former can prove more flexible, appropriate, and effective.
- To limit the amount of traffic placed in the so called *scavenger* class therefore helping on the contention of worm propagation. Packets falling in the premium classes (business applications, real time, and the like), or even in the best effort class, have a guaranteed amount of bandwidth, whereas the remaining traffic (nonclassified or deemed abnormal) is constrained by the bandwidth assigned to the scavenger class.

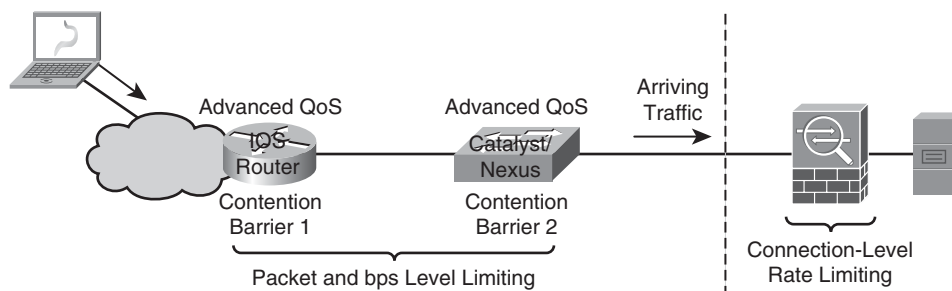


Figure 17-7 Interaction of Firewalls with QoS Features Provided by Routers and Switches

As discussed in Chapter 2, one of the most relevant firewall performance metrics is CPS (connections per second). Other chapters have taught how to limit the number of total and embryonic connections on a per-client or per-server basis, and how to handle timeout values for TCP- or UDP-based applications. By combining these resources, it becomes possible to somewhat influence the rate of connections being processed by the firewall (even on a per-class basis).

This collaborative work of the various network devices is useful for minimizing the impact of worm propagation or denial-of-service (DoS) attacks.

Firewalls and Private VLANs

Because a VLAN is a way to delimit a broadcast domain, it is common to associate one VLAN with one IP subnet. This inherent ability of segmenting subnets has induced many administrators to start using VLANs in the limit case scenario of separating individual hosts.

Although technically possible and still popular, the *one VLAN per user* paradigm can be cumbersome for some designs. For many Data Centers and other large networking environments, 4096 isolated hosts are simply not enough. Besides that, in most cases, there is no specific requirement to provide host segmentation by means of IP subnetting. What actually matters is the resultant isolation effect.

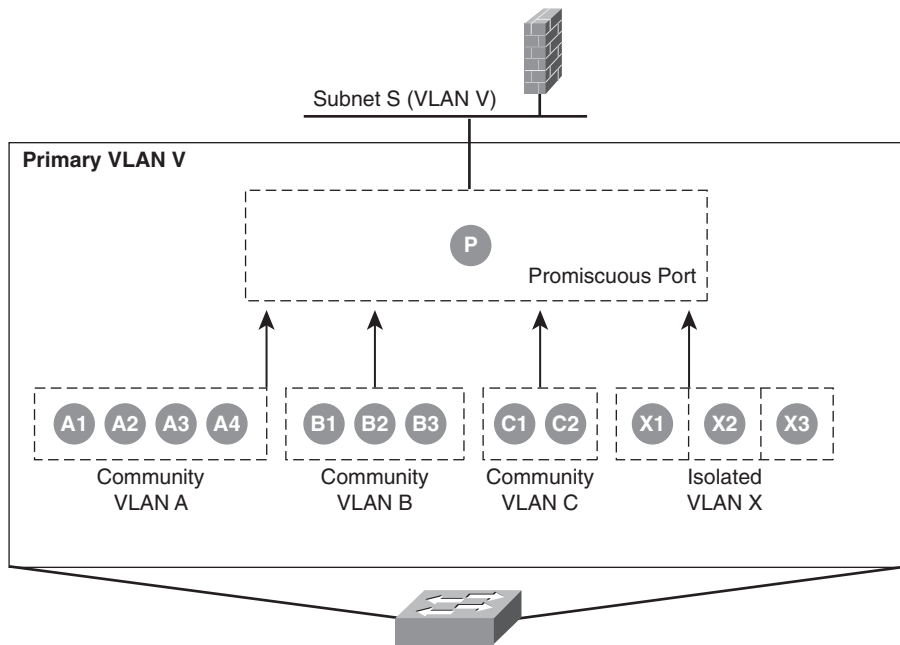


Figure 17-8 Overview of Private VLANs

The Private VLAN (PVLAN) resource, shown in Figure 17-8, introduces an additional layer of segmentation in a Primary VLAN, providing the desired host isolation capability without the need of creating new IP subnets. The main concepts pertaining to the PVLAN feature are defined in the following:

- **Promiscuous Port:** This can communicate with any port in the PVLAN. Only one of these ports, typically used to connect to the default gateway (firewall, router, Layer 3 switch, and so on), is allowed per PVLAN.
- **Community Ports:** These are allowed to communicate with other ports belonging to the same community (a *community VLAN*) and with the promiscuous port.
- **Isolated Ports:** Although they belong to the same broadcast domain (*isolated VLAN*), the isolated ports cannot communicate with each other. Traffic from an isolated port is forwarded solely to the promiscuous port within a primary VLAN.
- **Secondary VLANs:** This is the term used to refer to the isolated and community VLANs.

Figure 17-8 shows a Primary VLAN V that has been subdivided into three community VLANs (A, B, and C) and one isolated VLAN, X. Ports belonging to community A can exchange traffic with other ports in the same community (labeled A1) and with the promiscuous Port P. A totally analogous behavior is associated with ports on communities B and C.

Ports contained in the isolated VLAN X do not talk to each other and always rely on the promiscuous port to forward traffic. In the scheme of Figure 17-8, a firewall is connected to Port P (promiscuous port) and can now establish access control rules for secondary VLAN interconnection. It is important to remember that all the PVLAN hosts are assigned addresses that belong to the same IP subnet.

Private VLANs are useful on Demilitarized Zones (DMZ) because they can separate servers at Layer 2 while still maintaining a single Layer 3 subnet (avoiding the burden of managing multiple host routes). If PVLANS are in place, the eventual compromise of a given server does not immediately provide the attacker with access to adjacent L2 hosts because there is a need to pass through the firewall rules to reach servers that belong to a different community or to an isolated port.

PVLANS can be of great help for worm contention within intranets. Considering that the traditional campus model is to deploy servers in a centralized fashion, it makes sense to isolate client side hosts on each VLAN so that they cannot talk directly to each other. (In most scenarios, there is no traffic interest between two such stations.) If a user's computer is compromised by a worm, it is not allowed to get directly in touch with another host in the same primary VLAN. Rather, it needs to go through the L3 element (connected to the promiscuous port), in which access control rules can be applied. These rules normally limit access to the server VLANs and restrict intrasubnet traffic.

Firewalls and Server Load Balancing

Server Load Balancers (SLB) such as the Cisco Application Control Engine (ACE) have been present on Data Centers since the Internet explosion in the second half of the 1990s. They can scale the performance of websites through the mere increment in the number of servers.

The basic SLB service employs a Virtual IP (VIP) that receives the client-side traffic and, after applying some connection distribution method (*predictor*), selects a real server to take care of the user request. The SLB device periodically sends application-level probes to the real servers to verify if the pertinent service works properly. By doing that, it is possible to avoid the undesirable situation of directing connections to a failed server.

Load balancers have evolved to incorporate sophisticated features such as URL switching and SSL offload, and now are as important to Data Centers as stateful firewalls.

Chapter 6, "Virtualization in the Firewall World," proposed some ways to combine virtual contexts on firewalls and load balancers (ACE module). This section complements that initial analysis by detailing other common interactions between these two types of devices.

Figure 17-9 presents the three classic ways of integrating firewall and SLB devices:

- **Inline SLB:** In this mode (presented in Chapter 6), all the traffic passes through both devices. This design is the most intuitive choice because the Virtual IP (representing the load balanced servers) is in the direct path between client and real server, avoiding

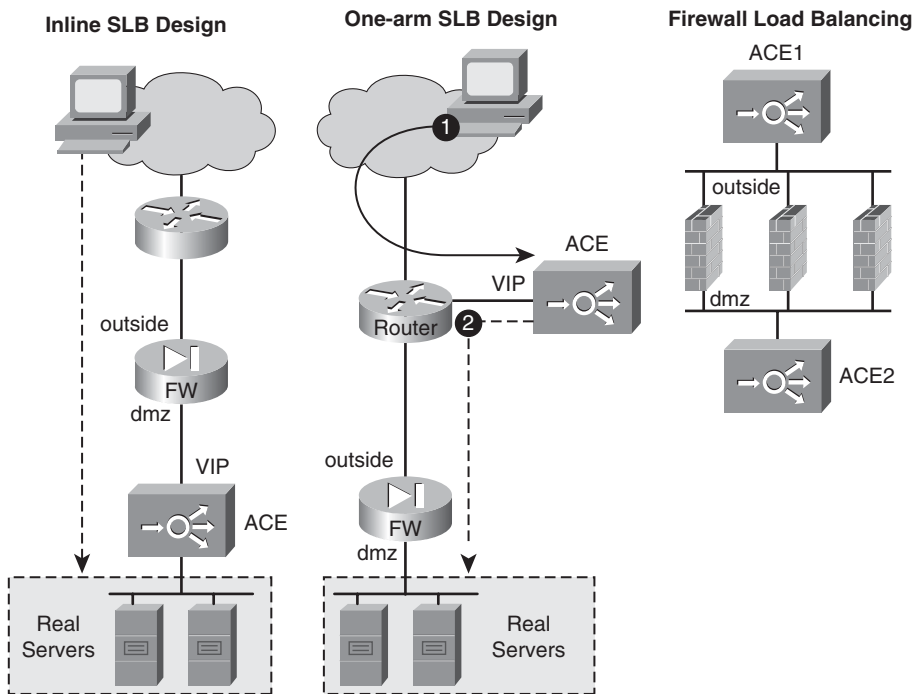


Figure 17-9 Summary of Firewall and SLB Integration Modes

the need for any special routing scheme. Some other aspects pertaining to this option are listed here:

- The firewall has no access to the real servers and, therefore, the ACL rules make reference to the VIP.
- Any additional security feature available in the SLB device can be used for all flows.
- One shortcoming of this scenario is that even traffic that does not require load balancing consumes SLB resources.
- **One-arm SLB:** Although the firewall is always inline with the servers, the SLB can be deployed in this somewhat *parallel mode*. Some characteristics of this design follow:
 - Only traffic that needs load balancing services passes through the SLB device, which ends up saving SLB performance.
 - The firewall rules refer to the addresses of real servers. This is useful for more precise correlation of security events, assuming that it would be more likely for an event correlation system to understand messages generated by a classic Cisco Firewall than supporting events from an SLB device.

- Another interesting aspect associated with this approach resides in the possibility of using the anti-DoS features of the SLB devices (such as SYN cookies) before traffic actually reaches the firewall. By doing so, the packets that arrive at the firewall outside interface are already filtered for this category of attacks, and the overall result is a performance saving on the firewall side.
- Although having the potential to increase aggregate performance, one-arm mode is more complex in terms of routing. One of the possible ways of dealing with the routing issues is further detailed in Figure 17-10.
- **Firewall Load Balancing:** This way to scale firewall performance with the aid of SLB devices is described later in this section.

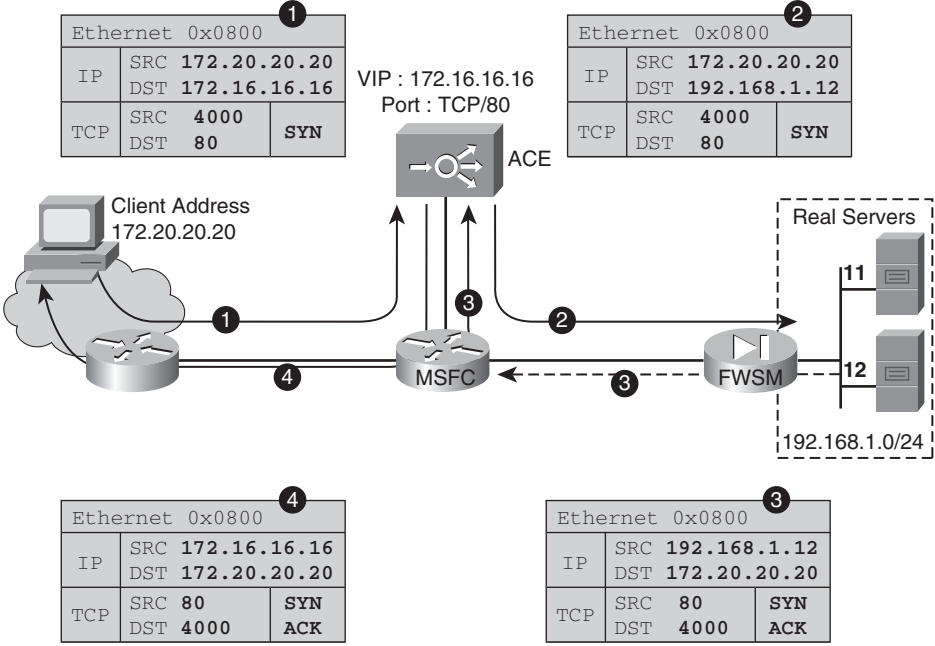


Figure 17-10 Packet Flow in the SLB One-Arm Scenario

You need to take special care when using the one-arm mode. The designer must guarantee that the balanced traffic always passes through the SLB in both directions (client to server and server to client). This does not happen automatically because the router shown in Figure 17-9 does not consider it necessary to send the traffic returning from the real servers through the SLB device (because it has a direct route pointing to the originating client).

To correct this routing issue, you can apply one of the following techniques:

- **Client-side NAT on the SLB device:** The classic operation in SLB scenarios is to perform only server-side NAT (*destination NAT*). If the load balancer is configured for this new class of NAT, it starts translating the client (source) address to one of its addresses. The real server always sees connection requests sourced from the SLB and answers them accordingly. The drawback of this approach relates to lost visibility of original client addresses, which is generally unacceptable from a security and auditing standpoint in many practical environments.
- **Policy Based Routing (PBR):** A router, such as the Multilayer Switch Feature Card (MSFC) in a Catalyst 6500, has to force the return traffic through the SLB device whenever it recognizes it as being sourced from a real server.

The steps involved in the PBR solution are shown in Figure 17-10:

1. Client traffic is routed to the Virtual IP without any change in the source address.
2. The Access Control Engine (ACE) module routes traffic to the selected real server. Destination NAT is performed.
3. The real server sees the connection request as sourced from the original client and simply responds using its default gateway address (the firewall, if deployed in routed mode). The Firewall Services Module (FWSM) points a default route to the MSFC, which uses PBR to direct the traffic to the ACE.
4. The ACE recognizes the return traffic as part of an existing flow and just routes it back to the outside client.

Note PBR may severely affect performance on most of the router platforms that only deal with it at the process level. (Remember that the IP routing paradigm is destination-oriented.) For special scenarios that do not involve recursive routing or use of tunnel interfaces (such as GRE), the Catalyst 6500 equipped with SUP720/MSFC3 has minimal impact. Another scalable platform for PBR is the Nexus 7000.

Many firewall vendors employ clustering mechanisms for performance and scaling purposes. Quite often these technologies are proprietary and sometimes rely on cumbersome flooding schemes. Firewall Load Balancing (FWLB), a technique that counts on SLB devices to scale the use of firewalls, offers an alternative to clustering and is shown in Figure 17-11.

In this case, two active SLBs balance the connections among all the firewalls. Because the firewalls under consideration are stateful in nature, one critical principle must be obeyed: the packets belonging to a given connection must always traverse the same firewall. This is usually achieved by some kind of coherence in the predictor algorithm. Cisco ACE, for example, performs a hashing that takes into account a combination of source and destination IP addresses.

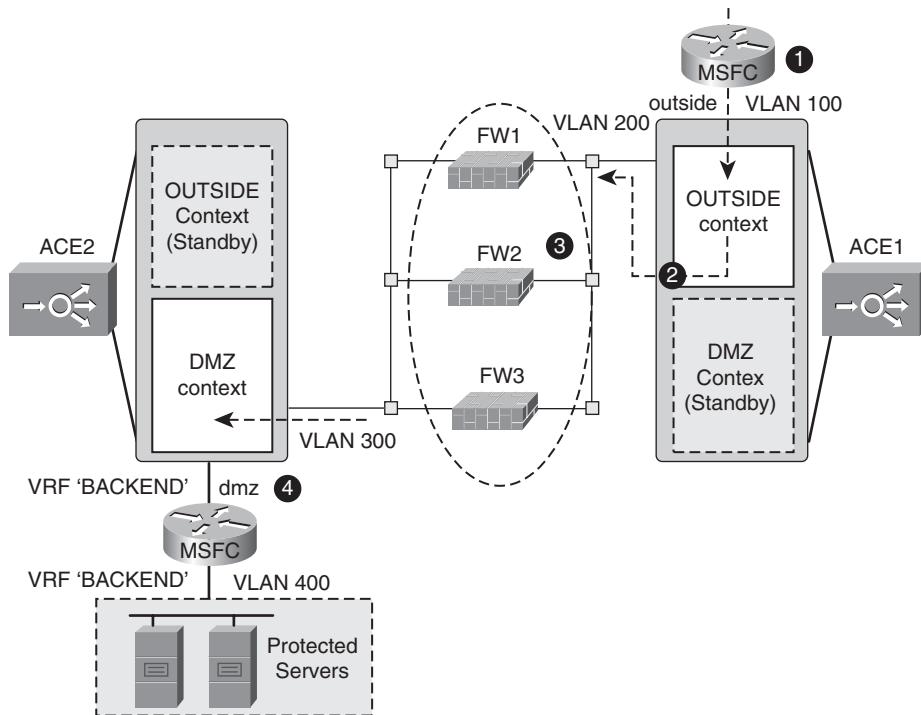


Figure 17-11 Firewall Load Balancing with 02 SLB Devices in Multiple Context Mode

The particular scenario, as shown in Figure 17-11, can be described as follows:

1. Inbound traffic arrives at the Catalyst 6500 chassis where the active ACE OUTSIDE context is installed. The MSFC (Global Table) routes this traffic to ACE1 using VLAN 100.
2. ACE1 load balances traffic to FWSMs over VLAN 200 using a source/destination IP address hash.
3. Traffic traverses the selected FWSM according to the configured rules. If permitted, it is delivered to ACE2 on the active DMZ context via VLAN 300.
4. ACE2 connects to a new Virtual Routing and Forwarding instance (VRF), named BACKEND, to reach the protected servers.

Note When the connection request is destined to a VIP, an ACE configured for FWLB also performs regular SLB.

FWLB has some significant advantages over common firewall clustering techniques:

- All the firewalls are simultaneously active.
- It enables the use of heterogeneous firewalls.

It is important for the SLB device selected for this kind of arrangement to surpass (or at least equalize) the aggregate performance of the firewalls being load balanced. As would be done with a single stateful firewall, the following set of metrics should be taken into account:

- Supported bandwidth
- Packets per second (PPS)
- Concurrent connections
- New connections per second (CPS)

Time for Work With the goal of structuring knowledge you are encouraged to sketch a practical scenario that involves all the firewall interactions discussed so far. *From a Security Design standpoint, how can you take the most advantage of each of the elements or features examined?*

Firewalls and Virtual Machines

The ample use of server virtualization is changing IT in the 21st century. Some would even say that the Virtual Machines (VM) are the new atomic unit for the Next Generation Data Center and its spin-off paradigm, *the cloud*.

Although one of the fundamental design goals of *server virtualization* is transparency to the users and the applications they use, this architecture introduced various concepts such as virtual networking and virtual appliances.

Virtual networking can be defined as the set of functions and devices aimed at delivering networking functionality to virtual machines (VM). Ideally, it is done in *an as close as possible* way to what is available for physical hosts. The intent is to provide administra-

tors with the ability of building complex networks within a single physical host or even across multiple physical hosts.

One of the key components of the virtual networking architecture is the virtual switch. A good representative of the evolution of this type of device is Cisco Nexus 1000V, already outlined in Chapter 6.

A *virtual appliance* is a purpose-specific VM image designed to run on a virtualization platform. These VMs are expected to be easily installed and are usually configured via a web browser. Some popular examples of such a virtual entity are middlewares, web design tools, collaboration tools, management and monitoring platforms, storage devices, and also security elements (naturally including our loyal friend, *the firewall*).

This new *virtual stuff* terminology naturally raises some questions to security designers:

- Is it possible to get the same level of protection when using virtual machines?
- What class of firewall is more suitable for protecting the virtual networking infrastructure? (The firewalls studied so far or a virtual appliance?)

The following discussion outlines the implications of each option, always assuming that the Cisco Nexus 1000V is the virtual switch in place.

Protecting Virtual Machines with External Firewalls

The deployment of an external firewall to control traffic between virtualized servers does not differ that much from the corresponding deployment considering physical servers. The use of VLANs is still recommended in these cases as a way to force the traffic, originated from or destined to a virtual machine, to traverse the firewall. This is shown in Figure 17-12.

As shown in the figure, the VM on VLAN X can communicate with the VMs on VLAN Y if permitted by the firewall policy. By default, the VMs connected to VLAN Y can talk directly to each other, unless the Private VLAN resource comes into the scene. (These two VMs could be connected to isolated ports or to ports residing in different community VLANs.) The behavior of the PVLAN mechanism in the Nexus 1000V virtual switch is identical to what was presented earlier for physical switches.

This firewall deployment model for virtual machines becomes an interesting choice when more *north-south* traffic has to be dealt with, that is, when great part of the VM's communication is maintained with elements located outside of the physical server. This situation is frequent when virtual machines are outsourced within a Service Provider Data Center because there is little probability that a VM owned by a certain customer will try to establish a connection with a VM from another customer.

When the firewall is inserted as an external element, its performance and availability parameters are completely independent of those associated with the virtualized servers. This is interesting if you remember that many customers prefer to sign Service Level Agreements (SLA) related to the firewall performance with their service provider of choice.

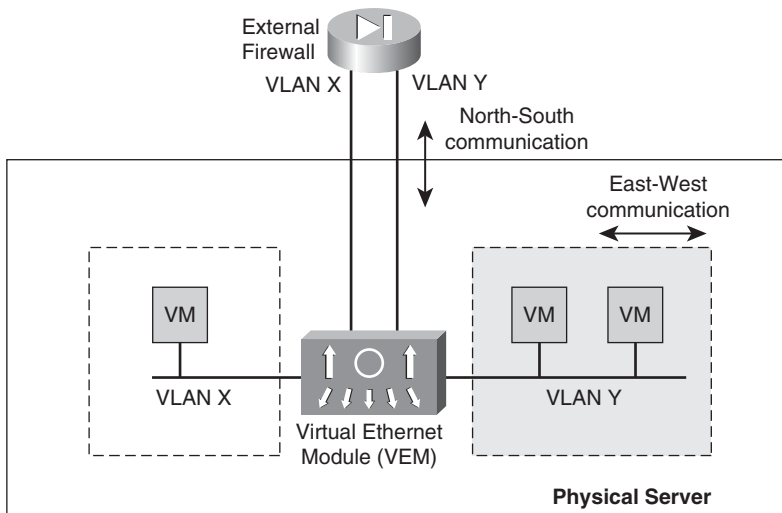


Figure 17-12 *Nexus 1000V and External Firewalls*

One case of particular interest for the external firewalls model is the desktop virtualization design, in which a user desktop is implemented as a hosted virtual machine within a Data Center. This computing model, roughly sketched in Figure 17-13, is getting popular as the cost of managing, updating, and securing desktops is dramatically increasing.

A classic way to access a virtualized desktop in the DC is by using a cheap thin-client that relies on a screen share protocol such as the Remote Desktop Protocol (RDP). In this case, although the external firewall helps protect the RDP connection and the virtualized desktop traffic (to the Internet, for example), the Nexus 1000V makes the virtual desktop interfaces manageable and secure.

Protecting Virtual Machines Using Virtual Firewall Appliances

But what if the traffic in this virtualized Data Center is mainly *east-west*, meaning that VMs communicate with other VMs installed in the same physical server (or cluster of servers)? *Is it still possible to have a firewall protecting the traffic that is confined to a physical server?*

Deploying a firewall as a virtual appliance might be the answer to both questions. For VMs connected to different VLANs inside a physical server, it is just a matter of pointing the default gateway on a given VM to the appropriate interface on the virtual appliance.

For VMs connected to the same VLAN, however, the scenario is a bit more challenging. There should be some means to force the traffic between the VMs to traverse the virtual appliance where the firewall policies are enforced.

The Nexus 1000V virtual switch supports a feature called Virtual Service Domain (VSD), illustrated in Figure 17-14 and that can be quite helpful in the latter situation:

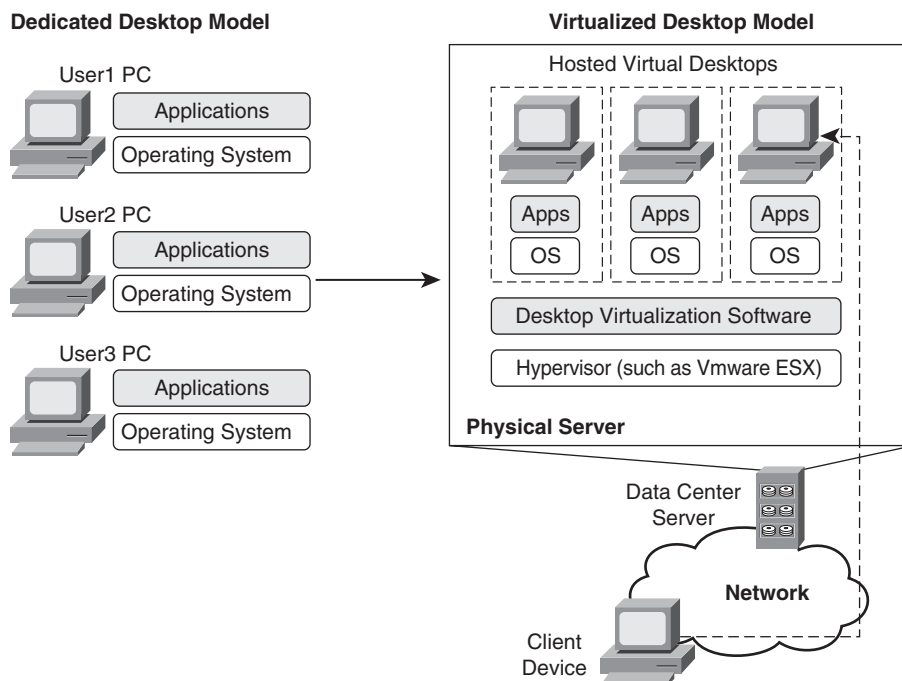


Figure 17-13 Overview of the Virtualized Desktop Model

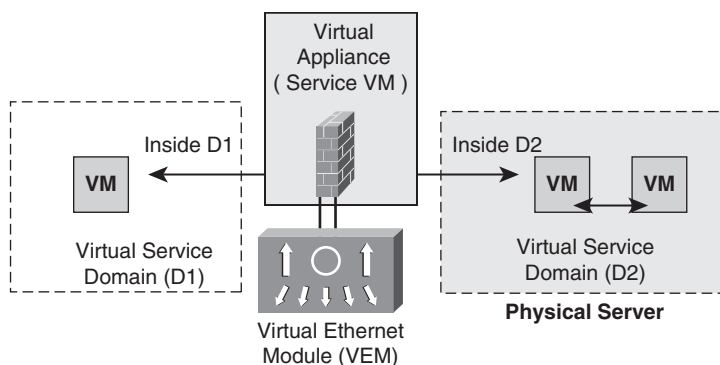


Figure 17-14 Nexus 1000V, Virtual Security Domains and Virtual Appliances

- A VSD is a set of interfaces that are protected by a certain virtual appliance. A VM's interface belonging to a VSD is assigned a sort of security tag that is used for traffic classification.
- A virtual appliance connected to a VSD uses two reference ports for that domain: one *outside* and one *inside*. Any traffic entering the VSD uses the *outside* port, whereas traffic leaving the VSD uses the *inside* port.
- When a VM wants to communicate with a different domain, its traffic is forced to go through a virtual appliance port that belongs to the same VSD.

- Traffic originating and terminating in the same VSD is not required to be diverted to the virtual appliance.
- It is important to emphasize that the concept of VSD takes precedence over the VLAN definition for a given VM. This means that machines on the same VLAN can belong to different VSDs.

VSDs have the flexibility to provide network services by means of any L4-to-L7 virtual appliance. Nevertheless, VSD-based implementations suffer from two drawbacks:

- They are dependent on the performance resources allotted to them by the hypervisor.
- They usually demand a virtual appliance to be installed on every host.

Cisco addressed these issues with the introduction of the *Cisco vPath* architecture. With vPath, the virtual appliance providing the network service of interest is named a *Virtual Service Node (VSN)* and can be placed on any host.

The typical Cisco vPath operation involves two main steps:

- The first packet in a flow that requires a certain network service is redirected to the VSN providing that particular service. This is often referred to as *traffic steering*.
- Upon receiving the VSN's decision for the flow, vPath implements the appropriate network service in all subsequent packets (of the flow) in the Nexus 1000V Virtual Ethernet Module (VEM). This process is called *decision caching* and contributes to accelerating the virtualized network service in the hypervisor kernel.

The first VSN taking advantage of the vPath enhancements is the Cisco *Virtual Security Gateway (VSG)*. In this new firewall model, the control plane (VSG) is completely decoupled from the data plane (Nexus 1000V). The VSG security rules (management plane) are taken care of by the Virtual Network Management Center (VNMC). The basic VSG operation is illustrated in Figure 17-15:

1. A VM starts a communication with another VM.
2. The first packet is steered to the VSG by vPath.
3. VSG sends the appropriate security rules to the Nexus 1000V VEM. This policy controls inter-VM communication. One noteworthy feature associated with this step is *decision caching*: Nexus 1000V remembers the policy sent by VSG and can, thereafter, reduce traffic steering.
4. If permitted by the access rules in place, the first packet arrives at the destination VM.
5. Subsequent packets belonging to the same flow do not pass through the VSG. They are checked against the ACL that was offloaded to the Nexus 1000V.

Unlike other Virtual Firewall Appliances, the VSG does not need to be installed in every physical server. Another interesting feature of VSG resides in its capability to be deployed in an active-standby mode to help ensure a highly available operating environment. In this mode, vPath directs packets to the standby VSG if the active one becomes unavailable.

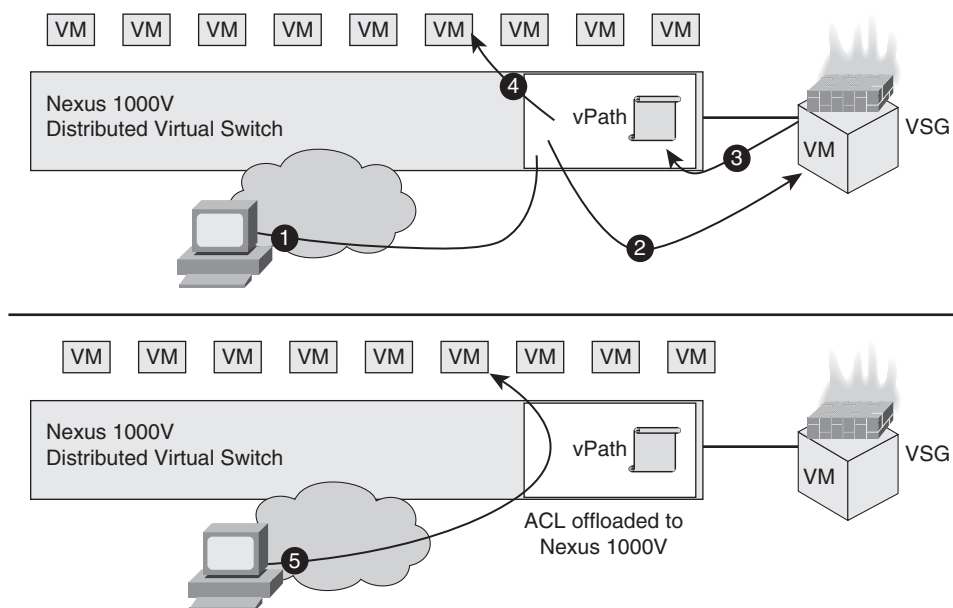


Figure 17-15 Overview of the VSG for the Nexus 1000V

Note The major concern when using the VSG approach is that it is not exactly “stateful” in the classic meaning of the term. It is, however, an important deployment option to be aware of and may fit well for some of your virtualized environments.

Firewalls and IPv6 Tunneling Mechanisms

Chapter 16, “Cisco Firewalls and IPv6,” explored the IPv6 features currently supported on Cisco firewalls. This section complements that knowledge by analyzing the firewall placement in some IPv6 tunneling environments, which are among the most important transition methods available. Although not covering all the tunneling mechanisms, the analysis that follows enables you to grasp the principles involved in the inspection of tunneled traffic.

Figure 17-16 shows a classic IPv6 tunneling environment using Cisco IOS routers:

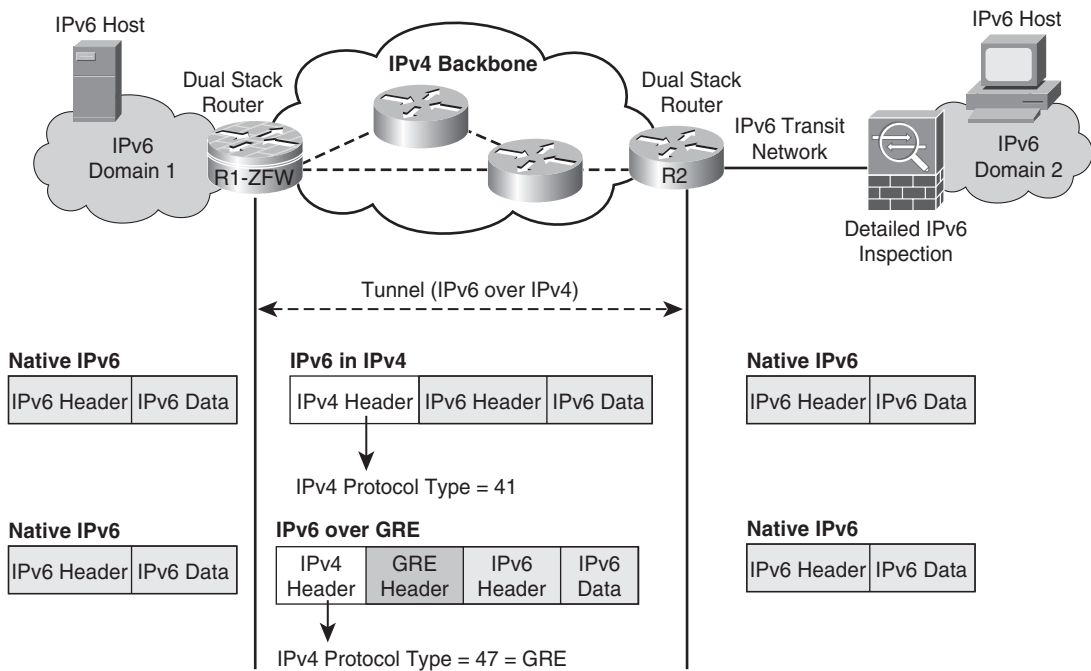


Figure 17-16 Overview of the IPv6 Tunneling Mechanisms

- Two separate IPv6 domains are interconnected using an existing IPv4 backbone.
- The main types of encapsulation are IPv6 over IPv4 (IPv4 protocol field = 41) and Generic Routing Encapsulation (GRE), which is assigned protocol number 47. Even though any tunneling introduces overhead and reduces the usable MTU, tunnels are a practical way of promoting IPv4 to IPv6 migration.
- Filters can be applied to the IPv4-facing interfaces to restrict the types of traffic that flow through them. For example, if these interfaces were to be used just for transition purposes, there could be (stateless) ACLs permitting only those tunnel connections that were originated by a specific set of routers.
- The tunnel is configured between two dual-stack routers, R1 and R2. In Figure 17-16, it is assumed that R2 is in the central site, whereas R1 is deployed in a remote office. With that in mind, performance considerations lead to the selection of an ASA appliance to provide detailed IPv6 inspection for IPv6 domain 2. For the remote site, an IOS router running the Zone-based Policy Firewall (ZFW) is deployed to inspect IPv6 traffic that arrives through the tunnel and is intended to reach IPv6 domain 1.

It is relevant to mention the firewall features in place:

- Origin and destination of tunnels (IPv4 addresses) are restricted with stateless ACLs on router interfaces. Antispoofing techniques (such as those studied in Chapter 11, “Additional Protection Mechanisms”) can bring an additional layer of security.
- Stateful inspection is used to restrict access to the IPv6 domains. These controls apply to the internal addresses. (Those that are exposed after tunneled packets have been decapsulated.)

To provide you with some practical knowledge pertaining to tunnel operations, two methods (static and dynamic 6to4 tunnels) are briefly analyzed here.

Figure 17-17 portrays a network in which Routers R1 and R2 have been configured to use a static 6to4 tunnel. This method is characterized by the **tunnel mode ipv6ip** command at the tunnel interface. The IPv6 network address assigned to the tunnel interfaces is 2001:db8:0:1111::/64, and the tunnel behaves as a point-to-point (L3) link, regardless of the number of hops in the IPv4 backbone.

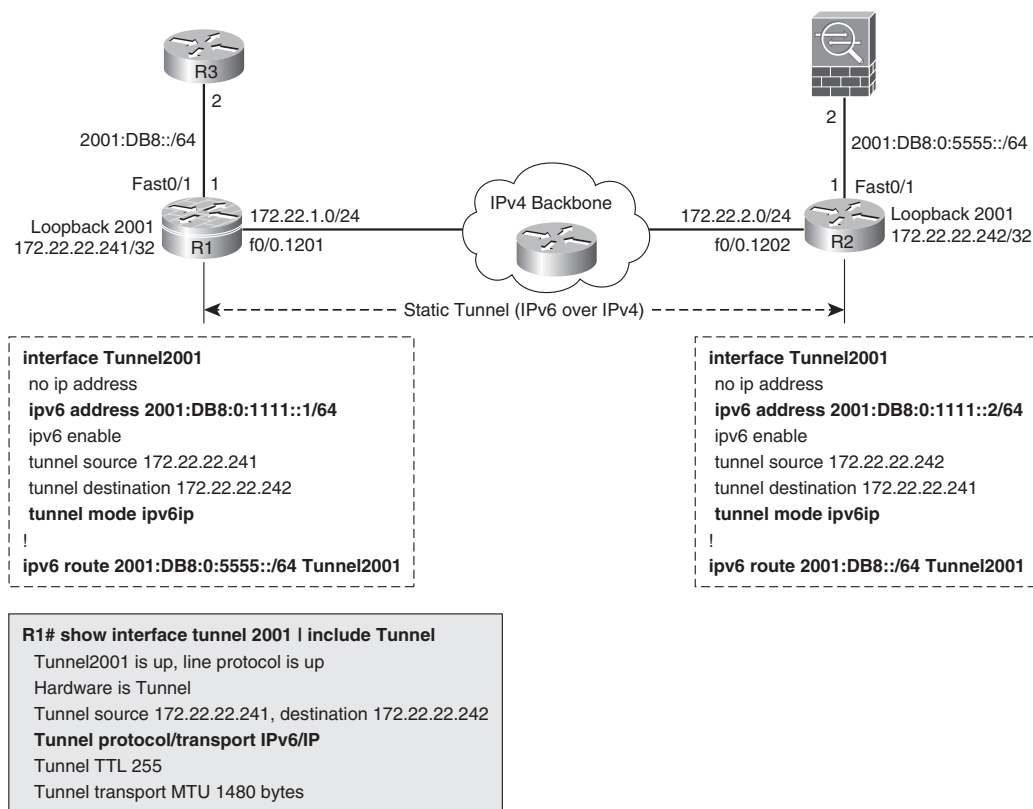


Figure 17-17 Sample IPv6 over IPv4 Static Tunnel

Example 17-1 shows more details about this topology:

- The **show ip cef exact-route** command reveals that R1's tunnel destination (172.22.22.242) is reached via interface f0/0.1201. The **ipv6** version of the command further shows the recursion involved in traffic forwarding via a tunnel. The next-hop address used to reach an IPv6 destination (via tunnel 2001) is the same IPv4 address associated with the tunnel destination.
- R2 sends an IPv6 ping packet to R1 over the tunnel. R1 receives an IPv4 packet that uses protocol 41 over interface f0/0.1201. Upon removal of the tunnel header, the original IPv6 packet is exposed to R1, which can then send the IPv6 echo reply to R2.

Example 17-1 Static IPv6 over IPv4 Tunnel

```
! Determining exit interface that leads to the tunnel destination at R2
(172.22.22.242)
R1# show ip cef exact-route 172.22.22.241 172.22.22.242
172.22.22.241 -> 172.22.22.242 => IP adj out of FastEthernet0/0.1201, addr
172.22.1.1
!
! Determining the exit interface to reach IPv6 address 2001:db8:0:1111::2 (R2)
R1# show ipv6 cef exact-route 2001:db8:0:1111::1 2001:db8:0:1111::2
2001:DB8:0:1111::1 -> 2001:DB8:0:1111::2 => IP adj out of FastEthernet0/0.1201,
addr 172.22.1.1
!
! R2 sends a 1000-bytes IPv6 ping to R1

R2# ping 2001:db8:0:1111::1 source 2001:db8:0:1111::2 repeat 1 size 1000
Packet sent with a source address of 2001:DB8:0:1111::2
Success rate is 100 percent (1/1), round-trip min/avg/max = 8/8/8 ms
!
! R1's perspective (upon receipt of the echo request)
IP: s=172.22.22.242 (FastEthernet0/0.1201), d=172.22.22.241, len
1020, input feature, proto=41, Ingress-NetFlow(18), rtype 0, forus
FALSE, sendself FALSE, mtu 0, fwdchk FALSE

IP: s=172.22.22.241 (local), d=172.22.22.242 (FastEthernet0/0.1201), len 1020,
sending, proto=41
FIBipv4-packet-proc: route packet from (local) src 172.22.22.241 dst 172.22.22.242
FIBfwd-proc: packet routed by adj to FastEthernet0/0.1201 172.22.1.1
```

Note GRE is the default encapsulation used by tunnel interfaces on Cisco routers. In the network of Figure 17-17, simply removing the **tunnel mode** commands from both tunnel interfaces switches the operation mode to GRE. Except from some extra overhead (and the protocol type value of 47), the behavior is virtually identical to that just presented for the IPv6 over IPv4 mode.

Note The IPv6 over GRE over IPv4 encapsulation (shown in Figure 17-16) is needed only when the routing protocol in place is IS-IS. For the great majority of scenarios, basic IPv6 over IPv4 tunneling is a simpler and more efficient option.

Figure 17-18 and Example 17-2 are used together to register some basic information about (dynamic) 6to4 tunnels:

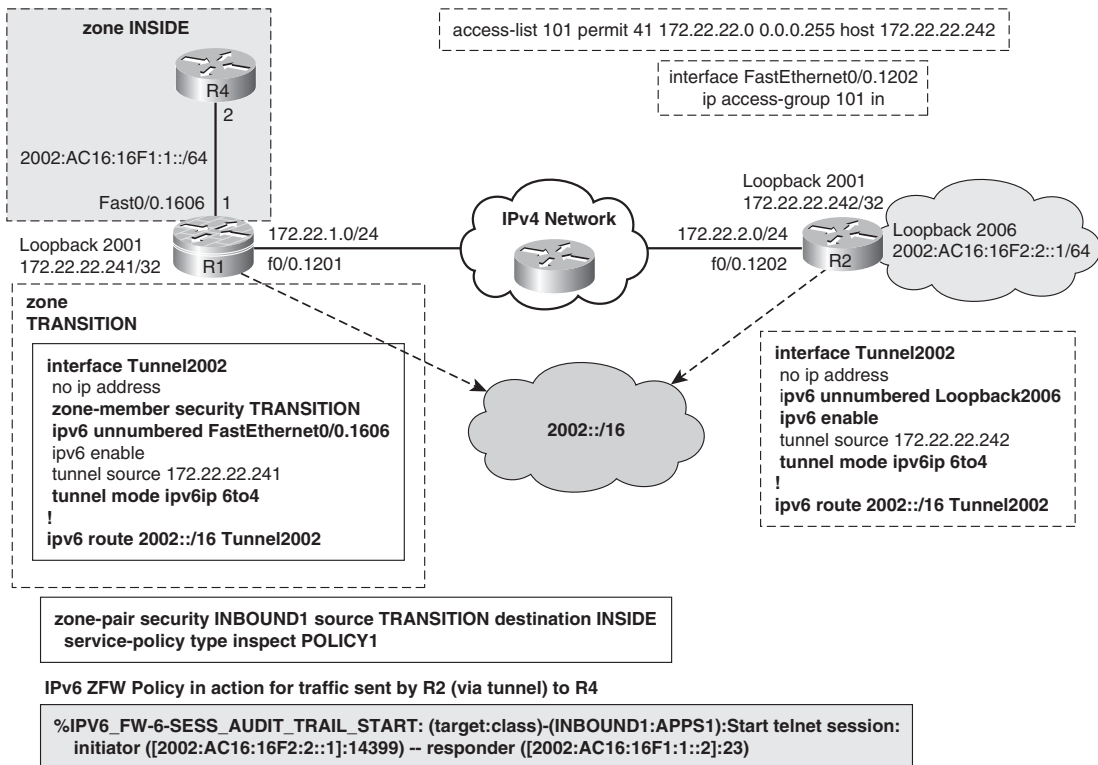


Figure 17-18 Sample 6to4 Dynamic Tunnel

- The 6to4 operation is defined with the **tunnel mode ipv6ip 6to4** command.
- The 2002::/16 prefix is specifically reserved for 6to4 operation.
- Each 6to4 site is automatically assigned a /48 IPv6 prefix that corresponds to the juxtaposition of 2002 and the IPv4 address used by the 6to4 router as its tunnel source. For instance, R1 tunnel source address is 172.22.22.241. Expressing each octet as a hexadecimal number leads to AC.16.16.F1. Concatenating the 2002 prefix and using the IPv6 scheme (fields containing 04 hex digits) yields 2002:AC16:16F1::/48 as the 6to4 prefix for the IPv6 site connected to R1. In a similar fashion, R2 (172.22.22.242) automatically receives the prefix 2002:AC16:16F2::/48.

- Each 6to4 site subdivides the assigned prefix to deal with local addressing needs. For instance, interface f0/0.1606 (connecting R4 to R1) is configured with the prefix 2002:AC16:16F1:1::/64. As another example, Loopback 2006 on R2 is configured with the prefix 2002:AC16:16F2:2::/64.
- Two noticeable configuration aspects of the 6to4 model is the absence of the **tunnel destination** command under the tunnel interface and the existence of a static route to 2002::/16 pointing to the tunnel. Given the scheme used for building 6to4 addresses, this combination is enough to determine the IPv4 address of the edge router in charge of a particular 6to4 (IPv6) prefix.
- Figure 17-18 reveals the existence of two security zones, named INSIDE and TRANSITION, in the IPv6 ZFW arrangement of R1. It is important to emphasize that R1's tunnel interface 2002 (and not the underlying logical interface f0/0.1201) is configured as a member of the TRANSITION zone. This happens because the tunnel interface has the IPv6 visibility.
- A static ACL is applied to R2's f0/0.1202 interface to control the acceptable sources of 6to4 packets. In a similar manner, the same type of ACL could have been defined on R1.

Example 17-2 Sample 6to4 Dynamic Tunnel

```
! Basic information about the 6to4 tunnel
R1# show interface tunnel 2002 | include Tunnel
Tunnel2002 is up, line protocol is up
  Hardware is Tunnel
  Tunnel source 172.22.22.241
Tunnel protocol/transport IPv6 6to4
  Tunnel TTL 255
  Tunnel transport MTU 1480 bytes
!
R1# show ipv6 interface Tunnel 2002
Tunnel2002 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::AC16:16F1
  No Virtual link-local address(es):
  Interface is unnumbered. Using address of FastEthernet0/0.1606
  No global unicast address is configured
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF16:16F1
  MTU is 1480 bytes
!
! Obtaining reachability information for a given IPv6 destination

R1# show ipv6 route 2002:ac16:16fb:b::1
Routing entry for 2002::/16
```

```

Known via "static", distance 1, metric 0
Route count is 1/1, share count 0
Routing paths:
  directly connected via Tunnel2002
    Last updated 00:53:24 ago
!
R1# show ipv6 cef exact-route 2002:ac16:16f1:1::1 2002:ac16:16fb:b::1
2002:AC16:16F1:1::1 -> 2002:AC16:16FB:B::1 => Lookup in table IPv4:Default
!
! Tracing the route to 2002:AC16:16F2:2::1 (behind R2), from R4
R4# trace ipv6 2002:AC16:16F2:2::1
Type escape sequence to abort.
Tracing the route to 2002:AC16:16F2:2::1
  1 2002:AC16:16F1:1::1 4 msec 0 msec 0 msec
  2 2002:AC16:16F2:2::1 4 msec 4 msec 4 msec

```

Some other tunnel-based transition mechanisms are even more automatic in nature than 6to4. Even though the discussion of such methods is beyond the scope of this book, one simple concept should not be forgotten: If a given transition mechanism is not used in your organization, the recommendation is to block it.

Firewalls and IPsec VPNs

Virtual private networks (VPN) extend the reach of corporate networks in a secure manner and without the need to use expensive dedicated circuits. Among the various VPN technologies available, the IPsec framework is certainly the major responsible for the large proliferation of this cost-effective approach of providing connectivity to remote entities (users and sites).

The IPsec framework successfully deals not only with the tasks associated with ensuring confidentiality, integrity, authentication of origin, and secure key distribution but also accomplishes this in a way that is totally independent of the transport used by the application (UDP, TCP, or raw IP). Further, IPsec's architecture enables the replacement of one or more of its components (such as the hash functions and cryptographic algorithms) to keep up with hardware evolution and what it means in terms of feasibility of brute force attacks.

Basic knowledge of IPsec operations is assumed here. Even though this section is not meant to explore IPsec configuration details, some configuration information has been included whenever necessary to facilitate the discussion.

The two basic interactions covered follow:

- Issues that arise when firewalls are in the path of VPN tunnels: How do NAT settings affect the tunnels? What rules need to be created in this firewall to guarantee that the tunnel is successfully established?
- **The behavior of a stateful firewall that also terminates the VPN tunnels:** What can be done to provide granular inspection for traffic inside the tunnel, either by the VPN terminator itself or by a dedicated firewall connected to it?

Before examining these interactions, it is important to mention the overall positioning of Cisco VPN platforms:

- Cisco IOS routers are optimized for site-to-site IPsec deployments, particularly the large and very large ones. The breadth of advanced VPN models like Dynamic Multipoint VPN (DMVPN) and Group Encrypted Transport (GET VPN) allied with support for enhanced QoS and routing functionality (just to name a few) make the Cisco routers a natural choice for this type of environment.
- Although ASA appliances do support site-to-site, they are the most flexible platform for Remote Access VPN implementations. They can simultaneously terminate IPsec and SSL-based remote access connections with an always-evolving feature set and a considerable level of scalability.

Classic IPsec Site-to-Site for IOS

Figure 17-19 shows a reference topology and the relevant configurations for a classic IOS site-to-site deployment. The term classic here means that no special logical interface is in use and none of the advanced models (DMVPN, GET VPN, and so on) is in place.

Example 17-3 provides additional information about this setup.

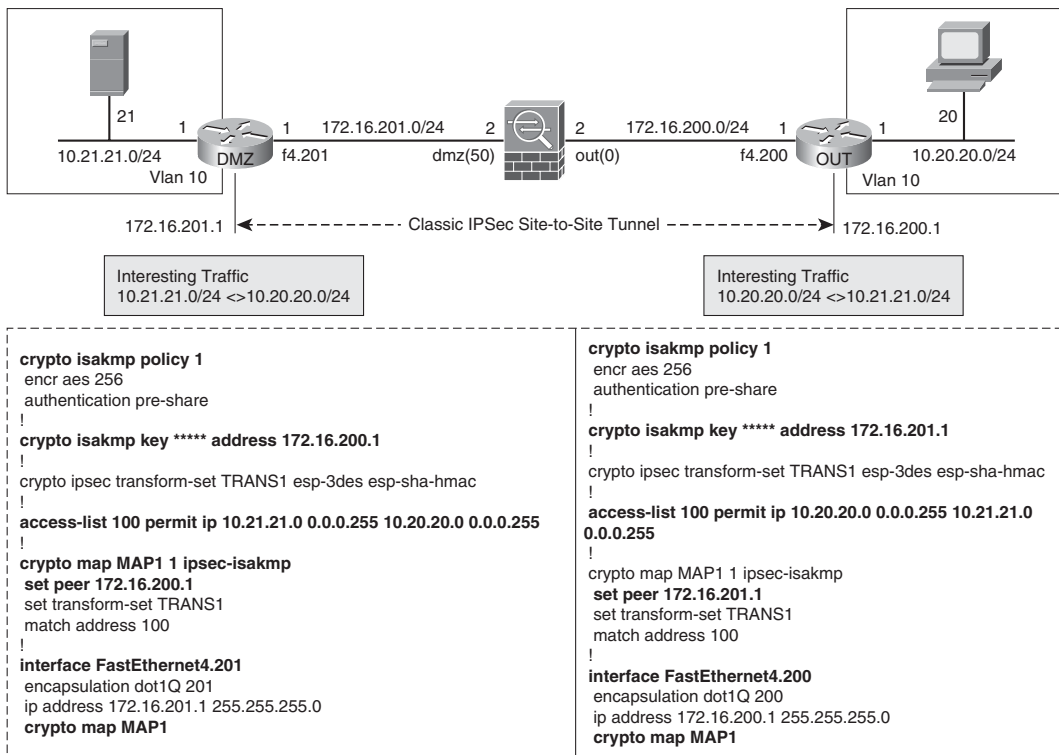


Figure 17-19 Reference Topology for Classic IPsec Site-to-Site Scenario

Example 17-3 Basic Information About the Classic IPsec Site-to-Site Scenario

```

DMZ# show crypto session
Crypto session current status
Interface: FastEthernet4.201
Session status: UP-ACTIVE
Peer: 172.16.200.1 port 500
IKE SA: local 172.16.201.1/500 remote 172.16.200.1/500 Active
IPSEC FLOW: permit ip 10.21.21.0/255.255.255.0 10.20.20.0/255.255.255.0
Active SAs: 2, origin: crypto map
!
DMZ# show crypto isakmp sa detail
Codes: C - IKE configuration mode, D - Dead Peer Detection
K - Keepalives, N - NAT-traversal
T - cTCP encapsulation, X - IKE Extended Authentication
psk - Preshared key, rsig - RSA signature
renc - RSA encryption
IPv4 Crypto ISAKMP SA

```

C-id	Local	Remote	I-VRF	Status	Encr	Hash	Auth	DH
Lifetime Cap.								
2001	172.16.201.1	172.16.200.1		ACTIVE	aes	sha	psk	1 21:39:02
Engine-id:Conn-id = SW:1								

Example 17-4 documents two possible ways to add filtering rules for the post-decryption VPN traffic (with no need of an external firewall). The configurations refer to the router named DMZ in the arrangement of Figure 17-19:

- Using an interface ACL applied to the internal interface (in this case, this corresponds to interface VLAN 10).
- By adding the **set ip access-group** command under the **crypto map** definition. Another **ip access-group** reference could have been added in the *out* direction. It is critical that you do not confuse this filtering-purpose ACL with that used for the definition of the traffic that triggers the tunnel setup (for example, ACL 100 in the context of Figure 17-19).

Example 17-4 Interface ACL in Action in the Classic IPsec Scenario

```
! Applying ACL to interface Vlan 10 in the 'out' direction
access-list 110 permit icmp 10.20.20.0 0.0.0.255 10.21.21.0 0.0.0.255
access-list 110 deny udp any any range 0 65535 log-input
access-list 110 deny tcp any any range 0 65535 log-input
!
interface Vlan10
  ip access-group 110 out
!
! ACL 110 comes into the scene - blocking Syslog and FTP connection attempts
%SEC-6-IPACCESSLOGP: list 110 denied udp 10.20.20.1(53171) (FastEthernet4.201 ) ->
10.21.21.21(514), 1 packet
%SEC-6-IPACCESSLOGP: list 110 denied tcp 10.20.20.20(1117) (FastEthernet4.201 ) ->
10.21.21.21(21), 1 packet
!
! Another way of filtering packets after decryption

crypto map MAP1 1 ipsec-isakmp
  set peer 172.16.200.1
  set ip access-group 131 in
  set transform-set TRANS1
  match address 100
!
! ACL 131 comes into play
%SEC-6-IPACCESSLOGP: list 131 denied tcp 10.20.20.20(1351) (FastEthernet4.201 ) ->
10.21.21.21(21), 1 packet
```

```

!
DMZ# show access-1 131
Extended IP access list 131
 10 permit udp 10.20.20.0 0.0.0.255 host 10.21.21.21 eq syslog (3 matches)
 20 permit icmp 10.20.20.0 0.0.0.255 10.21.21.0 0.0.0.255 echo (4 matches)
 30 permit icmp 10.20.20.0 0.0.0.255 10.21.21.0 0.0.0.255 echo-reply
 40 deny udp any any range 0 65535 log-input
 50 deny tcp any any range 0 65535 log-input (3 matches)

```

Although the filters just presented are already useful, they are stateless in nature. As extensively covered throughout the book, a stateful option not only provides better security but is also easier to manage. Figure 17-20 shows an example of the interaction of the Zone Policy Firewall with a Classic IPSec tunnel (both features enabled in the same router). Only a sample inbound ZFW policy (from the VPN to the internal LAN) is demonstrated. The construction of the LAN-to-VPN policy is left as an exercise for you.

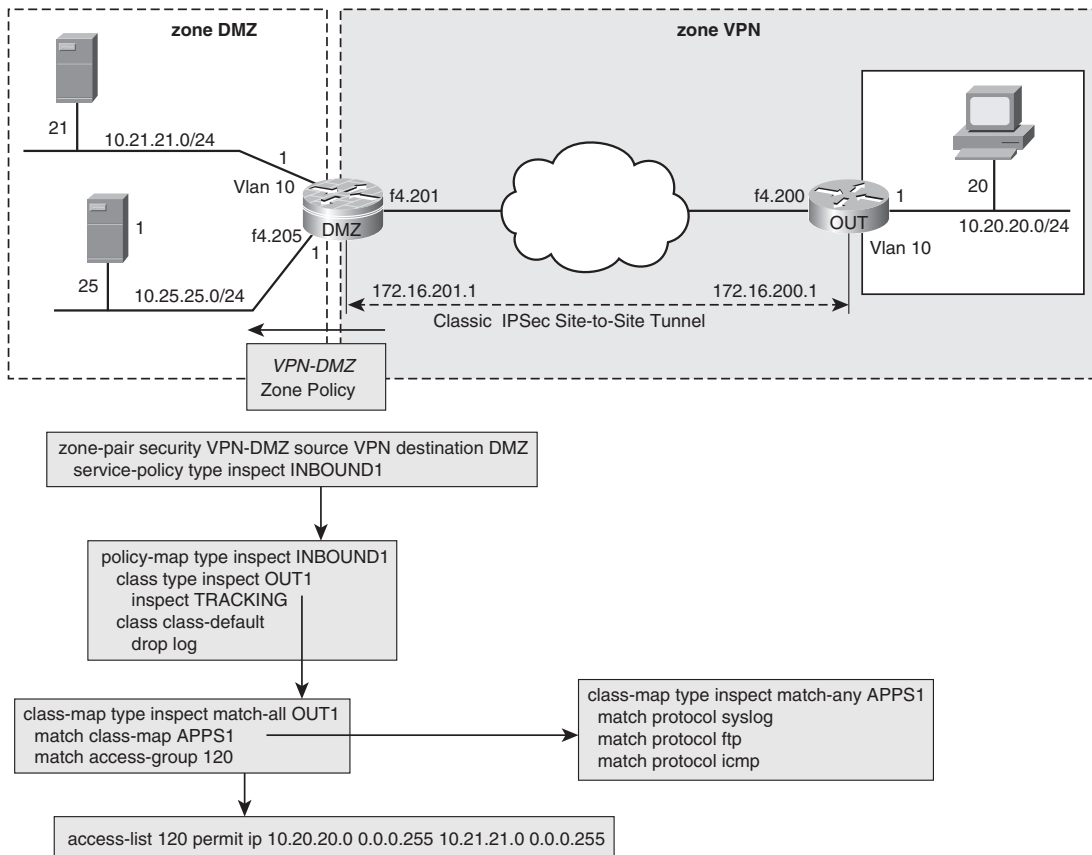


Figure 17-20 Classic IPSec Site-to-Site Scenario and the Zone Policy Firewall

Examples 17-5 and 17-6 illustrate the ZFW in action for the VPN scenario represented in Figure 17-20:

- Example 17-5 demonstrates two situations (ICMP and FTP flows) in which the traffic is permitted by the ZFW policy. The example also employs Netflow to provide a view of clear-text and encrypted packets (protocol type 50 = 0x32, corresponding to ESP).
- Example 17-6 concentrates on denied traffic. First, an attempt to reach the 10.25.25.0/24 subnet is blocked. Later, an arriving packet that matches the **crypto map** definitions in Figure 17-19 (10.20.20.0/24 > 10.21.21.0/24), should have been encrypted but was not. This type of consistency check is performed even before the ZFW comes to the scene.

Example 17-5 ZFW in Action in the Classic IPSec Scenario (1)

```
! Ping from 10.20.20.20 to 10.21.21.21
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(VPN-DMZ:OUT1):Start
icmp session: initiator (10.20.20.20:0) — responder
(10.21.21.21:0)
!
DMZ# show ip cache flow | begin Src

```

SrcIf	SrcIPAddress	DstIf	DstIPAddress	Pr	SrcP	DstP	Pkts
Fa4.201	172.16.200.1	V110	172.16.201.1	32	3C47	EA1D	4
V110	10.21.21.21	Fa4.201	10.20.20.20	01	0000	0000	4
V110	172.16.201.1	Fa4.201*	172.16.200.1	32	1EEA	2756	4
Fa4.201	10.20.20.20	V110*	10.21.21.21	01	0000	0800	4

```
!
! Sample FTP session from client 10.20.20.20 to Server 10.21.21.21
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(VPN-DMZ:OUT1):Start ftp
session: initiator (10.20.20.20:1256) — responder
(10.21.21.21:21)
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(VPN-DMZ:OUT1):Start
ftp-data session: initiator (10.21.21.21:20) — responder
(10.20.20.20:1260)
!
DMZ# show ip cache flow | begin Src

```

SrcIf	SrcIPAddress	DstIf	DstIPAddress	Pr	SrcP	DstP	Pkts
V110	10.21.21.21	Null	10.20.20.20	06	0014	04EC	201
V110	10.21.21.21	Fa4.201	10.20.20.20	06	0015	04E8	11
Fa4.201	172.16.200.1	V110	172.16.201.1	32	3C47	EA1D	395
Fa4.201	10.20.20.20	V110*	10.21.21.21	06	04EC	0014	70
Fa4.201	10.20.20.20	V110*	10.21.21.21	06	04E8	0015	10
V110	172.16.201.1	Fa4.201*	172.16.200.1	32	1EEA	2756	906

```
!
%FW-6-SESS_AUDIT_TRAIL: (target:class)-(VPN-DMZ:OUT1):Stop ftp-data
```

```

session: initiator (10.21.21.21:20) sent 271777 bytes — responder
(10.20.20.20:1260) sent 0 bytes
!
! After some FTP sessions, there are more encrypted packets than decrypted ones

DMZ# show crypto session detail | begin IPSEC
IPSEC FLOW: permit ip 10.21.21.0/255.255.255.0 10.20.20.0/255.255.255.0
    Active SAs: 2, origin: crypto map
    Inbound:  #pkts dec'd 1101 drop 0 life (KB/Sec) 4471035/1411
    Outbound: #pkts enc'd 2577 drop 0 life (KB/Sec) 4467604/1411

```

Example 17-6 ZFW in Action in the Classic IPsec Scenario (2)

```

! Sample packet dropped by the ZFW (no VPN-side traffic can reach 10.25.25.0/24)
%FW-6-LOG_SUMMARY: 1 packet were dropped from 10.20.20.20:8 =>
10.25.25.25:0 (target:class)-(VPN-DMZ:class-default)
!
! Arriving packet (from 10.20.20.20 to 10.21.21.21) should have been encrypted...
%CRYPTO-4-RECD_PKT_NOT_IPSEC: Rec'd packet not an IPSEC packet.
(ip) vrf/dest_addr= /10.21.21.21, src_addr= 10.20.20.20, prot= 1

```

IPsec Site-to-Site Using a Virtual Tunnel Interface (VTI)

The use of IPsec VTIs is aimed at simplifying the configuration process that characterizes the Classic IPsec model and, at the same time, reducing the overhead associated with the IPsec/GRE option. The existence of this type of logical interface at the tunnel endpoint not only shows the routing benefits that are typical of GRE but also facilitates the use of interface-related features such as ACLs, QoS, and multicast forwarding. And as demonstrated later in this section, it does help with the construction of ZFW policies.

As a reference, the VTI packet flow is summarized in Figure 17-21.

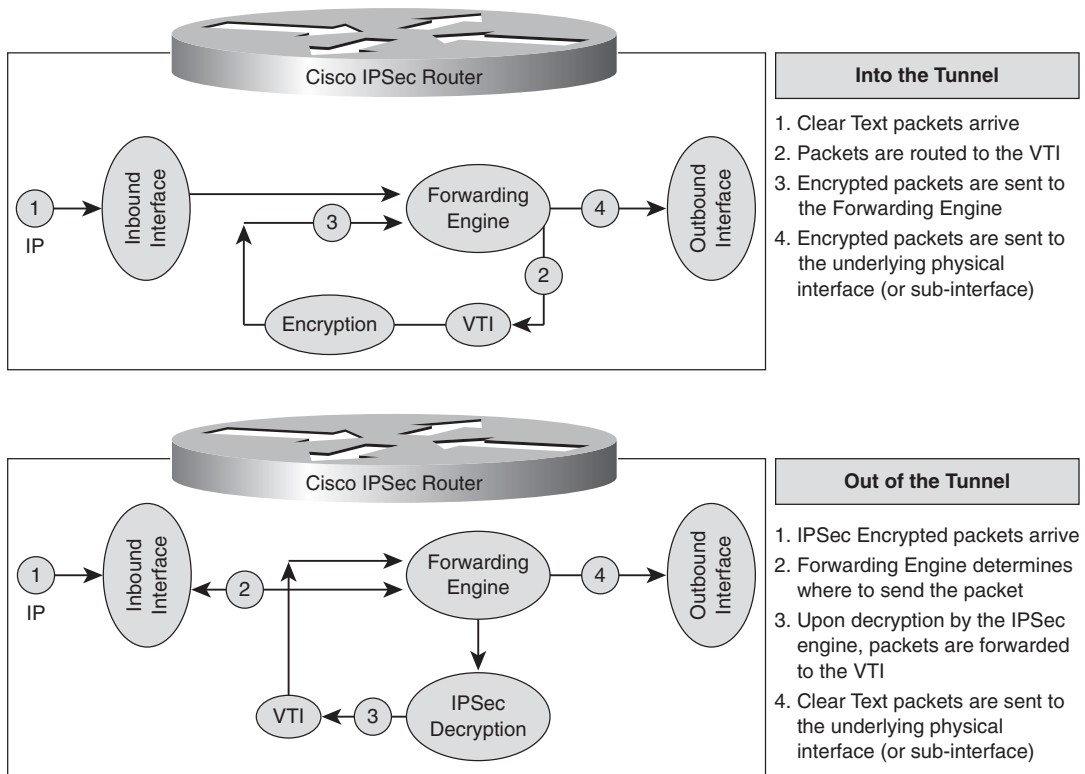


Figure 17-21 Packet Flow for VTI Interfaces

Figure 17-22 portrays a scenario in which a VTI-based IPsec tunnel is built between routers named DMZ and OUT. Example 17-7 brings more details about this environment:

- As opposed to the Classic IPsec model, there is no **crypto map** anymore. The so-called interesting traffic for IPsec is naturally selected when the VTI is used for routing. As shown in Figure 17-21 for the *into the tunnel* packet flow, IPsec-related encryption automatically follows for those packets routed via the VTI.
- It is instructive to compare the ‘IPSEC FLOW’ information unveiled by the **show crypto session** command with that associated with the Classic IPsec model (refer to Example 17-3).
- Like other logical interfaces, the VTI has its own IP address and, from a routing point of view, behaves as a point-to-point link. The routing and forwarding details (from the perspective of router DMZ) are shown for subnet 10.20.20.0/24.

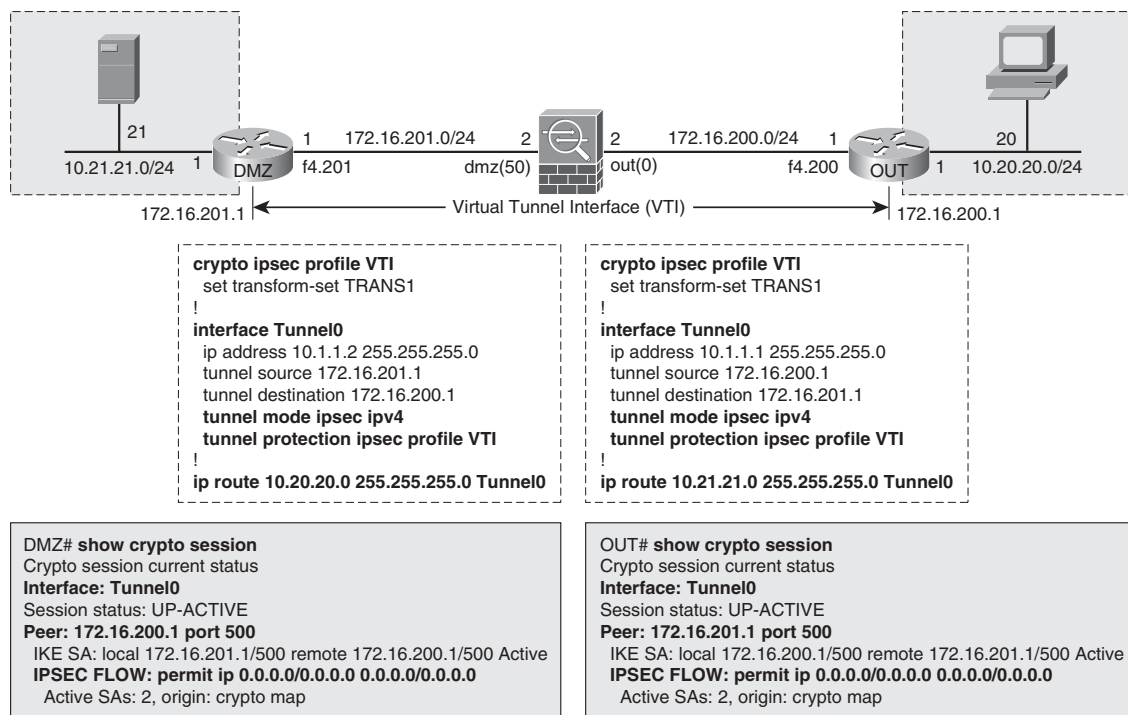


Figure 17-22 Reference Topology for VTI analysis

Example 17-7 More Details About the VTI Model

```
! IPSEC and IKE information
DMZ# show crypto session detail | begin Interface
Interface: Tunnel0
Uptime: 00:12:12
Session status: UP-ACTIVE
Peer: 172.16.200.1 port 500 fvrf: (none) ivrf: (none)
Phase1_id: 172.16.200.1
Desc: (none)
IKE SA: local 172.16.201.1/500 remote 172.16.200.1/500 Active
Capabilities:(none) connid:2006 lifetime:23:47:47
IPSEC FLOW: permit ip 0.0.0.0/0.0.0.0 0.0.0.0/0.0.0.0
Active SAs: 2, origin: crypto map
Inbound: #pkts dec'd 2 drop 0 life (KB/Sec) 4459330/2867
Outbound: #pkts enc'd 2 drop 0 life (KB/Sec) 4459330/2867
!
! Details about the tunnel interface
DMZ# show interface Tunnel 0 | include Tunnel
```

```

Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  Tunnel source 172.16.201.1, destination 172.16.200.1
Tunnel protocol/transport IPSEC/IP
  Tunnel TTL 255
Tunnel transport MTU 1443 bytes
  Tunnel transmit bandwidth 8000 (kbps)
  Tunnel receive bandwidth 8000 (kbps)
Tunnel protection via IPsec (profile "VTI")
!
! Forwarding details (remote destination 10.20.20.20, behind 'OUT')

DMZ# show ip cef 10.20.20.20
10.20.20.0/24
  attached to Tunnel0
!
DMZ# show ip cef exact-route 10.21.21.21 10.20.20.20
10.21.21.21 -> 10.20.20.20 => IP adj out of FastEthernet4.201, addr 172.16.201.2

```

Example 17-8 was built by promoting a slight modification to Figure 17-20: The VPN-DMZ **zone-pair** was moved from interface f4.201 to Tunnel 0 (VTI). This small change makes life much easier in terms of ZFW configuration workload because the simple existence of a logical tunnel interface clearly delimits traffic bound to the VPN. The underlying physical interface (or subinterface, as in Figure 17-22) can then be part of another security zone and be assigned new policies (for instance, governing what is accessible from the outside or establishing what are the acceptable VTI endpoints).

Example 17-8 also shows the flows created for a simple ICMP connection that has been routed through (and protected by) the VTI. Comparing the behavior herein documented with that of Example 17-5 (Classic IPsec) might be an interesting exercise.

Example 17-8 VTI and ZFW

```

! ICMP from 10.20.20.20 to 10.21.21.21
%FW-6-SESS_AUDIT_TRAIL_START: (target:class)-(VPN-DMZ:OUT1):Start
icmp session:      initiator (10.20.20.20:0) — responder
(10.21.21.21:0)
!
DMZ# show ip cache flow | begin Src

```

SrcIf	SrcIPAddress	DstIf	DstIPAddress	Pr	SrcP	DstP	Pkts
Local	172.16.201.1	Fa4.201*	172.16.200.1	32	0771	D655	3
Vl10	10.21.21.21	Tu0	10.20.20.20	01	0000	0000	3
Tu0	10.20.20.20	Vl10*	10.21.21.21	01	0000	0800	3
Fa4.201	172.16.200.1	Vl10	172.16.201.1	32	A7DD	6A16	3

IPsec Site-to-Site Using a GRE Tunnel

In terms of routing and ability to handle interface-related features, the GRE model is similar to VTI. The main distinction resides on the extra layer of encapsulation required by GRE. This layer represents an additional overhead but, on the other hand, does not limit GRE to the transport of IP Packets. Being *generic* in its capability to carry any routed protocol stack is one of the key design goals of GRE (RFCs 1701 and 1702).

Figure 17-23 shows, for the sake of comparison, a simple network configured for GRE over IPsec. The figure also includes the GRE header scheme for ESP-based IPsec operating in Tunnel Mode. Example 17-9 provides additional details about GRE/IPsec:

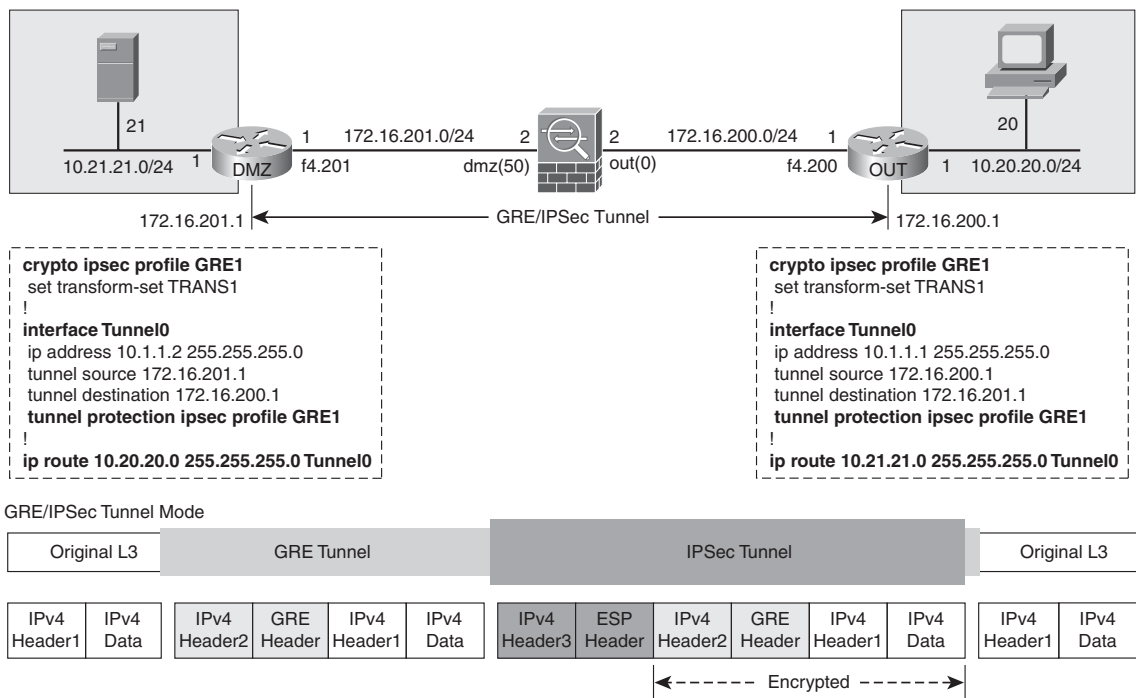


Figure 17-23 Reference Topology for IPsec/GRE Analysis

- GRE/IP is the default mode of operation for a tunnel interface in IOS. This was seen with the analysis of IPv6 tunneling mechanisms.
- The 'IPSEC FLOW' contained in the **show crypto session** command now displays a **permit 47** statement, in which 47 corresponds to the IPv4 protocol number assigned to GRE. (Compare with VTI and Classic IPsec.)
- The example also reveals some parameters pertaining to the tunnel interface itself. The MTU value might look strange at first sight. How come GRE, having more encapsulations than VTI, has a larger MTU? (Refer to Example 17-7.) This is not the case. The MTU shown for the GRE tunnel subtracts only 24 bytes (GRE Header + IPv4

Header2) from the original IP MTU (1500 bytes). This happens because, as opposed to VTI, GRE is not dedicated to IPsec.

Example 17-9 *More Details About the IPsec/GRE Model*

```
DMZ# show crypto session detail | begin Interface
Interface: Tunnel0
Uptime: 00:16:38
Session status: UP-ACTIVE
Peer: 172.16.200.1 port 500 fvrf: (none) ivrf: (none)
    Phase1_id: 172.16.200.1
    Desc: (none)
IKE SA: local 172.16.201.1/500 remote 172.16.200.1/500 Active
    Capabilities:(none) connid:2009 lifetime:23:43:21
IPSEC FLOW: permit 47 host 172.16.201.1 host 172.16.200.1
    Active SAs: 2, origin: crypto map
    Inbound:  #pkts dec'ed 3 drop 0 life (KB/Sec) 4569744/2601
    Outbound: #pkts enc'ed 3 drop 0 life (KB/Sec) 4569744/2601
!
DMZ# show interface tunnel 0 | include Tunnel
Tunnel0 is up, line protocol is up
    Hardware is Tunnel
    Tunnel source 172.16.201.1, destination 172.16.200.1
Tunnel protocol/transport GRE/IP
    Tunnel TTL 255
Tunnel transport MTU 1476 bytes
    Tunnel transmit bandwidth 8000 (kbps)
    Tunnel receive bandwidth 8000 (kbps)
Tunnel protection via IPsec (profile "GRE1")
```

Time for Work At this point, it is instructive to compare the IPsec tunnels with those as an IPv4 to IPv6 integration method. What are the similarities in terms of encapsulation, traffic forwarding, and firewall placement (particularly, ZFW policy definition)?

NAT in the Middle of an IPsec Tunnel

The topology depicted in Figure 17-19 is reused for the analysis of the NAT-IPsec interaction. The significant configuration change in the new scenario resides in that ASA is now performing static NAT for the external address of the DMZ router so that OUT sees the IPsec packets originated by 172.16.201.1 as if they had been sourced from 172.16.200.201.

Example 17-10 refers to Figure 17-19 and shows some details about the specific situation in which all the NAT traversal mechanisms have been disabled. Nevertheless, given that

the IPsec SA uses ESP (and not AH) and that only static NAT is in place, the tunnel is successfully built. Also notice the following:

- ASA logs the ISAKMP connection setup (UDP/500) but the ESP connection is shown only in the `conn` table.
- The OUT router has the impression of being in touch with 172.16.200.201 and not with the real IP, 172.16.201.1.

Example 17-10 *Static NAT in the Path of an IPsec Tunnel*

```

! Disabling NAT Transparency mechanisms on the IOS Routers ('DMZ' and 'OUT')
no crypto ipsec nat-transparency udp-encaps
!
! What ASA sees (ISAKMP connection setup is logged. ESP appears in
the conn table)
%ASA-6-302015: Built inbound UDP connection 150 for
out:172.16.200.1/500 (172.16.200.1/500) to dmz:172.16.201.1/500
(172.16.200.201/500)
!
ASA# show conn
4 in use, 6 most used
ESP out 172.16.200.1 dmz 172.16.201.1, idle 0:00:02, bytes 756
UDP out 172.16.200.1:500 dmz 172.16.201.1:500, idle 0:00:15, bytes 1192, flags -
ESP out 172.16.200.1 dmz 172.16.201.1, idle 0:00:02, bytes 756
!
OUT# show crypto isakmp sa detail
Codes: C - IKE configuration mode, D - Dead Peer Detection
       K - Keepalives, N - NAT-traversal
       T - cTCP encapsulation, X - IKE Extended Authentication
       psk - Preshared key, rsig - RSA signature
       renc - RSA encryption
IPv4 Crypto ISAKMP SA
C-id Local Remote I-VRF Status Encr Hash Auth DH
Lifetime Cap.
2011 172.16.200.1 172.16.200.201 ACTIVE aes sha psk 1 23:57:42
Engine-id:Conn-id = SW:11
    
```

Example 17-11 differs from Example 17-10 in a minor detail; the default IOS NAT transparency method (UDP encapsulation) has been reconfigured:

- Now ASA logs not only the ISAKMP connection, but also that associated with the IOS NAT transparency resource (UDP/4500). The `show conn` does not display any ESP entry because the packets originally protected via ESP are further encapsulated using UDP/4500.

- The use of the NAT- resource is also revealed by means of the **show crypto isakmp sa detail** command.
- This mechanism enables the IPsec tunnel to be established even across dynamic PAT environments.

Example 17-11 IOS NAT-Transparency Using UDP Encapsulation

```

! Connection setup from ASA standpoint
%ASA-6-302015: Built inbound UDP connection 154 for
out:172.16.200.1/500 (172.16.200.1/500) to dmz:172.16.201.1/500
(172.16.200.201/500)
%ASA-6-302015: Built inbound UDP connection 155 for
out:172.16.200.1/4500 (172.16.200.1/4500) to dmz:172.16.201.1/4500
(172.16.200.201/4500)
!
ASA# show conn
3 in use, 6 most used
UDP out 172.16.200.1:4500 dmz 172.16.201.1:4500, idle 0:00:00, bytes 2264, flags -
UDP out 172.16.200.1:500 dmz 172.16.201.1:500, idle 0:00:13, bytes 800, flags -

OUT# show crypto isakmp sa detail
Codes: C - IKE configuration mode, D - Dead Peer Detection
       K - Keepalives, N - NAT-traversal
       T - cTCP encapsulation, X - IKE Extended Authentication
       psk - Preshared key, rsig - RSA signature
       renc - RSA encryption
IPv4 Crypto ISAKMP SA
C-id Local Remote I-VRF Status Encr Hash Auth DH
Lifetime Cap.
2012 172.16.200.1 172.16.200.201 ACTIVE aes sha psk 1
23:58:35 N
Engine-id:Conn-id = SW:12

```

Time for Review You probably noticed that in most of the scenarios studied so far, there is an ASA appliance in the path of the IOS-based tunnel. This is done intentionally to give you the chance to study this type of interaction. By revisiting the examples, you are encouraged to write the appropriate Access Control Entries that are needed to allow tunnel initiation from OUT to DMZ, in each scenario: Classic IPsec, VTI, GRE, and NAT Transparency. For this task, consider the **security-levels** in the figures. (**permit ip any any** or similar rules are not acceptable answers.)

Example 17-12 simply shows NAT-T verification in action for an ASA appliance, which corresponds to the default behavior. ASA also supports the IPsec over TCP and IPsec over UDP NAT transparency mechanisms.

Example 17-12 *ASA and NAT Transparency*

```

ASA# show run all crypto isakmp
crypto isakmp identity auto
crypto isakmp nat-traversal 20
!
%ASA-5-713041: IP = 172.16.202.2, IKE Initiator: New Phase 1, Intf
dmz1, IKE Peer 172.16.202.2 local Proxy Address 10.43.43.0, remote
Proxy Address 10.42.42.0,
Crypto map (MAP1)
%ASA-6-302015: Built outbound UDP connection 127 for
vpn1:172.16.202.2/500 (172.16.202.2/500) to identity:172.16.203.2/500
(172.16.203.2/500)
%ASA-6-713172: Group = 172.16.202.2, IP = 172.16.202.2, Automatic NAT
Detection Status:
Remote end is NOT behind a NAT device      This end is NOT behind a
NAT device
!
ASA# show run all crypto isakmp
crypto isakmp identity auto
crypto isakmp nat-traversal 20

```

Post-Decryption Filtering in ASA

Figure 17-24 shows a classic site-to-site IPSec tunnel between an IOS router and an ASA appliance. Basic information for this setup has been included as a reference.

The default ASA behavior is to permit any decrypted packet that derives from an IPSec connection. This means that arriving packets matching the IPSec interesting traffic definition (in this case, established by ACL 102, under **crypto map** MAP1) will be allowed through. This behavior is governed by the global command **sysopt connection permit-vpn**, which is enabled by default. This command takes precedence over interface ACLs aimed at filtering decrypted packets.

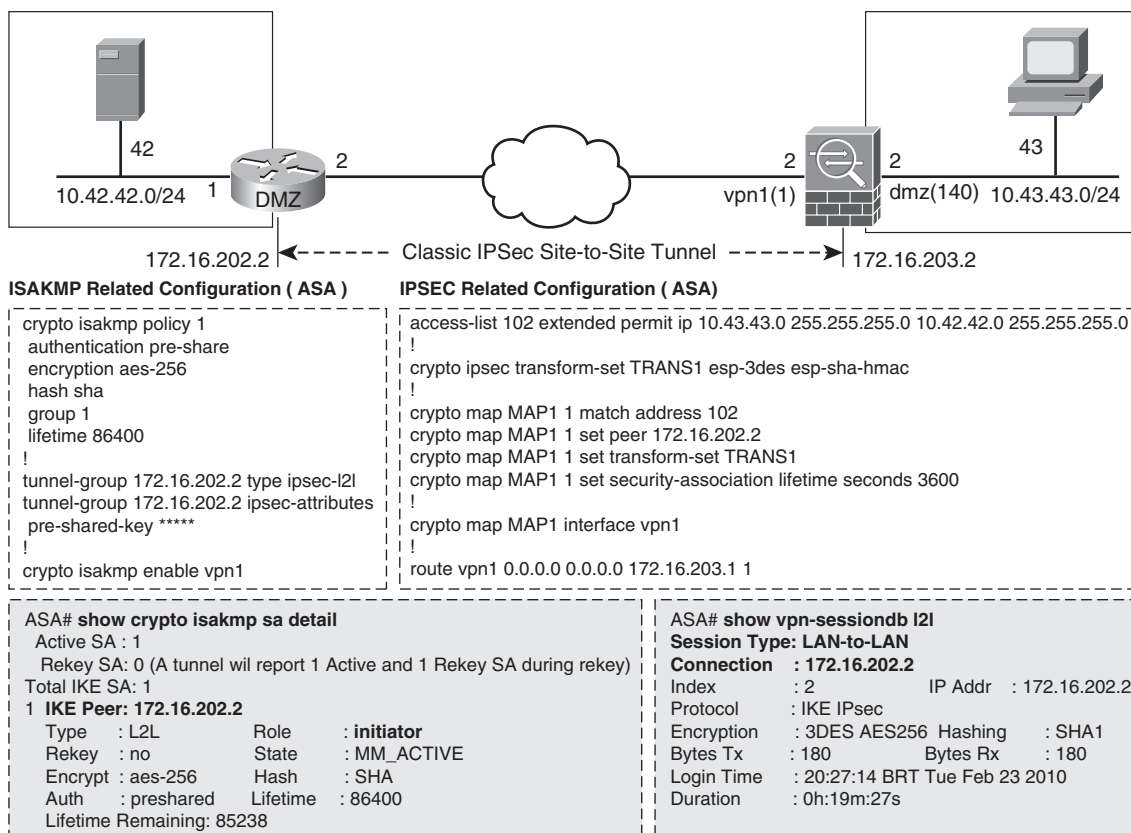


Figure 17-24 Reference Topology for Classic IPsec on ASA

In summary, if the goal is to enable granular filtering for the traffic inside the tunnels, two basic options can be employed:

- When the ASA appliance shown in Figure 17-24 is dedicated for VPN termination, you can use another firewall (connected to the 'dmz1' interface) to filter the clear text packets.
- You can use the **no sysopt connection permit-vpn** command to render effective the interface ACLs you might configure.

Example 17-13 *ASA Behavior for Decrypted Packets*

```

! Sample ping session through ASA
%ASA-6-302020: Built outbound ICMP connection for faddr 10.42.42.42/0 gaddr
10.43.43.43/512 laddr 10.43.43.43/512
%ASA-6-302020: Built inbound ICMP connection for faddr 10.42.42.42/0 gaddr
10.43.43.43/512 laddr 10.43.43.43/512
!
ASA# show conn all
5 in use, 7 most used
ICMP vpn1 10.42.42.42:0 dmz1 10.43.43.43:512, idle 0:00:01, bytes 128
ESP vpn1 172.16.202.2 NP Identity Ifc 172.16.203.2, idle 0:00:01, bytes 336
UDP vpn1 172.16.202.2:500 NP Identity Ifc 172.16.203.2:500, idle 0:00:07, bytes
27588, flags -
ICMP vpn1 10.42.42.42:0 dmz1 10.43.43.43:512, idle 0:00:01, bytes 128
ESP vpn1 172.16.202.2 NP Identity Ifc 172.16.203.2, idle 0:00:01, bytes 336
!
! Modifying the 'default permit' behavior for VPN traffic

no sysopt connection permit-vpn

```

Tip The `sysopt connection permit-vpn` command is enabled via the ASDM path **Configuration > Site-to-Site VPN > Advanced > System Options**, by checking the following box: *Enable Inbound IPSEC Sessions to Bypass Interface access-lists. Group policy and Per-User Authorization access-lists Still Apply To The Traffic.*

Firewalls and SSL VPNs

Even though IPsec technology successfully deals with many security-related tasks, SSL VPN has conquered some important ground on real life implementation cases. *Why was it ever made possible?*

The main motivation for developing another VPN solution focused on remote access had to do with that SSL VPN does not require a dedicated client. In its original form, SSL VPN was even deemed a synonym of *clientless access* (because the web browser is considered a true type of universal client available on any networked machine). Precisely this capability of providing secure remote access, even for those stations that were not managed by corporate IT, sounded appealing.

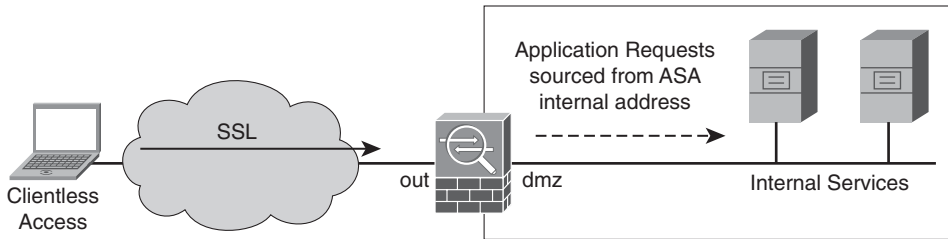
After becoming acquainted with this new paradigm for remote access, many customers started to request a client-based SSL-VPN option that could eventually substitute IPsec. The Cisco response to this demand is the *Cisco AnyConnect VPN Client*, a software solution initially focused on SSL VPN functionality but reflecting in the development roadmap a justification for the term *any* in its name. (Features such as IPsec and 802.1x support were about to be delivered by the time this chapter was finalized.)

This section is ASA-centric and assumes a basic understanding of SSL-VPN technology. The main types of interactions covered follow:

- **Global filtering functionalities:** Either provided by an external firewall or on the ASA appliance that is in charge of terminating the SSL connections.
- **Group-level features:** The resources that enable better differentiation among users requesting the remote access services.

Clientless Access

The clientless SSL VPN model can be deemed an advanced reverse proxy solution, whose basic operations are summarized in the following list and later shown in Figure 17-25:



Web VPN Portal (Service Selection)

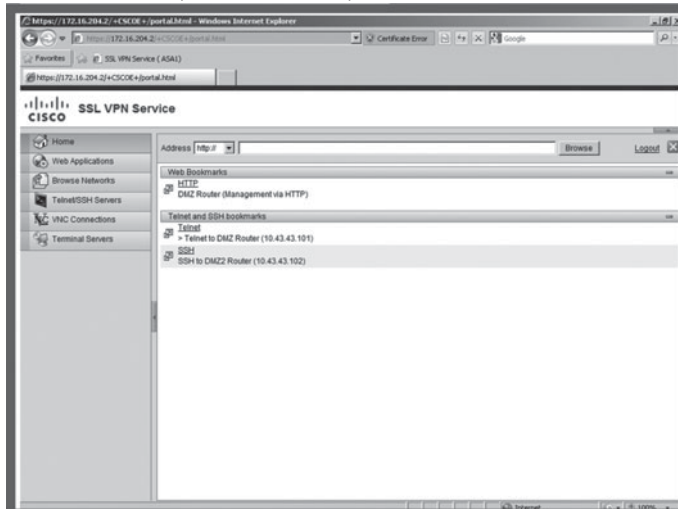


Figure 17-25 Basic Model for Clientless Access

- A browser connects via SSL to the *external* interface of ASA.
- Access is granted after successful authentication (and possibly authorization) of the user. The authentication protocols normally used for VPN are RADIUS and Lightweight Directory Access Protocol (LDAP).
- A portal showing a customized URL list, corresponding to the available applications, is presented to the user.
- Upon selection of the intended application by the user, ASA appliance initiates a new connection (using the address of one its *internal* interfaces) to the server hosting that specific application.

If you use a dedicated firewall (after SSL-VPN termination) to segment the internal networks, be aware that in the clientless model, the source address seen by the servers always belongs to the set of ASA internal addresses.

Figure 17-26 represents a sample clientless topology and assembles the commands used for a basic setup. As stated before, the purpose here is not to present configuration models for SSL-VPN but, rather, to analyze some of the filtering functionalities available. Additional information pertaining to this environment is shown in Example 17-14:

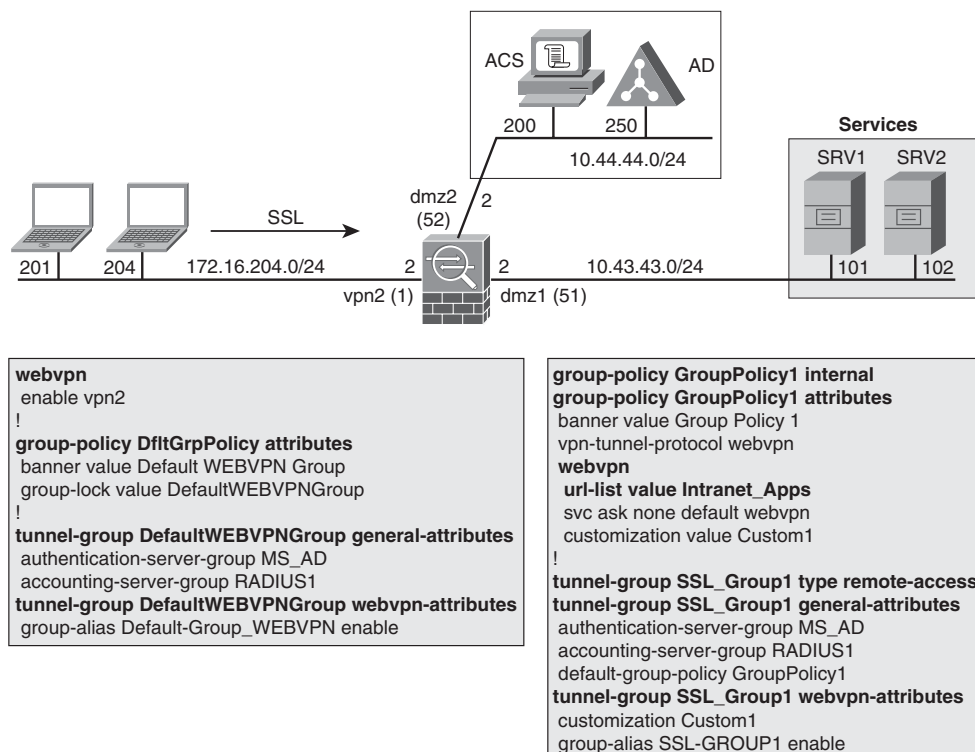


Figure 17-26 Reference Topology for Clientless Analysis

- The external user named *user1* accesses the SSL-VPN service from the address 172.16.204.204.
- The user is successfully authenticated in the LDAP server at 10.44.44.250.
- Even though the LDAP server has informed the user-specific group policy for *user1* (who is part of group *SSL_Group1*), the default group policy (*DfltGrpPolicy*) is selected. As a result, the user inherited the Connection Profile (Tunnel Group) named *DefaultWEBVPNGroup*. This is the normal behavior for AAA access control when the group list is not presented to the user (during the login process). This option is shown in a later example.
- The user chooses the URL <http://10.43.43.101> (from the portal) and ASA initiates a connection to the web server (sourced from address 10.43.43.2).

Displaying the bookmark list in the portal does not prevent the user from visiting other sites (including those that do not comply with your company's acceptable use policy). One way to obtain this restrictive effect is to configure a special category of ACL that uses the **webtype** keyword (often referred to as a Web ACL). This is documented in Example 17-15. If you forget to remove a URL (that should be blocked) from the portal, it is grayed out and not selectable. As such, the recommendation is to present only in the portal those URLs actually permitted. This avoids user confusion when navigating (and the undesirable support calls).

At this point, it is important to highlight two points:

- Web ACLs are used for restricting TCP traffic, which is the normal transport protocol for clientless scenarios.
- Regular interface ACLs are not suited for restricting access in the clientless model.

Example 17-14 Basic Information About Clientless Access

```
! Authentication process (using an LDAP server)
%ASA-6-302013: Built outbound TCP connection 254 for
dmz2:10.44.44.250/389 (10.44.44.250/389) to identity:10.44.44.2/57300
(10.44.44.2/57300)
%ASA-6-113004: AAA user authentication Successful : server =
10.44.44.250 : user = user1
%ASA-6-113003: AAA group policy for user user1 is being set to
GroupPolicy1
%ASA-6-113011: AAA retrieved user specific group policy (GroupPolicy1) for
user = user1
%ASA-6-113009: AAA retrieved default group policy (DfltGrpPolicy) for user = user1
%ASA-6-113008: AAA transaction status ACCEPT : user = user1
%ASA-6-734001: DAP: User user1, Addr 172.16.204.204, Connection
Clientless: The following DAP records were selected for this
connection: DfltAccessPolicy
```

```
%ASA-6-716001: Group <GroupPolicy1> User <user1> IP <172.16.204.204>
WebVPN session started.
%ASA-6-716038: Group <GroupPolicy1> User <user1> IP <172.16.204.204>
Authentication: successful, Session Type: WebVPN.
!
! Information about the established connection
ASA# show vpn-sessiondb webvpn
Session Type: WebVPN
Username      : user1                      Index      : 11
Public IP      : 172.16.204.204
Protocol       : Clientless
License        : SSL VPN
Encryption     : RC4                      Hashing     : SHA1
Bytes Tx       : 61693                    Bytes Rx    : 13718
Group Policy  : GroupPolicy1              Tunnel Group : DefaultWEBVPNGroup
Login Time     : 08:23:07 UTC Fri Mar 26 2010
Duration       : 0h:00m:55s
Inactivity     : 0h:00m:00s
NAC Result     : Unknown
VLAN Mapping   : N/A                      VLAN        : none
!
! User user1 selects HTTP URL displayed in the portal

%ASA-6-716003: Group <GroupPolicy1> User <user1> IP <172.16.204.204>
WebVPN access GRANTED: http://10.43.43.101//
%ASA-6-302013: Built outbound TCP connection 264 for
dmz1:10.43.43.101/80 (10.43.43.101/80) to identity:10.43.43.2/37802
(10.43.43.2/37802)
```

Example 17-15 Controlling Clientless Access with a Web ACL

```
! Defining a webtype ACL and attaching it to a specific Group Policy
access-list WEB-ACL1 webtype permit url telnet://10.43.43.101 log default
!
group-policy GroupPolicy1 attributes
    webvpn
        filter value WEB-ACL1
!
! The denied entries logged by ASA appear grayed out in the Web VPN Portal
%ASA-6-716003: Group <GroupPolicy1> User <user1> IP <172.16.204.204>
    WebVPN access GRANTED: telnet://10.43.43.101/
%ASA-6-716004: Group <GroupPolicy1> User <user1> IP <172.16.204.204>
    WebVPN access DENIED to specified location: ssh://10.43.43.102/
%ASA-6-716004: Group <GroupPolicy1> User <user1> IP <172.16.204.204>
```

```

!
WebVPN access DENIED to specified location: http://10.43.43.101/
!
ASA# show vpn-sessiondb detail webvpn | begin Tunnels
Clientless Tunnels: 1
Clientless:
  Tunnel ID      : 13.1
  Public IP      : 172.16.204.204
[ output suppressed ]
  Client Type    : Web Browser
  Client Ver     : Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
  Bytes Tx       : 72131
  Bytes Rx       : 20916
  Filter Name    : WEB-ACL1

```

Example 17-16 shows the VLAN Mapping feature in action. This additional mechanism that can be used in both clientless and client-based environments defines a specific VLAN to which the user must remain confined throughout the VPN session.

Example 17-16 Controlling Clientless Access with VLAN Restriction

```

! VLAN Mapping is configured at the Group Policy level
group-policy GroupPolicy1 attributes
  vlan 44
!
ASA# show vpn-sessiondb webvpn
Session Type: WebVPN
Username      : user1
Index         : 15
Public IP     : 172.16.204.204
Protocol      : Clientless
[ output suppressed ]
NAC Result    : Unknown
VLAN Mapping  : Static
VLAN          : 44
!
! Access granted but routing fails
%ASA-6-716003: Group <GroupPolicy1> User <user1> IP <172.16.204.204>
WebVPN access GRANTED: http://10.43.43.101//
%ASA-6-302013: Built outbound TCP connection 362 for
dmz2:10.43.43.101/80 (10.43.43.101/80) to identity:10.44.44.2/45613
(10.44.44.2/45613)
%ASA-6-110003: Routing failed to locate next hop for TCP from
identity:10.44.44.2/45613 to dmz2:10.43.43.101/80
%ASA-6-302014: Teardown TCP connection 362 for dmz2:10.43.43.101/80
to identity: 10.44.44.2/45613 duration 0:00:03 bytes 0 No valid
adjacency

```

Example 17-17 registers the operation of the *group-lock* feature, a classic access control mechanism that can be used in both SSL (clientless and client-based) and IPsec remote access scenarios. The purpose of this resource is to ensure that the user is granted only access for the group to which he was originally assigned (on the AAA server). Given that many access permissions are defined at the group level, this is a relevant layer of protection that, ideally, should always be considered.

Some other details regarding *group-lock* are summarized in the following list:

- For AAA-based access control, the normal behavior of ASA is to assign the user to the *DefaultWEBVPNGroup*, as shown in Example 17-14. The first requirement for *group-lock* is to enable the **tunnel-group-list** to be presented to the user during login time. (A group name displays as a preconfigured *alias*, and not the actual name.)
- If *group-lock* is not configured (and **tunnel-group-list enable** is), users can log into any group (other than their own). The only requirement for succeeding in such an attempt would be to present a correct username/password combination.
- *Group-lock* is governed by the IETF RADIUS standard attribute #25, which is called *class*. This parameter can be sent to ASA directly via RADIUS or, alternatively, via some LDAP parameter that will be later mapped to the RADIUS *class*.
- The argument used for the **group-lock value** command is the name of an existing Tunnel-Group (called Connection Profile in ASDM terminology).

Example 17-17 *Group-lock in Action*

```
! Enabling ASA to display a list of groups at the logon screen
webvpn
tunnel-group-list enable
!
! Binding the Group-Policy to the Tunnel-Group (Connection Profile)
group-policy GroupPolicy1 attributes
    group-lock value SSL_Group1
!
! User logs into the correct group
%ASA-6-113003: AAA group policy for user user1 is being set to GroupPolicy1
%ASA-6-113011: AAA retrieved user specific group policy
(GroupPolicy1) for user = user1
%ASA-6-113009: AAA retrieved default group policy (GroupPolicy1) for
user = user1
%ASA-6-113008: AAA transaction status ACCEPT : user = user1
%ASA-6-734001: DAP: User user1, Addr 172.16.204.201, Connection
Clientless: The      following DAP records were selected for this
connection: DfltAccessPolicy
    %ASA-6-716001: Group <GroupPolicy1> User <user1> IP <172.16.204.201>
WebVPN session started.
```

```
%ASA-6-716038: Group <GroupPolicy1> User <user1> IP <172.16.204.201>
Authentication: successful, Session Type: WebVPN.
!
ASA# show vpn-sessiondb webvpn
Username      : user1                               Index      : 24
Public IP     : 172.16.204.201
Protocol      : Clientless
[ output suppressed ]
Group Policy : GroupPolicy1           Tunnel Group : SSL_Group1
!
! User tries to log into a different group
%ASA-6-113003: AAA group policy for user user1 is being set to
GroupPolicy1
%ASA-6-113011: AAA retrieved user specific group policy
(GroupPolicy1) for user = user1
%ASA-6-113009: AAA retrieved default group policy (GroupPolicy2) for
user = user1
%ASA-6-113008: AAA transaction status ACCEPT : user = user1
%ASA-6-734001: DAP: User user1, Addr 172.16.204.201, Connection
Clientless: The      following DAP records were selected for this
connection: DfltAccessPolicy
!
! Login into wrong group ('debug webvpn' perspective - only relevant
messages)
webvpn_login_transcend_cert_auth_cookie: tg_cookie = NULL, tg_name = SSL_Group2
webvpn_login_resolve_tunnel_group: tunnel group name from group list
WebVPN: user: (user1) authenticated.
webvpn_auth.c:http_webvpn_auth_accept[2939]
User came in on group he wasn't supposed to come in on!
```

Tip The value passed to ASA via the *class* attribute must be a string corresponding to the Group Policy name. The **group-lock value** command, within the context of a Group Policy, takes care of mapping the received string to the appropriate *Tunnel Group*. In ACS, for example, the value **OU=GroupPolicy1** for the *class* attribute instructs ASA to assign the Group Policy called **GroupPolicy1** to the user being authenticated.

Tip The LDAP authentication method was used in the examples. A possible configuration is to map the *department* LDAP parameter to the Radius *class* attribute (internally used by ASA) as shown in the following:

```
ldap attribute-map department-to-GP
  map-name department IETF-Radius-Class
!
aaa-server MS_AD protocol ldap
aaa-server MS_AD (dmz2) host 10.44.44.250
  ldap-attribute-map department-to-GP
```

ASA supports the mapping of parameters contained in digital certificates to automatically select the group to which the user is assigned. If you use certificate-based access control in your Remote Access VPN deployment, this is an option to know about.

Client-Based Access (AnyConnect)

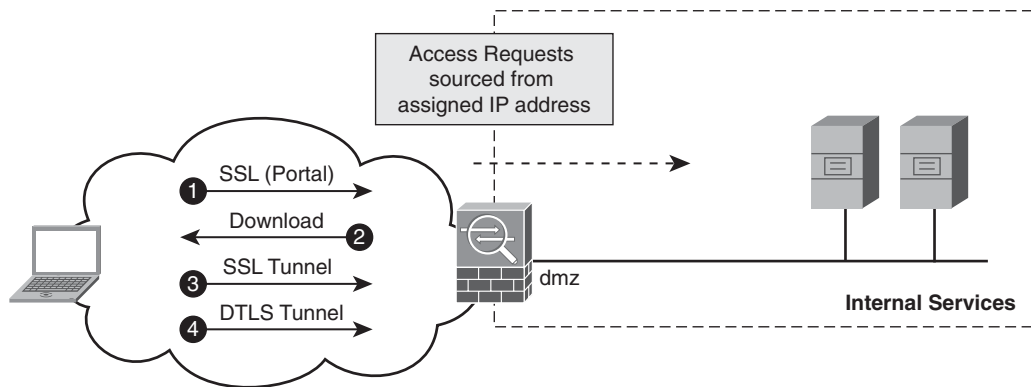
One interesting point of the Cisco's SSL-VPN implementation resides in the possibility of downloading (and installing) the AnyConnect client from the portal. This ensures mobility even for those users who need client-based access.

Datagram Transport Layer Security (DTLS) protocol was created as a sort of UDP-based version of *Transport Layer Security* (TLS). Two main factors contributed to the introduction of this new transport protocol:

- The underlying goal to use SSL-VPN as a potential replacement for IPsec in the Remote Access arena. Such an ambitious proposal requires the ability of conveying any kind of traffic over the SSL connection.
- The known drawbacks of TCP with respect to the transport of real-time traffic.

Figure 17-27 briefly describes, as a reference, the situation in which a user obtains the AnyConnect client from the Web VPN Portal. The steps are summarized as follows:

1. The user establishes an SSL clientless connection to the portal.
2. The user selects the **Start AnyConnect** option from the portal, which actually instructs the ASA to download and install the client.
3. Upon successful installation of the client, AnyConnect automatically establishes an SSL tunnel to the ASA appliance.
4. After finishing the setup of the regular SSL tunnel, DTLS negotiation takes place. During this stage, data passes through the first tunnel.



```
%ASA-6-302013: Built inbound TCP connection 38 for vpn2:172.16.204.201/1076 (172.16.204.201/1076)
to identity:172.16.204.2/443 (172.16.204.2/443)
%ASA-6-725001: Starting SSL handshake with client vpn2:172.16.204.201/1076 for TLSv1 session.
%ASA-6-725002: Device completed SSL handshake with client vpn2:172.16.204.201/1076
%ASA-5-737003: IPAA: DHCP configured, no viable servers found for tunnel-group 'DefaultWEBVPNGroup'
%ASA-6-737026: IPAA: Client assigned 10.203.203.101 from local pool
%ASA-6-737006: IPAA: Local pool request succeeded for tunnel-group 'DefaultWEBVPNGroup'
%ASA-5-722033: Group <GroupPolicy3> User <user3> IP <172.16.204.201>
First TCP SVC connection established for SVC session.
[-----]
%ASA-6-302015: Built inbound UDP connection 39 for vpn2:172.16.204.201/1077 (172.16.204.201/1077)
to identity:172.16.204.2/443 (172.16.204.2/443)
%ASA-6-725001: Starting SSL handshake with client vpn2:172.16.204.201/1077 for DTLSv1 session.
%ASA-6-725003: SSL client vpn2:172.16.204.201/1077 request to resume previous session.
%ASA-6-725002: Device completed SSL handshake with client vpn2:172.16.204.201/1077
%ASA-5-722033: Group <GroupPolicy3> User <user3> IP <172.16.204.201>
First UDP SVC connection established for SVC session.
```

Figure 17-27 Basic Model for Anyconnect Access

When the DTLS tunnel becomes available, data preferably flows through it. Its SSL companion is then used mainly for control traffic or as a backup path if the DTLS connection is torn down.

Note In Cisco SSL-VPN terminology, SVC stands for *SSL VPN Client*.

Figure 17-28 shows a sample SSL-VPN topology and assembles the commands used to allow client-based connections to the group called `SSL_Group3`. (Notice the `vpn-tunnel-protocol` under the `group-policy` definition.)

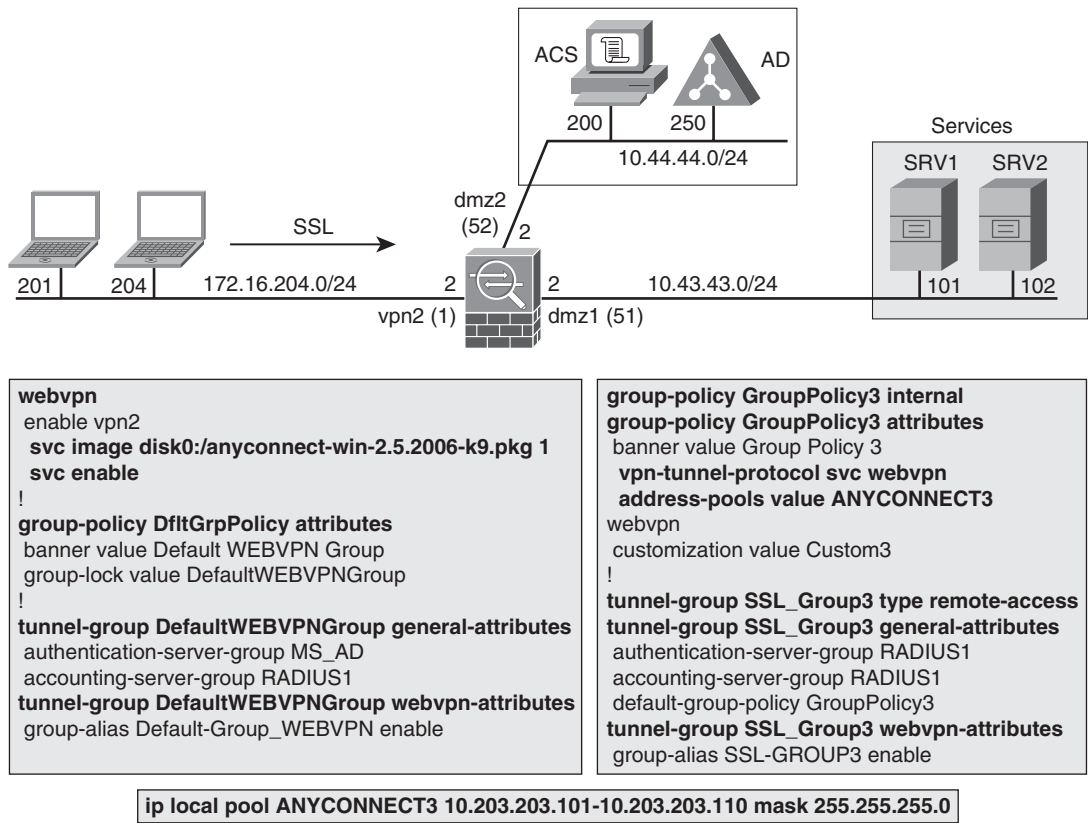


Figure 17-28 Reference Topology for AnyConnect Analysis

Example 17-18 refers to the scenario of Figure 17-28 and documents the three connections that compose an AnyConnect session:

- Clientless connection
- Regular SSL-VPN tunnel (TCP-based)
- DTLS tunnel (UDP based)

The first portion of the output of the `show vpn-sessiondb detail svc` command displays condensed information about the session. For instance, the protocol is presented as

Clientless SSL-Tunnel DTLS-Tunnel, which is an indication three components are involved.

Example 17-18 *Sample AnyConnect Session (3 Connections)*

```

ASA# show vpn-sessiondb detail svc
Session Type: SVC Detailed
Username      : user3                      Index      : 4
Assigned IP   : 10.203.203.101             Public IP   : 172.16.204.201
Protocol      : Clientless SSL-Tunnel DTLS-Tunnel
License       : SSL VPN
[ output suppressed ]
Clientless:
  Tunnel ID    : 4.1
  Public IP    : 172.16.204.201
  Encryption   : RC4                      Hashing     : SHA1
  Encapsulation: SSLv3                    TCP Dst Port: 443
  Auth Mode    : userPassword
  Idle Time Out: 30 Minutes                Idle TO Left: 29 Minutes
  Client Type  : Web Browser
  Client Ver   : Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
  Bytes Tx     : 225517                    Bytes Rx    : 40360
SSL-Tunnel:
  Tunnel ID    : 4.2
  Assigned IP   : 10.203.203.101             Public IP   : 172.16.204.201
  Encryption   : RC4                      Hashing     : SHA1
  Encapsulation: TLSv1.0                  TCP Src Port: 1289
  TCP Dst Port : 443                      Auth Mode   : userPassword
  Idle Time Out: 30 Minutes                Idle TO Left: 29 Minutes
  Client Type  : SSL VPN Client
  Client Ver   : Cisco AnyConnect VPN Agent for Windows 2.5.2006
  Bytes Tx     : 620                      Bytes Rx    : 0
  Pkts Tx      : 1                        Pkts Rx     : 0
  Pkts Tx Drop : 0                       Pkts Rx Drop: 0
DTLS-Tunnel:
  Tunnel ID    : 4.3
  Assigned IP   : 10.203.203.101             Public IP   : 172.16.204.201
  Encryption   : AES128                   Hashing     : SHA1
  Encapsulation: DTLSv1.0                UDP Src Port: 1290
  UDP Dst Port : 443                      Auth Mode   : userPassword
  Idle Time Out: 30 Minutes                Idle TO Left: 29 Minutes
  Client Type  : DTLS VPN Client
  Client Ver   : Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
  Bytes Tx     : 0                        Bytes Rx    : 1250
  Pkts Tx      : 0                        Pkts Rx     : 14

```

```

Pkts Tx Drop : 0                               Pkts Rx Drop : 0
!
! Connections associated with an AnyConnect session

ASA# show conn all
3 in use, 6 most used
TCP vpn2 172.16.204.201:1291 NP Identity Ifc 172.16.204.2:443, idle 0:00:44, bytes
7410, flags UOB
UDP vpn2 172.16.204.201:1290 NP Identity Ifc 172.16.204.2:443, idle 0:00:05, bytes
15055, flags -
TCP vpn2 172.16.204.201:1289 NP Identity Ifc 172.16.204.2:443, idle 0:00:01, bytes
2763, flags UOB

```

Example 17-19 presents two options for establishing post-decryption access control in the AnyConnect environment:

- A group level option, based on the **vpn-filter** command. This option employs regular (extended) ACLs.
- Interface ACLs that become effective after configuring the **no sysopt connection permit-vpn** command. In this example, two **access-lists** have been defined to better illustrate the feature behavior: an inbound ACL called VPN2 and an outbound one named DMZ1-OUT.

Example 17-19 Controlling AnyConnect Access with ACLs

```

! Applying an ACL (extended access-list) at the group Level
group-policy GroupPolicy3 attributes
  vpn-filter value ACL1
!
%ASA-6-106102: access-list ACL1 denied icmp vpn2/10.203.203.101(8) ->
dmz1/10.43.43.101(0) hit-cnt 1 first hit [0xc365c0b6, 0x0]
%ASA-6-106102: access-list ACL1 denied tcp vpn2/10.203.203.101(1183)
-> dmz1/10.43.43.101(80) hit-cnt 1 first hit [0x51951df6, 0x0]
!
! ACLs applied at the interface level

no sysopt connection permit-vpn
!
access-list VPN2 extended permit icmp any 10.43.43.0 255.255.255.0 echo log
interval 120
access-list VPN2 extended permit tcp 10.0.0.0 255.0.0.0 host 10.43.43.101 eq tel-
net log interval 120
access-group VPN2 in interface vpn2
!
access-list DMZ1-OUT extended permit icmp 10.0.0.0 255.0.0.0 10.43.43.0
255.255.255.0 echo log interval 120

```

```

access-group DMZ1-OUT out interface dmz1
!
! Ping from 10.203.203.102 to 10.43.43.101 (permitted by both ACLs)

%ASA-6-106100: access-list VPN2 permitted icmp vpn2/10.203.203.102(8) ->
dmz1/10.43.43.101(0) hit-cnt 1 120-second interval [0x89f638bc, 0x0]
%ASA-6-106100: access-list DMZ1-OUT permitted icmp
vpn2/10.203.203.102(8) -> dmz1/10.43.43.101(0) hit-cnt 1 first hit
[0x5bf63fa6, 0x0]
%ASA-6-302020: Built inbound ICMP connection for faddr
10.203.203.102/768 gaddr 10.43.43.101/0 laddr 10.43.43.100/0 (user3)
!
! Telnet from 10.203.203.102 to 10.43.43.101 (denied by the DMZ1-OUT ACL)

%ASA-6-106100: access-list VPN2 permitted tcp vpn2/10.203.203.102(1701) ->
dmz1/10.43.43.101(23) hit-cnt 1 first hit [0xe8145039, 0x0]
%ASA-4-106023: Deny tcp src vpn2:10.203.203.102/1701 dst dmz1:10.43.43.101/23
by access-group "DMZ1-OUT" [0x0, 0x0]

```

It is convenient to emphasize that the *VLAN Restriction* and *group-lock* features, earlier studied for clientless access are equally available for AnyConnect.

Firewalls and MPLS Networks

I recall many customer-facing conversations in which I was asked about correct firewall placement when dealing with MPLS-based networks. Although multiple answers exist for this question, a good starting point is to understand *where not* to place the firewall.

Figure 17-29 provides an overview of a typical MPLS network, highlighting the categories of the routers involved, and clearly delimiting the IP lookup and Label lookup domains.

Features that rely on IP lookup must not be used inside the Label domain, meaning that, for instance, there should not be any ACL (let alone a firewall) between a P and PE or between two P-routers. This is paramount to avoid breaking label switching in the MPLS core.

Figure 17-30 illustrates the separation of routes naturally accomplished by the MPLS-VPN architecture and provides the backdrop for a brief discussion about some key MPLS characteristics. For the analysis that follows, please assume that, in this topology, the PEs provide basic Intranet connectivity services to each connected customer, which reflects in the existence of three independent routing tables listed below:

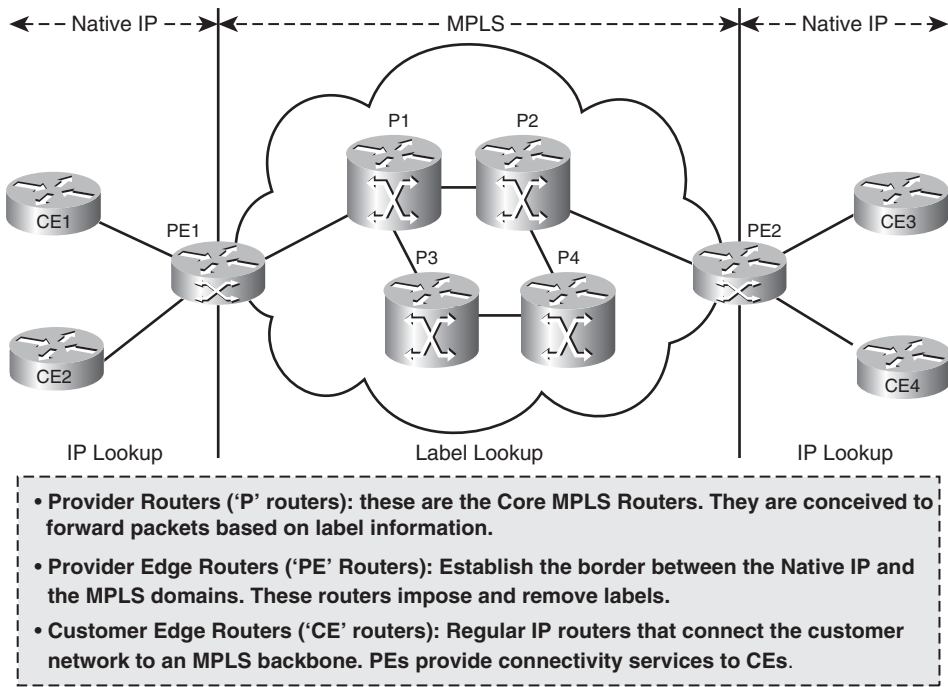


Figure 17-29 Overview of the MPLS Infrastructure

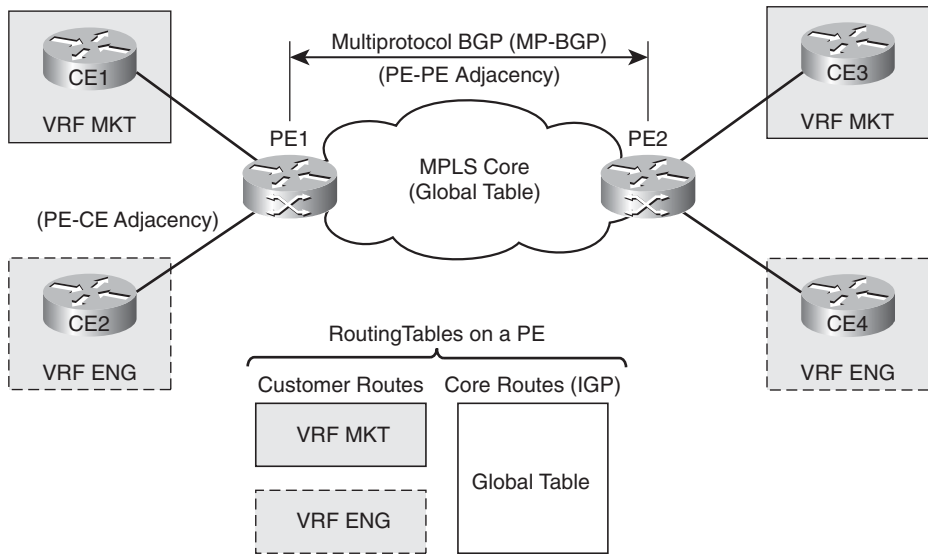


Figure 17-30 Routing Tables on a PE Router

- **The Global table:** This is the IGP-based routing table used to interconnect the PEs and serves as the reference for label allocation in the core using the Label Distribution Protocol (LDP).
- **Two customer VRFs:** Each VRF table concentrate sthe routes belonging to a specific customer.

The other router classes represented in Figure 17-29 have the following perspective with respect to routing tables:

- P-routers know only the Global table and are completely unaware of the existence of any customer VRF. (This is a feature, not a bug.) This sort of routing hierarchy contributes decisively to security, scalability, and stability within the core.
- CE-routers know about only their own table, which is, by default, the Global Table.

Chapter 6 examines how VRFs provide the *local segmentation* of routes in a router, a concept that is of particular importance on PEs that acquire customer routes over a PE-CE adjacency. After receiving the routes from a certain CE (and placing them in the appropriate VRF), a PE inserts a special type of identifier (the *Route Target* extended community) for each customer prefix exported via Mutiprotocol BGP (MP-BGP) to another PE. The receiving PE relies on this ID to correctly select the routes imported to each of its locally defined VRFs.

The MPLS-VPN model has the capability to separate routes at various levels (VRF from VRF and any VRF from the core), without the need to use ACLs, NAT, or any similar resources. From an MPLS service provider's point of view, the core is a trusted zone, and the weak link corresponds to the PE-CE links. (CEs are always deemed untrusted, for they are typically installed within customer premises.)

On the other end of the spectrum, the CE routers do not have much choice. Their gateway to the rest of the world is precisely the PE, and as such they implicitly rely on the MPLS-VPN architecture's capability to isolate them from other customers.

These two complementary perspectives make the PE-CE neighborhood the first candidate spot for the insertion of a firewall or, at least, for the configuration of firewall features. Some of the protection functionalities that can be enabled on the PE-CE link are described in the following list:

- The CE routers within a particular VRF can be upgraded to support the ZFW features, thereby adding the notion of *perimeter* and facilitating the definition of trust relationships. To guide the creation of the Zone-based policies, it is convenient to notice that each VRF on a PE contains the routes from the whole VPN address space of that customer. Knowing the traffic interest between sites, you can configure the pertinent inspection rules (including restrictions for acceptable pairings of source and destination addresses).

- Enabling routing protocol authentication between PE and CE decreases the risk of creating an adjacency with a rogue router. This is even more important when using Ethernet-based connectivity because of the multiaccess nature of Ethernet.
- Antispoofing features like uRPF are more than welcome on both side of the links.
- Establishing an upper bound for the number of routes learned by a PE (on a per-VRF basis) effectively helps to mitigate routing-related DoS attacks.
- Enforcing QoS definitions (on both the PE and CE routers) can also help to achieve DoS contention.

Figure 17-31 reveals that the CEs in the ENG VRF, a more security conscious customer, have been upgraded to support the ZFW functionality. Further, a centralized firewall has been deployed to provide Internet access to this particular VRF. In this scenario, a default route (within the context of VRF ENG) is in charge of directing Internet bound traffic to PE3, which in turn contacts the CENTRAL1 CE.

At this point, it should be clear that the firewalls are always located outside the MPLS cloud.

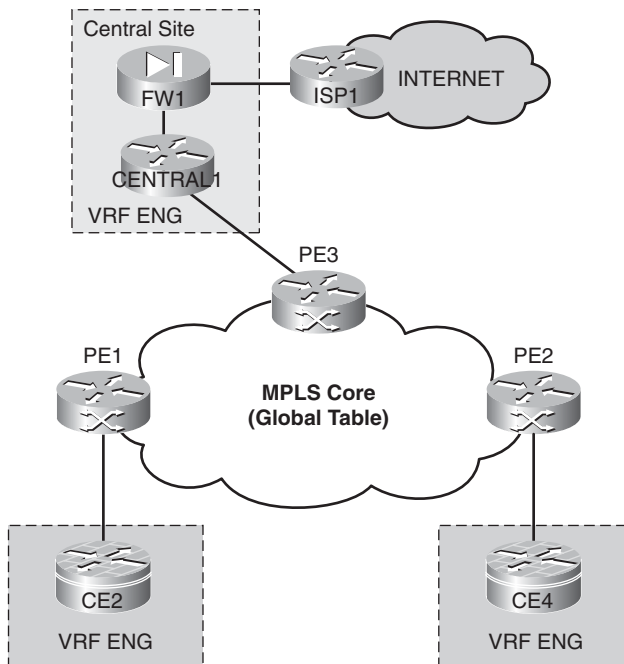


Figure 17-31 A Possible Model of Centralized Internet Access on MPLS-VPN Environments

Note This was just a quick discussion about issues pertaining to MPLS, a vast topic that has been covered in many *Cisco Press* books. Further discussion is naturally beyond the scope of this book.

Network and security architects should never forget that the MPLS-VPN service excels on routing segmentation, while still providing each customer with any-to-any connectivity among their respective sites. However, this class of VPN does not include any provision for resources such as confidentiality. (This is a design feature and not a limitation.)

If your security policy mandates the use of encryption on the WAN domain, you should consider the Cisco Group Encrypted Transport VPN solution (GET VPN), an IPsec heir that is a perfect add-on to MPLS-VPN. When combined, these solutions deliver an incredible security level while still ensuring optimal routing. GET VPN and ZFW are available in the same IOS feature set. If you buy a security-enabled IOS router, you can take advantage of both resources.

Borderless Networks Vision

As stated in the beginning of the first chapter, networks have become flexible enough to accommodate the concurrent transport of data, voice, and video applications. Intelligent networks proved to be a leading factor for the productivity and success of organizations and, as such, are perceived as a foundational business asset.

Although this all remains true, some additional trends have been reshaping the future of IT and networking. These include the following:

- **Cloud computing:** This concept is often described as the capability to abstract classic IT resources from the physical infrastructure and deliver them in the *X as a service* model (where X typically represents software, platform, or infrastructure). Cisco goes a bit further on the definition and characterizes cloud-based services as those that can be provided *on demand* and *at scale*, on a multitenant environment managed by someone else.
- **Mobility:** Besides the natural task of providing access to employees, there should be some safe means to grant access to various classes of external users (customers, contractors, partners, and guests). This needs to be accomplished even for devices not under the control of the company's IT department, for an ever-increasing number of categories of computing elements. One critical aspect here is the ability to *qualify* the machine from which access is requested, for the sake of minimizing the likelihood of compromises being caused by such a machine.
- **Media-rich collaboration solutions:** Cisco Webex and Cisco Telepresence are great examples of these new offerings that enable true collaboration, regardless of users being located outside the corporate domains.

- **Culture of connectivity:** Smart mobile wireless equipments such as the *iPhones* and *BlackBerrys* are proliferating quickly and providing the average user with instant access to multimedia applications, social networking tools, and other attractive resources. And when home users are presented with such a variety of options on their personal devices, they expect to have similar functionality within the companies they work. (Acceptable Use Policies should come into play to reflect what level of access is the adequate for a particular Organization).

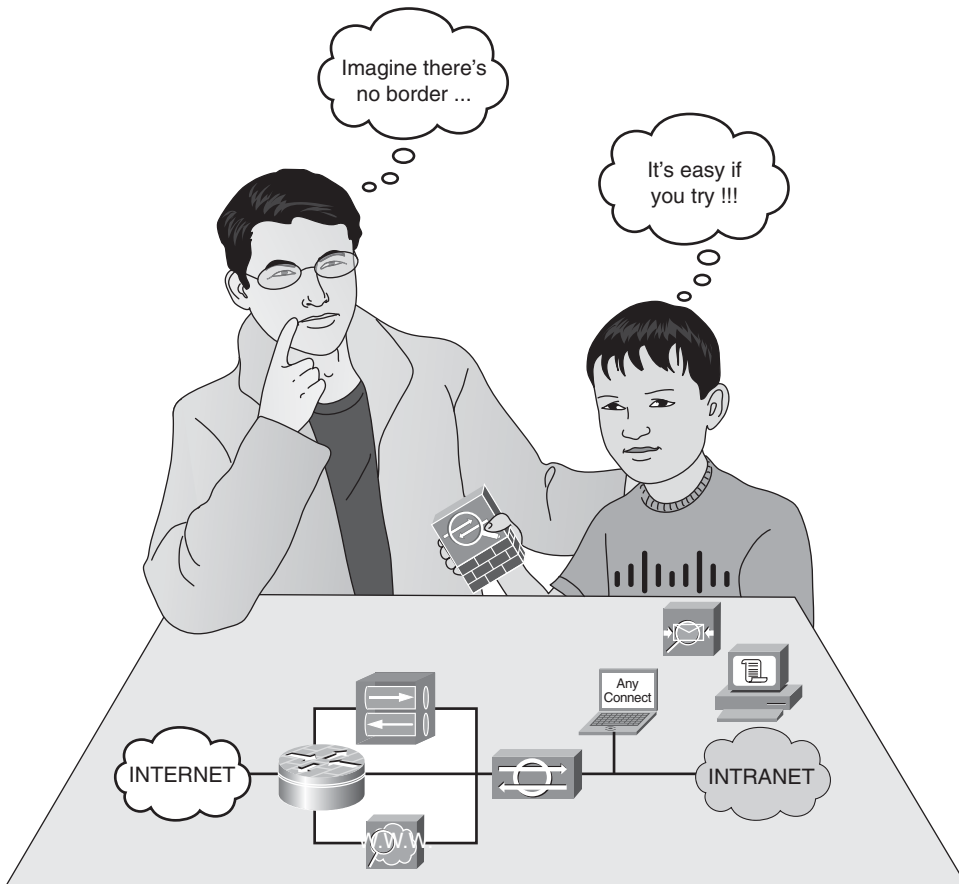
Note The *on demand*, *at scale*, and *multitenant* attributes of the Cisco cloud computing definition are generally associated with applications hosted on Virtual Machines. As shown in this chapter, firewalls (either external or internal to the physical server) can effectively protect this type of scenario.

The topics just enumerated characterize that the classic notion of the network perimeter is being redefined. And this new *borderless* paradigm, in which access to the IT resources of an organization might be initiated by any category of user, from multiple types of devices and using all sorts of connectivity models, requires the security measures in place to be adapted accordingly.

The Cisco Security Architecture for Borderless Networks is defined by taking into account these new requirements and mainly consists of the elements listed in the following:

- **Scanning engines:** The entities taking care of tasks such as content inspection, application recognition, user authentication (employing both identity and posture-based information) and anomaly detection. The scanning engines are the components in charge of policy enforcement.
- **Policy management:** The central policy definition points in which the actual rules are created and maintained. These security rules are delivered to the scanning engines for policy enforcement.
- **Security intelligence services:** A combination of products, solutions, and experienced people responsible for clearly separating the good from the bad traffic. This helps prevent actual damage to a company's assets. Botnet Traffic Filtering on ASA appliances, Global Correlation data for Cisco IPS, and reputational databases such as Ironport's *Senderbase* are examples of such services.
- **The borderless agent:** A lightweight piece of software present in every type of networked device, with the well-defined responsibility of directing the connections (wired, wireless, VPN, and so on) to the appropriate type of scanning engine. This approach virtually extends the corporate borders and provides the secure and flexible access needed in a true borderless environment.

The Cisco AnyConnect client, whose initial functionality has been already described in the SSL-VPN section, is receiving lots of investments so that it can materialize the intelligent software agent at the heart of a truly secure borderless network. Other well-known Cisco products such as ASA and Ironport appliances, IPS devices, and the Cisco Secure Access Control Server (CS-ACS) are also key constituents of the borderless architecture.



Summary

This chapter examined some interactions of the firewall functionality with other systems and features that can add value to the security design. The list of topics included was not meant to be exhaustive but, hopefully, it might have been somehow *useful*.

The overall contents, approach, and organization of this book should bring contribute to at least one of the following domains:

- Providing you with accurate information about the theory of operations of Cisco firewall features and demonstrating how they can be leveraged to produce better security designs
- Reinforcing the perception that the more you invest on building your networking knowledge, the more you can secure real-life networked environments
- Promoting the vision that security practice requires a systems approach, rather than a focus on mere products

Presenting some thought-provoking discussions to stimulate you the to study, plan, and practice more about security

Further Reading

Network Security Technologies and Solutions (Yusuf Bhajji)

<http://www.ciscopress.com/bookstore/product.asp?isbn=1587052466>

Network Security Architectures (Sean Convery)

<http://www.ciscopress.com/bookstore/product.asp?isbn=158705115X>

MPLS VPN Security (Michael H. Behringer, Monique J. Morrow)

<http://www.ciscopress.com/bookstore/product.asp?isbn=1587051834>

Group Encrypted Transport VPN (GET VPN) – Design and Implementation Guide

http://www.cisco.com/en/US/prod/collateral/vpndevc/ps6525/ps9370/ps7180/GETVPN_DIG_version_1_0_External.pdf

Cisco Software-as-a-Service (SaaS) Access Control – Cisco Anyconnect Mobility Client

http://www.cisco.com/en/US/prod/collateral/vpndevc/ps5743/ps5699/ps10884/white_paper_c11-596141.html

NAT and ACL Changes in ASA 8.3

This appendix covers the following topics:

- Network Object NAT examples
- Twice NAT examples
- Summary of NAT syntaxes in ASA 8.3
- Global ACLs

“We shall not cease from exploration/And the end of all our exploring/Will be to arrive where we started/And know the place for the first time” —T.S. Eliot.

Most of the Adaptive Security Appliance (ASA) topics studied throughout the book assume the use of ASA version 8.2. This appendix presents some important changes introduced by ASA 8.3 in the way ASA handles Network Address Translation (NAT) and Access Control Lists (ACL):

- **NAT Redesign:** NAT is completely redesigned in 8.3 code to simplify operations and facilitate creation of rules that concurrently translate source and destination. The ability to manually establish the order in which NAT statements are processed is also a key feature of the new NAT model. In 8.3, two NAT modes are available:
 - **Network Object NAT:** The translation rule is defined inside a network object. This mode, well suited for source-only rules, may be referred to as *Auto NAT* because the entries are automatically ordered by the ASA appliance according to a set of predefined internal guidelines.
 - **Twice NAT:** This method is adequate for Policy NAT and whenever source and destination addresses need to be translated simultaneously. Twice NAT is often called *Manual NAT* because it renders the manual definition of NAT processing order possible.
- **Unified NAT Table:** The NAT table for ASA 8.3 is divided into three sections. These sections are illustrated in Figure A-1, and the rules are processed in a top-down order

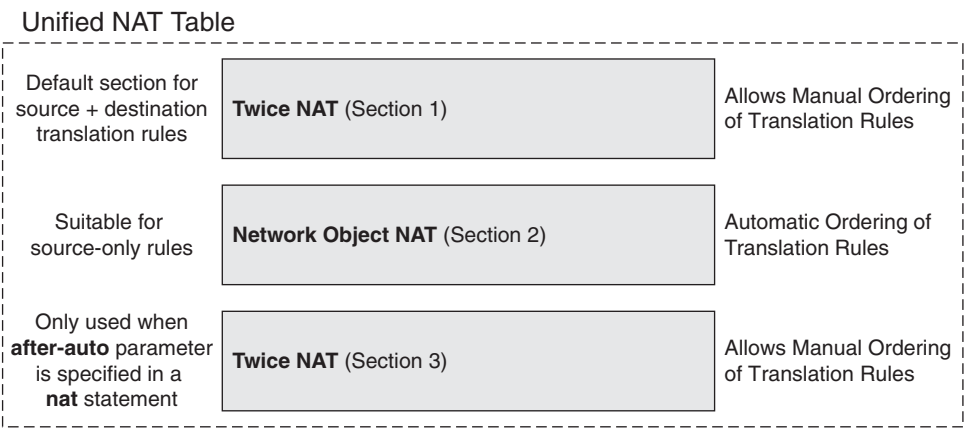


Figure A-1 The Unified NAT Table in ASA 8.3

with first match. Network Object NAT entries are always placed in Section 2, whereas Twice NAT rules are, by default, inserted in Section 1.

- **Use of Real Address within Access Control Entries (ACEs):** ASA 8.3 code removes the need of specifying the pre-NAT and post-NAT addresses within ACEs. Regardless of any kind of translation, the rule must always reference the real address.
- **Global ACLs:** Besides the classic interface ACLs, ASA 8.3 offers the possibility of using access control rules that simultaneously applies to all ASA interfaces. The use of Global ACLs does not change the definition of the Access Control Entries (ACE), already studied in Chapter 7, “Through ASA Without NAT.” It simply changes the way these rules are applied.

This appendix extensively employs examples to highlight the differences between 8.3 and previous ASA releases. If you need any conceptual reference about ACL construction or the behavior of each category of NAT (static, PAT, dual NAT, and so on), Chapters 7 and 8, “Through ASA Using NAT,” should respectively be revisited.

Network Object NAT Examples

The main characteristics associated with this NAT construction method follow:

- The real address is informed in a network object definition. This object can be later used in another section of the configuration, for instance in an ACL.
- The translated address is created by using a **nat** statement as a parameter under the network object.
- Object NAT rules are automatically placed in Section 2 of the unified NAT table.
- An individual Object NAT rule can be employed to translate either the source or the destination address of a packet but not both at the same time. If Policy NAT behavior is wanted, Twice NAT is the natural choice.

Figure A-2 depicts the reference topology for the NAT examples presented throughout this appendix. One noteworthy change, with respect to pre-8.3 releases, is that the real addresses are now used to specify the destination in the ACL definition.

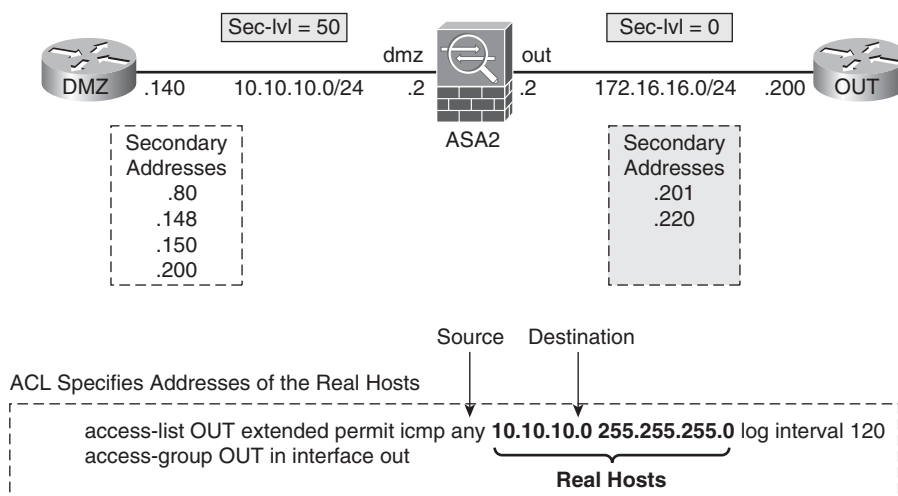


Figure A-2 Reference Topology for NAT Analysis

Example A-1 assembles the network objects later used for the illustration of NAT rules.

Example A-1 Network Objects Used for NAT Analysis

```
! Real Addresses on subnet 10.10.10.0/24
object network N10-R140
  host 10.10.10.140
object network N10-R148
  host 10.10.10.148
object network N10-S24
  subnet 10.10.10.0 255.255.255.0
object network N10-HIGH-S25
  subnet 10.10.10.128 255.255.255.128
object network N10-HIGH-S26
  subnet 10.10.10.128 255.255.255.192
object network N10-HIGH-S27
  subnet 10.10.10.128 255.255.255.224
object network N10-HIGH-S28
  subnet 10.10.10.128 255.255.255.240
!
! Real Addresses on subnet 172.16.16.0/24
object network N16-R200
```

```

host 172.16.16.200
!
! Translated Addresses on subnet 172.16.16.0/24

object network N16-T140
  host 172.16.16.140
object network N16-T148
  host 172.16.16.148
object network N16-RANGE1
  range 172.16.16.129 172.16.16.254
object network N16-PAT1
  host 172.16.16.125
object network N16-PAT2
  host 172.16.16.126
!
! Translated Addresses on subnet 172.17.17.0/24

object network N17-RANGE1
  range 172.17.17.129 172.17.17.158
!
! Translated Addresses on subnet 172.18.18.0/24

object network N18-HIGH-S28
  subnet 172.18.18.128 255.255.255.240
object network N18-T148
  host 172.18.18.148

```

Dynamic NAT

The NAT rule in Example A-2 establishes that *dmz* addresses in the 10.10.10.128/25 subnet should be mapped to the range 172.16.16.129-254, when connecting to destinations reachable via the *out* interface. The example shows NAT in action for a ping from 10.10.10.150 to 172.16.16.200.

Example A-2 *Dynamic NAT*

```

! Dynamic NAT Rule
object network N10-HIGH-S25
  nat (dmz,out) dynamic N16-RANGE1
!
! Ping from 10.10.10.150 to 172.16.16.200

%ASA-6-305009: Built dynamic translation from dmz:10.10.10.150 to
out:172.16.16.248
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.200/0 gaddr
172.16.16.248/2 laddr 10.10.10.150/2

```

```

ICMP echo request from dmz:10.10.10.150 to out:172.16.16.200 ID=2 seq=1 len=72
ICMP echo request translating dmz:10.10.10.150 to out:172.16.16.248
%ASA-6-106100: access-list OUT permitted icmp out/172.16.16.200(0) ->
dmz/10.10.10.150(0) hit-cnt 1 first hit [0x8577a13e, 0x0]
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.200/0 gaddr
172.16.16.248/2 laddr 10.10.10.150/2
ICMP echo reply from out:172.16.16.200 to dmz:172.16.16.248 ID=2 seq=1 len=72
ICMP echo reply untranslating out:172.16.16.248 to dmz:10.10.10.150
!
ASA# show nat interface dmz detail
Auto NAT Policies (Section 2)
1 (dmz) to (out) source dynamic N10-HIGH-S25 N16-RANGE1
    translate_hits = 1, untranslate_hits = 1
    Source - Origin: 10.10.10.128/25, Translated: 172.16.16.129-172.16.16.254
!
ASA# show xlate
Flags: D - DNS, i - dynamic, r - portmap, s - static, I - identity, T - twice
NAT from dmz:10.10.10.150 to out:172.16.16.248 flags i idle 0:01:04 timeout 3:00:00

```

Dynamic PAT

The NAT rule shown in Example A-3 states that *dmz* hosts in the 10.10.10.0/24 network should undergo Port Address Translation (PAT) using the global address 172.16.16.126.

Example A-3 *Dynamic PAT*

```

! Dynamic PAT Rule
object network N10-S24
    nat (dmz,out) dynamic N16-PAT2
!
%ASA-6-305011: Built dynamic ICMP translation from dmz:10.10.10.80/3 to
out:172.16.16.126/41388
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.200/0 gaddr
172.16.16.126/41388 laddr 10.10.10.80/3
ICMP echo request from dmz:10.10.10.80 to out:172.16.16.200 ID=3 seq=1 len=72
ICMP echo request translating dmz:10.10.10.80/3 to out:172.16.16.126/41388
%ASA-6-106100: access-list OUT permitted icmp out/172.16.16.200(0) ->
dmz/10.10.10.80(0) hit-cnt 1 first hit [0x8577a13e, 0x0]
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.200/0 gaddr
172.16.16.126/41388 laddr 10.10.10.80/3
ICMP echo reply from out:172.16.16.200 to dmz:172.16.16.126 ID=41388 seq=1 len=72
ICMP echo reply untranslating out:172.16.16.126/41388 to dmz:10.10.10.80/3
!
ASA# show nat interface dmz detail
Auto NAT Policies (Section 2)
1 (dmz) to (out) source dynamic N10-S24 N16-PAT2

```



```

        translate_hits = 1, untranslate_hits = 1
        Source - Origin: 10.10.10.0/24, Translated: 172.16.16.126/32
    !
ASA# show xlate
Flags: D - DNS, i - dynamic, r - portmap, s - static, I - identity, T - twice
ICMP PAT from dmz:10.10.10.80/3 to out:172.16.16.126/41388 flags ri idle 0:00:15
timeout 0:00:30

```

Static NAT

Example A-4 demonstrates an outbound static NAT rule in which host 10.10.10.148 was mapped to address 172.16.16.148.

Example A-5 illustrates an inbound connection from host 172.16.16.200 to the translated IP 172.18.18.140 of real host 10.10.10.140. In this case a subnet-to-subnet mapping has been employed.

Example A-4 Static NAT Configuration

```

! Creating a static entry (host-to-host)
ASA(config)# object network N10-R148
ASA(config-network-object)# nat (dmz,out) static N16-T148
%ASA-6-305009: Built static translation from dmz:10.10.10.148 to out:172.16.16.148
!
ASA# show nat interface dmz detail
Auto NAT Policies (Section 2)
1 (dmz) to (out) source static N10-R148 N16-T148
    translate_hits = 1, untranslate_hits = 1
    Source - Origin: 10.10.10.148/32, Translated: 172.16.16.148/32
!
ASA# show xlate
Flags: D - DNS, i - dynamic, r - portmap, s - static, I - identity, T - twice
NAT from dmz:10.10.10.148 to out:172.16.16.148
    flags s idle 0:00:14 timeout 0:00:00
!
! Static NAT (subnet-to-subnet)

ASA(config)# object network N10-HIGH-S28
ASA(config-network-object)# nat (dmz,out) static N18-HIGH-S28
%ASA-6-305009: Built static translation from dmz:10.10.10.128 to out:172.18.18.128
!
ASA# show nat interface dmz detail
Auto NAT Policies (Section 2)
1 (dmz) to (out) source static N10-HIGH-S28 N18-HIGH-S28
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.128/28, Translated: 172.18.18.128/28

```

Example A-5 *Inbound Static NAT (1)*

```

ASA# show xlate
Flags: D - DNS, i - dynamic, r - portmap, s - static, I - identity, T - twice
NAT from dmz:10.10.10.128/28 to out:172.18.18.128/28
      flags s idle 0:01:20 timeout 0:00:00
!
! Ping from 172.16.16.200 (out) to 172.18.18.140 (translated address of
10.10.10.140)
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.200/2 gaddr
172.18.18.140/0 laddr 10.10.10.140/0
ICMP echo request from out:172.16.16.200 to dmz:172.18.18.140 ID=2 seq=0 len=72
ICMP echo request untranslating out:172.18.18.140 to dmz:10.10.10.140
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.200/2 gaddr
172.18.18.140/0 laddr 10.10.10.140/0
ICMP echo reply from dmz:10.10.10.140 to out:172.16.16.200 ID=2 seq=0 len=72
ICMP echo reply translating dmz:10.10.10.140 to out:172.18.18.140

```

Example A-6 reveals what happens when a host tries to ping the real host instead of the mapped address in a static NAT scenario. The connection request is denied and the *acl-drop* and *nat-rpf-failed* counters are incremented.

Example A-6 *Inbound Static NAT (2)*

```

! Trying to ping the real address 10.10.10.140 from the 'out' interface
%ASA-5-305013: Asymmetric NAT rules matched for forward and reverse flows;
Connection
for icmp src out:172.16.16.200 dst dmz:10.10.10.140 (type 8, code 0) denied due to
NAT reverse path failure
!
ASA# show asp drop
Frame drop:
    Flow is denied by configured rule (acl-drop)                                1
Last clearing: 22:45:56 BRT Jan 22 2011 by enable_15
Flow drop:
    NAT reverse path failed (nat-rpf-failed)                                2
!
! What packet-tracer reveals

ASA# packet-tracer input out icmp 172.16.16.200 8 0 1 10.10.10.140
[ output suppressed ]
Phase: 6
Type: NAT

```

```

Subtype: rpf-check
Result: DROP
Config:
object network N10-HIGH-S28
  nat (dmz,out) static N18-HIGH-S28
[ output suppressed ]
Action: drop
Drop-reason: (acl-drop) Flow is denied by configured rule

```

Identity NAT

Example A-7 characterizes that Identity NAT is simply a particular case of static NAT in the ASA 8.3 paradigm.

Example A-7 Identity NAT

```

ASA(config)# object network N10-HIGH-S26
ASA(config-network-object)# nat (dmz,out) static N10-HIGH-S26
%ASA-6-305009: Built static translation from dmz:10.10.10.128 to out:10.10.10.128
!
ASA# show nat interface dmz detail
Auto NAT Policies (Section 2)
1 (dmz) to (out) source static N10-HIGH-S26 N10-HIGH-S26
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.128/26, Translated: 10.10.10.128/26
!
ASA# show xlate
Flags: D - DNS, i - dynamic, r - portmap, s - static, I - identity, T - twice
NAT from dmz:10.10.10.128/26 to out:10.10.10.128/26
flags sI idle 0:01:35 timeout 0:00:00

```

Precedence Rules for Network Object NAT

Network Object NAT automatically places the translation rules in Section 2 of the unified NAT table. The NAT statements are ordered by ASA according to the following criteria:

- Static rules always precede dynamic ones. It is important to emphasize that Identity NAT is treated just like any other static NAT instance.
- Within each category (static or dynamic), the ordering guidelines are the following:
 - Number of real IP addresses in the network object: from smallest to largest.
 - For two network objects that have the same quantity of addresses, the IP address is used (from lowest to highest). As an example, 10.10.10.0/24 is placed before 10.10.20.0/24.

- If the same real address is used in two objects, alphabetical ordering is used. For instance, if the objects named A-HOST and B-HOST both refer to the address 10.10.10.10, the NAT rule associated with A-HOST will be checked first.

Example A-8 displays the ordering automatically generated by ASA when the translation rules illustrated in Examples A-2 to A-6 were configured simultaneously:

- Static rules are always placed before dynamic ones.
- Static and Identity rules are ordered from the most specific to the most generic.

Example A-9 shows the reordering that follows from the addition of a /29 network object that has an associated Identity NAT rule.

Example A-8 Precedence Rules (1)

```
ASA# show nat interface dmz detail
Auto NAT Policies (Section 2)
1 (dmz) to (out) source static N10-R148 N16-T148
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.148/32, Translated: 172.16.16.148/32
2 (dmz) to (out) source static N10-HIGH-S28 N18-HIGH-S28
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.128/28, Translated: 172.18.18.128/28
3 (dmz) to (out) source static N10-HIGH-S26 N10-HIGH-S26
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.128/26, Translated: 10.10.10.128/26
4 (dmz) to (out) source dynamic N10-HIGH-S25 N16-RANGE1
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.128/25, Translated: 172.16.16.129-172.16.16.254
5 (dmz) to (out) source dynamic N10-S24 N16-PAT2
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.0/24, Translated: 172.16.16.126/32
```

Example A-9 Precedence Rules (2)

```
! Creating a new Identity NAT rule for subnet 10.10.10.128/29
object network N10-HIGH-S29
    subnet 10.10.10.128 255.255.255.248
    nat (dmz,out) static N10-HIGH-S29
!
ASA# show nat interface dmz detail
Auto NAT Policies (Section 2)
1 (dmz) to (out) source static N10-R148 N16-T148
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.148/32, Translated: 172.16.16.148/32
```

```

2 (dmz) to (out) source static N10-HIGH-S29 N10-HIGH-S29
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.128/29, Translated: 10.10.10.128/29
3 (dmz) to (out) source static N10-HIGH-S28 N18-HIGH-S28
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.128/28, Translated: 172.18.18.128/28
4 (dmz) to (out) source static N10-HIGH-S26 N10-HIGH-S26
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.128/26, Translated: 10.10.10.128/26

```

Twice NAT Examples

Twice NAT facilitates the creation of translation rules that specify source and destination addresses in a single **nat** statement and, as such, is the ideal choice for any category of Policy NAT. Some important characteristics of this method are listed in the following:

- In this type of rule, network objects are parameters of the **nat** commands. This is the opposite behavior of Object NAT, which employs the **nat** statement as a parameter of the network object.
- Twice NAT rules are by default placed in Section 1 of the unified NAT table, thus preceding any Object NAT statement.
- The use of the **after-auto** parameter allows the insertion of a twice NAT rule after the Object NAT statements. This corresponds to section 3.
- Twice NAT rules are inserted in the NAT table in the order you configure them and, in contrast with Object NAT, there is no automatic reordering. This default behavior can be modified by using a sequence number for each entry. This is a powerful resource because it enables the administrator to control exactly the order in which a given rule will be processed.

One noticeable difference in the following examples is the presence of both source and destination information in the Manual NAT policies (twice NAT rules).

Dynamic Policy NAT

In Example A-10, real hosts on the 10.10.10.128/27 subnet have their addresses translated to range 172.17.17.129-158 when they need to communicate with address 172.16.16.200.

Example A-10 *Dynamic Policy NAT*

```

nat (dmz,out) source dynamic N10-HIGH-S27 N17-RANGE1 destination static N16-R200
N16-R200
!
! Ping from 10.10.10.140 to 172.16.16.200
ASA# show xlate type dynamic

```

```

Flags: D - DNS, i - dynamic, r - portmap, s - static, I - identity, T - twice
NAT from dmz:10.10.10.140 to out:172.17.17.133 flags i idle 0:01:09 timeout 3:00:00
!
ASA# show nat interface dmz detail
Manual NAT Policies (Section 1)
1 (dmz) to (out) source dynamic N10-HIGH-S27 N17-RANGE1 destination static N16-
R200 N16-R200
    translate_hits = 1, untranslate_hits = 1
    Source - Origin: 10.10.10.128/27, Translated: 172.17.17.129-172.17.17.158
    Destination - Origin: 172.16.16.200/32, Translated: 172.16.16.200/32

Auto NAT Policies (Section 2)
1 (dmz) to (out) source static N10-R148 N16-T148
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.148/32, Translated: 172.16.16.148/32

```

Dynamic Policy PAT

Example A-11 shows a situation in which hosts belonging to the 10.10.10.128/26 subnet are translated to the PAT address 172.16.16.125 when connecting to 172.16.16.200.

Example A-11 *Dynamic Policy PAT*

```

nat (dmz,out) source dynamic N10-HIGH-S26 N16-PAT1 destination static N16-R200
N16-R200

! Ping from 10.10.10.180 to 172.16.16.200
ASA# show xlate type dynamic
Flags: D - DNS, i - dynamic, r - portmap, s - static, I - identity, T - twice
ICMP PAT from dmz:10.10.10.180/6 to out:172.16.16.125/4839 flags ri idle 0:00:09
timeout 0:00:30
NAT from dmz:10.10.10.140 to out:172.17.17.133 flags i idle 0:12:01 timeout 3:00:00
!
ASA# show nat interface dmz detail
Manual NAT Policies (Section 1)
1 (dmz) to (out) source dynamic N10-HIGH-S26 N16-PAT1 destination static N16-
R200 N16-R200
    translate_hits = 1, untranslate_hits = 1
    Source - Origin: 10.10.10.128/26, Translated: 172.16.16.125/32
    Destination - Origin: 172.16.16.200/32, Translated: 172.16.16.200/32

```

Static Policy NAT

The NAT rule in Example A-12 establishes that the real host 10.10.10.148 should be mapped to 172.18.18.148, when sending packets to 172.16.16.200. This example also shows that, if a sequence number is not specified, a new entry will be inserted immediately after the existent ones. (In this particular case, after the entry defined in Example A-11).

Example A-12 Static Policy NAT

```
! Adding a Static Policy NAT Rule to the previous examples
nat (dmz,out) source static N10-R148 N18-T148 destination static N16-R200 N16-R200
!
! The NAT Rule is inserted in position #2 of Section 1 (Manual NAT)
ASA# show nat interface dmz detail
Manual NAT Policies (Section 1)
1 (dmz) to (out) source dynamic N10-HIGH-S26 N16-PAT1 destination static N16-
R200 N16-R200
    translate_hits = 2, untranslate_hits = 1
    Source - Origin: 10.10.10.128/26, Translated: 172.16.16.125/32
    Destination - Origin: 172.16.16.200/32, Translated: 172.16.16.200/32
2 (dmz) to (out) source static N10-R148 N18-T148 destination static N16-R200 N16-
R200
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.148/32, Translated: 172.18.18.148/32
    Destination - Origin: 172.16.16.200/32, Translated: 172.16.16.200/32

Auto NAT Policies (Section 2)
1 (dmz) to (out) source static N10-R148 N16-T148
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.148/32, Translated: 172.16.16.148/32
!
! Static Policy NAT Rule (Section 1) appears before Static Rule (Section 2)
ASA# show xlate type static
Flags: D - DNS, i - dynamic, r - portmap, s - static, I - identity, T - twice
NAT from dmz:10.10.10.148 to out:172.18.18.148
    flags s idle 0:05:49 timeout 0:00:00
NAT from dmz:10.10.10.148 to out:172.16.16.148
    flags s idle 1:10:20 timeout 0:00:00
```

Example A-13 is aimed at illustrating two operations:

- How to delete a twice NAT statement.
- How to use sequence numbers to control the position in which a **nat** entry will be created. In this example, the dynamic rule of Example A-12 became entry #2.

Example A-13 *Order of NAT Statements in Section 1*

```

! Clearing a Manual NAT statement (entry #2 in Section 1)
ASA(config)# no nat 2
!
! Creating a Static Policy NAT rule with Sequence Number 1

nat (dmz,out) 1 source static N10-R148 N18-T148 destination static N16-R200 N16-
R200
!
ASA# show nat interface dmz
Manual NAT Policies (Section 1)
1 (dmz) to (out) source static N10-R148 N18-T148 destination static N16-R200 N16-
R200
    translate_hits = 0, untranslate_hits = 0
2 (dmz) to (out) source dynamic N10-HIGH-S26 N16-PAT1 destination static N16-
R200 N16-R200
    translate_hits = 2, untranslate_hits = 1

```

Example A-14 shows the usage of the parameter **after-auto** to insert a manual **nat** entry in Section 3 (instead of Section 1). The example also depicts a table containing NAT rules in each of the three possible sections.

Example A-14 *Inserting a Manual NAT Statement in Section 3*

```

! The 'after-auto' parameter indicates that the NAT entry must be created in
Section 3
nat (dmz,out) after-auto source dynamic N10-HIGH-S27 N17-RANGE1 destination static
N16-R200 N16-R200
!
! A sample NAT Table containing 03 sections
ASA# show nat interface dmz detail
Manual NAT Policies (Section 1)
1 (dmz) to (out) source static N10-R148 N18-T148 destination static N16-R200 N16-
R200
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.148/32, Translated: 172.18.18.148/32
    Destination - Origin: 172.16.16.200/32, Translated: 172.16.16.200/32

Auto NAT Policies (Section 2)
1 (dmz) to (out) source static N10-R148 N16-T148
    translate_hits = 0, untranslate_hits = 0
    Source - Origin: 10.10.10.148/32, Translated: 172.16.16.148/32

Manual NAT Policies (Section 3)
1 (dmz) to (out) source dynamic N10-HIGH-S27 N17-RANGE1 destination static N16-
R200 N16-R200
    translate_hits = 0, untranslate_hits = 0

```


Source - Origin: 10.10.10.128/27, Translated: 172.17.17.129-172.17.17.158
 Destination - Origin: 172.16.16.200/32, Translated: 172.16.16.200/32

Dual NAT

Example A-15 illustrates an outbound connection set up from 10.10.10.140 to the real host 172.16.16.201 in a dual NAT environment. Important information associated with this scenario is summarized as follows:

- The real host 10.10.10.140 sends a packet to the mapped address 10.10.10.201.
- The real host 172.16.16.201 (out) sees the packet as sourced from 172.16.16.140.
- The *T* flag reveals that NAT is being performed for both source and destination addresses.

Example A-16 complements Example A-15 by showing the packet tracer tool in action for an inbound connection setup. In this case, the out host 172.16.16.201 contacts the mapped address 172.16.16.140.

Example A-15 Outbound Connection in a Dual NAT Scenario

```
! Creating two new network objects associated with host 172.16.16.201
object network N10-T201
  host 10.10.10.201
object network N16-R201
  host 172.16.16.201
!
nat (dmz,out) 1 source static N10-R140 N16-T140 destination static N10-T201 N16-
R201
!
ASA# show xlate
Flags: D - DNS, i - dynamic, r - portmap, s - static, I - identity, T - twice
NAT from dmz:10.10.10.140 to out:172.16.16.140
  flags sT idle 0:01:41 timeout 0:00:00
NAT from out:172.16.16.201 to dmz:10.10.10.201
  flags sT idle 0:01:41 timeout 0:00:00
!
! Outbound ping from 10.10.10.140 to translated address of out host
(10.10.10.201)
%ASA-6-302020: Built outbound ICMP connection for faddr 172.16.16.201/0 gaddr
172.16.16.140/2 laddr 10.10.10.140/2
ICMP echo request from dmz:10.10.10.140 to out:10.10.10.201 ID=2 seq=0 len=72
ICMP echo request translating dmz:10.10.10.140 to out:172.16.16.140
ICMP echo request untranslating dmz:10.10.10.201 to out:172.16.16.201
%ASA-6-106100: access-list OUT permitted icmp out/172.16.16.201(0) ->
dmz/10.10.10.140(0) hit-cnt 1 first hit [0x8577a13e, 0x0]
```

```
%ASA-6-302020: Built inbound ICMP connection for faddr 172.16.16.201/0 gaddr
172.16.16.140/2 laddr 10.10.10.140/2
ICMP echo reply from out:172.16.16.201 to dmz:172.16.16.140 ID=2 seq=0 len=72
ICMP echo reply translating out:172.16.16.201 to dmz:10.10.10.201
ICMP echo reply untranslating out:172.16.16.140 to dmz:10.10.10.140
!
ASA# show nat interface dmz detail
Manual NAT Policies (Section 1)
1 (dmz) to (out) source static N10-R140 N16-T140 destination static N10-T201 N16-
R201
    translate_hits = 2, untranslate_hits = 2
    Source - Origin: 10.10.10.140/32, Translated: 172.16.16.140/32
    Destination - Origin: 10.10.10.201/32, Translated: 172.16.16.201/32
```

Example A-16 Inbound Connection in a Dual NAT Scenario

```
! Inbound ping from 172.16.16.201 to translated address 172.16.16.140
ASA# packet-tracer input out icmp 172.16.16.201 8 0 1 172.16.16.140
Phase: 1
Type: UN-NAT
Subtype: static
Result: ALLOW
Config:
nat (dmz,out) source static N10-R140 N16-T140 destination static N10-T201 N16-R201
Additional Information:
NAT divert to egress interface dmz
Untranslate 172.16.16.140/0 to 10.10.10.140/0
Phase: 2
Type: ACCESS-LIST
Subtype: log
Result: ALLOW
Config:
access-group OUT in interface out
access-list OUT extended permit icmp any 10.10.10.0 255.255.255.0 log interval 120
Additional Information:
[ output suppressed ]
Phase: 6
Type: NAT
Subtype:
Result: ALLOW
Config:
nat (dmz,out) source static N10-R140 N16-T140 destination static N10-T201 N16-R201
Additional Information:
Static translate 172.16.16.201/1 to 10.10.10.201/1
```

Phase: 7

Type: NAT

Subtype: rpf-check

Result: ALLOW

Config:

nat (dmz,out) source static N10-R140 N16-T140 destination static N10-T201 N16-R201

Additional Information:

Phase: 8

Type: FLOW-CREATION

Subtype:

Result: ALLOW

Config:

Additional Information:

New flow created with id 28, packet dispatched to next module

Note The previous packet tracer simulation reveals that, internally, a single **nat** entry is broken in two separate actions that are performed on distinct phases of the packet's travel through the firewall (phases 2 and 6, in this particular case).

Using Manual NAT to Build Source-Only Rules

Although twice NAT has been designed with source and destination in mind, it can also be used for source-only translation. In Example A-17 some of the NAT entries presented in the Object NAT section are rewritten to illustrate this possibility. The example also shows the use of the **inactive** keyword for disabling a given **nat** statement.

Example A-17 *Building Source-Only Rules with Manual NAT*

```
! Redefining source-only rules from the Object NAT section
ASA# show running-config nat
nat (dmz,out) source static N10-R148 N16-T148
nat (dmz,out) source dynamic N10-HIGH-S25 N16-RANGE1
nat (dmz,out) source dynamic N10-S24 N16-PAT2
!
ASA# show nat interface dmz detail
Manual NAT Policies (Section 1)
1 (dmz) to (out) source static N10-R148 N16-T148
   translate_hits = 0, untranslate_hits = 0
   Source - Origin: 10.10.10.148/32, Translated: 172.16.16.148/32
2 (dmz) to (out) source dynamic N10-HIGH-S25 N16-RANGE1
   translate_hits = 0, untranslate_hits = 0
   Source - Origin: 10.10.10.128/25, Translated: 172.16.16.129-172.16.16.254
3 (dmz) to (out) source dynamic N10-S24 N16-PAT2
```

```

translate_hits = 0, untranslate_hits = 0
Source - Origin: 10.10.10.0/24, Translated: 172.16.16.126/32
!
! Disabling a NAT rule

ASA(config)# nat (dmz,out) source dynamic N10-HIGH-S25 N16-RANGE1 inactive
!
ASA# show nat interface dmz detail | include inactive
2 (dmz) to (out) source dynamic N10-HIGH-S25 N16-RANGE1 inactive

```

Note The `inactive` keyword also works for `nat` statements within a network object.

Summary of NAT Syntaxes in ASA 8.3

As a reference for practical usage, Figure A-3 summarizes the NAT syntaxes introduced in ASA 8.3 code. Before creating any NAT entry, it is advisable to obtain information about the following parameters:

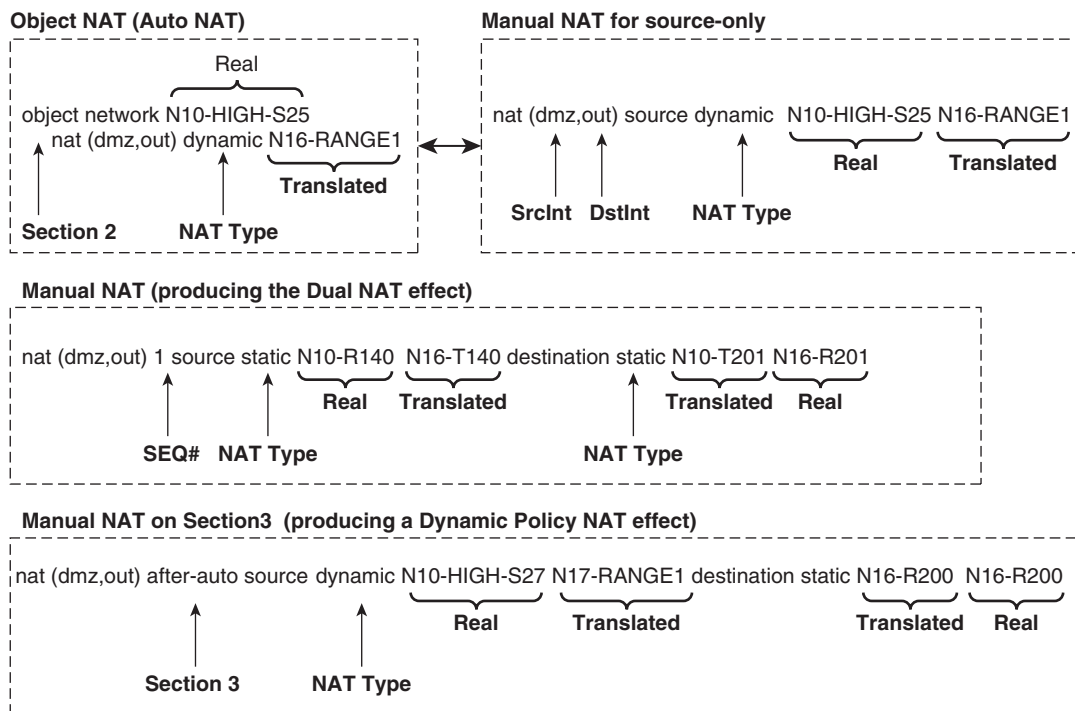


Figure A-3 Summary of NAT Syntax for ASA 8.3

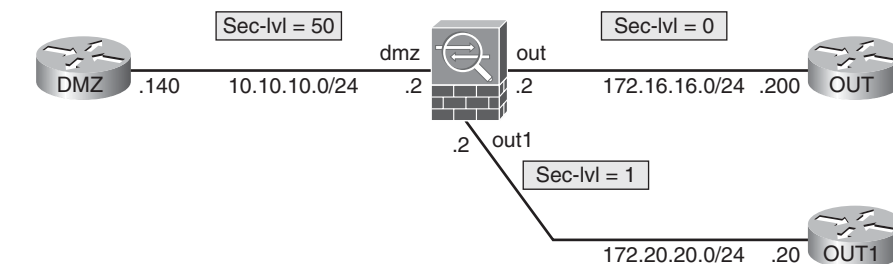
- Source and destination interfaces.
- NAT Type: Is it a static or dynamic rule?
- Is it a source-only rule or the destination needs also to be specified?
- Is it necessary to process this rule in a special order? If this is the case, manual NAT is the most flexible option.

Global ACLs

Another important feature introduced in ASA 8.3 was the capability to define ACLs that apply simultaneously to all interfaces, the so called Global ACLs. The main characteristics of this new category of ACLs follow:

- The ACE construction remains the same.
- This type of ACL is identified by the **global** keyword in the **access-group** command.
- Interface ACLs take precedence over Global ACLs.

Example A-18 refers to the topology in Figure A-4 and illustrates the main features of Global ACLs:



Classic interface-level ACL

```
access-list OUT extended permit icmp any 10.10.10.0 255.255.255.0
access-group OUT in interface out
```

Global ACL

```
access-list GLOBAL extended deny icmp any 10.10.10.0 255.255.255.0
access-list GLOBAL extended permit ip any any
access-group GLOBAL global
```

Figure A-4 Reference Topology for Global ACL Analysis

- The **permit icmp** rule in the interface-level ACL has a higher priority than the **deny icmp** entry in the Global ACL. This is first revealed by the ASP Table.
- Packet tracer shows that packets from 172.16.16.200 to 10.10.10.140 are allowed through. This happens because interface *out* has an interface ACL enabling ICMP.
- Packet tracer demonstrates that ICMP packets arriving on out1, for which there is no interface-level rule, are denied by the Global ACL.

Example A-18 *Global ACL in Action*

```

! Interface-level access-group in the ASP Table
in id=0xd8649008, priority=13, domain=permit, deny=false
    hits=0, user_data=0xd69f7070, cs_id=0x0, use_real_addr, flags=0x0,
    protocol=1
    src ip/id=0.0.0.0, mask=0.0.0.0, port=0
    dst ip/id=10.10.10.0, mask=255.255.255.0, port=0, dscp=0x0
    input_ifc=out, output_ifc=any

! Global access-group in the ASP Table
in id=0xd8330e20, priority=12, domain=permit, deny=true
    hits=0, user_data=0xd69f7110, cs_id=0x0, use_real_addr, flags=0x0,
    protocol=1
    src ip/id=0.0.0.0, mask=0.0.0.0, port=0
    dst ip/id=10.10.10.0, mask=255.255.255.0, port=0, dscp=0x0
    input_ifc=any, output_ifc=any
! Interface-level ACL takes precedence over Global ACL, thus permitting traffic

ASA# packet-tracer input out icmp 172.16.16.200 8 0 1 10.10.10.140
[ output suppressed ]
Phase: 2
Type: ACCESS-LIST
Subtype: log
Result: ALLOW
Config:
access-group OUT in interface out
access-list OUT extended permit icmp any 10.10.10.0 255.255.255.0
!
! Global ACL denies traffic
ASA# packet-tracer input out1 icmp 172.20.20.20 8 0 1 10.10.10.140
[ output suppressed ]
Phase: 2
Type: ACCESS-LIST
Subtype: log
Result: DROP
Config:
access-group GLOBAL global

```

```
access-list GLOBAL extended deny icmp any 10.10.10.0 255.255.255.0  
[ output suppressed ]  
Drop-reason: (acl-drop) Flow is denied by configured rule
```

Note Implicit interface rules also take precedence over global ACLs. For instance, the implicit **permit** on interface dmz has a higher priority than the **deny** statement on the global ACL.

Summary

This appendix reviewed the NAT and ACL changes introduced by ASA 8.3 code, therefore complementing the study presented in Chapters 7 and 8 for pre-8.3 releases.

Index

A

- AAA (Authentication, Authorization, and Accounting), 618, 666-667
 - sequence of activities, 619
 - user-level control, Cut-Through Proxy, 621-634
- AAA servers, 618
- abstraction, 200
- access control, IOS firewalls, 654-662
- Access Control Lists (ACLs). *See* ACLs
- access types, through ASA without NAT, 248-252
- accounting, 618
 - AAA (Authentication, Authorization, and Accounting)
 - sequence of activities*, 619
 - user-level control*, 621-634
- ACEs (Application Control Engines)
 - Auth-Proxy, 638-639
 - Cut-Through Proxy, 625-626
 - virtualized modules, 242-246
- ACLs (Access Control Lists), 90-92, 130
 - antispoofing, 416-417
 - Auth-Proxy, 639-640
 - construction logic, 745
 - Cut-Through Proxy
 - downloadable ACLs*, 629-632
 - locally, defined ACLs*, 627-628
 - fragmented packets, characterized, 90
 - handling, classic IOS firewalls, 338-343
 - IOS, 746
 - IPv6, 743-744, 762-764, 768
 - object, groups*, 769
 - maintenance tasks, 281-284
 - through ASA without NAT, handling, 274-284
 - time-based ACLs, 453-458
 - ZFW (Zone Policy Firewall), 379-391
- active FTP
 - client-side PAT, 517-518
 - server-side Static NAT, 517-518
 - uploading files, 513
- Adaptive Security Appliances (ASA). *See* ASA (Adaptive Security Appliances)
- address publishing
 - NAT exemption, 310-311
 - port redirection, 309-310
 - static command, 308-309
 - through ASA with NAT, 308-311
- addresses, L2, resolving, 736-737
- addressing
 - IPv6, 717-722
 - multicast addressing, 670

adjacency

- broadcast networks, OSPF (Open Shortest Path First), 172
- point-to-point networks, OSPF (Open Shortest Path First), 173-174

admin users, 618**administrative access control**

- ASA (Adaptive Security Algorithm) appliances, 662-665
- IOS firewalls, 654-662

Aggregation Services Routers, 39-41**antispoofing, 416-424**

- ACLs, 416-417
- IPv6, 776-778
- uRPF
 - ASA (*Adaptive Security Algorithm*), 776
 - ASA (*Adaptive Security Algorithm*) appliances, 420-424
 - IOS, 417-420, 776-778

application awareness

- ASA (Adaptive Security Appliances), 773
- stateful firewalls, 14-15

application inspection, 473-474

- ASA (Adaptive Security Appliances), 496-500
 - BTF (botnet traffic filtering)*, 537-544
 - DNS inspection*, 500-512
 - FTP inspection*, 512-523
 - HTTP inspection*, 525-534
 - IM traffic*, 534-536
 - tunneling traffic*, 534-536
- classic IOS firewalls, 474-478
- ZFW (Zone Policy Firewall), 478-479
 - DNS inspection*, 479-480
 - FTP inspection*, 481-485
 - HTTP inspection*, 487-491-493
 - IM inspection*, 494-496

application, level proxies, 1213**architecture, virtualization, 242-246****area ranges, OSPF (Open Shortest Path First), 181-182****ASA (Adaptive Security Appliances), 18, 27-34, 7477**

- administrative access control, 662-665
- antispoofing, uRPF (unicast Reverse Path Forwarding), 420-424
- application awareness, 773
- application inspection, 496-500
 - BTF (botnet traffic filtering)*, 537-544
 - DNS inspection*, 500-512
 - FTP inspection*, 512-523
 - HTTP inspection*, 525-534
 - IM traffic*, 534-536
 - tunneling traffic*, 534-536
- building state, 709
- CLI output filters, 48
- clock synchronization, 74-77
- configuring, 44-55
 - ASA5505 configuration*, 53-55
 - ASA5510 configuration*, 50
- connection limits, 458-463
- DHCP services, 82-86
- direct H.323 calls, 564-567
- enabling, Netflow, 112114
- exec mode, commands, 45
- firewall performance parameters, 29-30
- global unicast addresses, configuring on, 734-736
- H.323 calls through gatekeeper, 569-572
- hardware models, 32-34
- HTTPS access, 63-65
- implicit rules, 249-250
- interface allocation, 224
- interfaces, viewing information, 51
- IOS baseline configuration, 67-70
- IP addresses, obtaining, 77

IPv6

- enabling on*, 733
- stateful inspection*, 770-773
- multicast forwarding, rules, 712-714
- multicast routing, inserting in, 697-712
- NAT (Network Address Translation),
 - ACL changes, 849
- object, groups, IPv6, 766-770
- Packet Capture, embedding, 123-128
- packet tracers, 119-122
- positioning, 28-29
- post, decryption filtering, 826-828
- privileged mode, 47
- remote management access, 60-65
- RP definition, 708
- SCCP (Skinny Client Control Protocol), 556-560
- show version command, 46
- SIP INVITE message, 578-580
- SIP REGISTER messages, 576-578
- SMR (Stub Multicast Routing), 702-707
- SSH access, 62-63
- source registration, 710-712
- summary boot sequence, 44
- syslog messages, 93-94
- TCP normalization, 463-466
- Telnet access, 61-62
- threat detection, 466-470
- through ASA with NAT, 287-288
 - Address Publishing*, 308-311
 - defining connection limits with NAT rules*, 318-320
 - disabling TCP sequence number randomization*, 317-318
 - Dual NAT*, 315-317
 - inbound NAT analysis*, 311-314
 - NAT rules*, 321
 - NAT, control model*, 288-290
 - outbound NAT analysis*, 290, 298-308

- through ASA without NAT, 247-248, 285
 - access types*, 248-252
 - handling ACLs*, 274-284
 - ICMP connections*, 254-258-260
 - same security access*, 272-273
 - security levels*, 253-254
 - TCP connections*, 265-272
 - UDP connections*, 260-265
- transparent firewalls, 196
- upper bound connections, setting, 774-775
- virtual contexts, interconnecting, 235-236

- virtual fragment assembly, 783

ASA Phone, Proxy, voice inspection, 603-616

ASA TLS, Proxy, voice inspection, 596-603

ASDM (Adaptive Security Device Manager)

- HTTPS access, 63-65
- performance monitoring, 114-115

AuditConnection (AUCX) command (MGCP), 591

AuditEndpoint (AUEP) command (MGCP), 591

authentication, 618

- AAA (Authentication, Authorization, and Accounting)
 - sequence of activities*, 619
 - user-level control*, 621-634

- authentication proxy protocols, 618
- protocols, 620-621

- routing protocols, configuring for, 187-190

authentication proxy protocols, 618

authorization, 618

- AAA (Authentication, Authorization, and Accounting)
 - sequence of activities*, 619
 - user-level control*, 621-626-634

Auth-Proxy

CBAC (Context Based Access Control), 642-645

downloadable ACEs, 638-639

downloadable ACLs, 639-640

IOS firewalls, user-level control, 634-645

ZFW (Zone Policy Firewall), 650-653

Auto, RP, 690-697

availability, 3

B

B2BUA (Back-to-Back User Agent), SIP (Session Initiation Protocol), 575

Bacon, Francis, 43

Baselines Security Policy, 4

BGP4 (Border Gateway Protocol, version 4), 139

blocking, instant messengers, 494-496

blocking traffic, BTF (botnet traffic filtering), 543

botnet traffic filtering (BTF). *See* BTF (botnet traffic filtering)

Border Gateway Protocol, version 4 (BGP4), 139

border routers, OSPF (Open Shortest Path First), 186

borderless networks, 845-847

agents, 846

policy management, 846

scanning engines, 846

security intelligence services, 846

bridges, 190

Integrated Routing and Bridging (IRB), 194-195

IOS transparent bridging, 191-193

broadcast networks, adjacency, OSPF (Open Shortest Path First), 172

BTF (botnet traffic filtering)

ASA (Adaptive Security Algorithm) appliances, 537-544

classifying traffic, 541-542

malware database, 538-600

reports, 543-544

statistics, 543-544

traffic, blocking, 543

building state, ASA (Adaptive Security Appliances), 709

C

calculating feasible distance, 1574-158

call signaling protocols, 548

captures, defining, ACLs, 126-127

CBAC (Context Based Access Control), 323, 325-327

Auth-Proxy, 642-645

classic IOS firewalls, 355-359

convergence services, 325

FTP connections, inspecting, 755-757

ICMP connections, inspecting, 754-755

identity services, 325

motivations, 324-325

multiprotocol routing, 324

stateful inspection, 325

TCP connections, inspecting, 751-753

UDP connections, inspecting, 751-753

Chesterton, G.K., 617

circuit-level proxies, 1112

Cisco IP phones, digital certificates, 593-596

Cisco Unified Communication Manager (CUCM), 549

classes, connection limits, 774-775

classic IOS firewalls, 323-324

ACLs, handling, 338-343

application inspection, 474-478

CBAC (Context Based Access Control), 324-327, 355-359

Dual NAT, 351

Dynamic NAT, 349-350

- ICMP connections, 328-331
- IPv6, 751-757
- NAT (Network Address Translation), 343-355-359
 - flow accounting*, 353-355
- Policy NAT, 350-351
- TCP connections, 334-338
- UDP connections, 331, 333-334
- classifying traffic, BTF (botnet traffic filtering), 541-542**
- CLIs (command line interfaces), 44, 115-118**
 - device access, 44
 - graphical interfaces, correlation, 115-118
 - output filters, 48
- client-based access, SSL VPNs, 836-841**
- clientless access, SSL VPNs, 829-836**
- client-side PAT, active FTP, 515-18**
- clock synchronization, NTP, 74-77**
- cloud computing, 845**
- command line interfaces (CLIs).**
See CLIs (command line interfaces)
- commands**
 - ASA (Adaptive Security Appliances), exec mode, 45
 - MGCP (Media Gateway Control Protocol), 591-592
- confidentiality, 3**
- configurable resource types, 229**
- configuration**
 - ASA (Adaptive Security Appliances), 44-55
 - ASA5505 configuration*, 53-55
 - ASA5510 configuration*, 50
 - authentication, routing protocols, 187-190
 - Dual NAT, 315-316
 - EIGRP, 152-166
 - FWSM (Firewall Services Module), 55-60
 - IOS routers, interfaces, 69-70
 - OSPF (Open Shortest Path First), 169-187
 - RIP (Routing Information Protocol), 142-146
- connection limits**
 - ASA (Adaptive Security Appliances), 458-463
 - defining, NAT rules, 318-320
 - IPv6, 774-775
 - ZFW (Zone Policy Firewall), 403-406
- connections, 14**
- connectivity, 846**
 - IPv6, 724-743
- Context Based Access Control (CBAC).** *See* CBAC (Context Based Access Control)
- Control Plane (network traffic), 200**
- convergence services**
 - CBAC (Context Based Access Control), 325
 - Integrated Services Routers (ISR), 20
- Coolidge, Calvin, 547**
- CreateConnection (CRCX) command (MGCP), 592**
- CUCM (Cisco Unified Communication Manager), 549**
 - H.323, 562
 - MGCP (Media Gateway Control Protocol), 585
 - phone registration, 610-611
 - Public Key Infrastructure (PKI), 593-596
- Cut-Through Proxy**
 - AAA (Authentication, Authorization, and Accounting), user-level control, 621-626-634
 - downloadable ACEs, 625-626
 - downloadable ACLs, 629-632
 - HTTP Listener, 632-634
 - locally-defined ACL, 627-628
 - usage scenarios, 622-626

D

DAD (Duplicate Address Detection), ICMP, 725

Data FlowSet (Netflow), 106

Data Plane (network traffic), 200

VLANs (virtual LANs), 201-202

VRFs (Virtual Routing and Forwarding), 202-211

data records (Netflow), 106

databases, RIP (Routing Information Protocol), 143

debug commands, 97-98

dedicated appliances, stateful firewalls, 18

DeleteConnection (DLCX) command (MGCP), 592

Demilitarized Zone (DMZ), 20

Descartes, Rene, 27

destinations, syslog messages, 95-96

device access, CLI (command line interface), 44

device contexts, 18

DHCP services, 82-86

digital certificates, Cisco IP phones, 593-596

direct calls, H.323, 563-567

discontinuous subnets, RIP (Routing Information Protocol), 146

Discovery protocol, transparent firewalls, 197

displaying flow caches, Netflow, 101-102

Distance Vector, 139

distribute-list, RIP (Routing Information Protocol), defining and applying, 149

distributing packets, IP precedence value, 90

DMZ (Demilitarized Zone), 20

DNS doctoring, 505-508

DNS Guard, 502-505

DNS inspection

ASA (Adaptive Security Appliances), 500-512

parameters, 508-511

ZFW (Zone Policy Firewall), 479-480

domains of trust, 5-6

downloading files

active FTP, 513

passive FTP, 514

downstream interfaces, 672-674

Dual NAT, 315316

analyzing, Packet Tracker, 316317

classic IOS firewalls, 351

configuring, 315-316

through ASA without NAT, 315317

Durant, Will, 247

Dynamic NAT, 291293

classic IOS firewalls, 349-350

Dynamic PAT, 293296

inbound NAT analysis, 311313

dynamic routing protocols, VRFs (Virtual Routing and Forwarding), 207-211

E

editing, IOS IPv6 ACLs, 747-748

EGP (Exterior Gateway Protocols), 139

EIGRP (Enhanced Interior Gateway Routing Protocol), 150-170

authentication, 188

configuration, 152-166

DUAL (Diffused Update Algorithm), 151

metric parameters, visualizing, 169-187

metrics, 154-158

QUERY messages, 162-163

redistributing routes, 158-161

REPLY messages, 162-163

stub operation, 164-166

summary routes, 161-162

VRFs (Virtual Routing and Forwarding), 208-210
 Eisenhower, Dwight D., 1
 Eliot, Thomas S.199
 E-mail Security Appliance (ESA), 25
 embryonic connections, limiting, per-client basis, 775
 Endpoint Configuration (EPCF) command (MGCP), 592
 ESA (E-mail Security Appliance), 25
 event logging, 92-96
 exec mode (ASA), commands, 45
 exemption, NAT (Network Address Translation), 303-304
 address publishing, 310-311
 inbound NAT analysis, 314
 Exterior Gateway Protocols (EGP), 139
 external firewalls, VMs (virtual machines), 802-803
 extranet, topologies, 254

F

feasible distance, calculating, 157-158
 fields, IPv6 headers, 722-723
 files
 downloading
 active FTP, 513
 passive FTP, 514
 uploading, active FTP, 513
 filtering
 FTP file types, 523
 FTP request commands, 521-522
 IPv6 packets, 745
 post, decryption filtering, ASA (Adaptive Security Algorithm), 826-828
 TCP flags, 425-429
 TTL (Time-to-Live) value, 429-430
 Firewall Services Module (FWSM), 20, 36-38

firewalls, 2

application-level proxies, 12-13
 circuit-level proxies, 11-12
 domains of trust, 5-6
 Firewall Services Module (FWSM), 36-38
 IPS (intrusion prevention systems), 788-793
 IPv6, tunneling mechanisms, 806-812
 load balancing, 798
 MPLS networks, 841-845
 network topologies, 133-134
 IP routing, 134-135
 RIP (Routing Information Protocol), 140-150
 routing protocols, 138-140
 static routing, 135-138
 network topology, insertion, 6-10
 packet filters, 10
 performance parameters, ASA (Adaptive Security Algorithm) appliances, 29-30
 PVLANS (Private VLANs), 794-796
 QoS (Quality of Service), 793
 rules, 2-5
 Security Policies, 32
 security principles, 3
 Security Wheel, 4-5
 SLB (Server Load Balancers), 796-798, 796-801
 stateful firewalls, 13-14
 application awareness, 14-15
 dedicated appliances, 18
 identity awareness, 15
 intrusion prevention, 23-24
 IOS router, based firewalls, 38
 network segmentation, 17-18
 router, based firewalls, 18-20
 routing table for protection tasks, 16-17
 security design, 21-25

- specialized security appliances*, 25
- switch, based firewalls*, 20
- topologies*, 20
- unicast Reverse Path Forwarding (uRPF)*, 16-17
- virtualization*, 17-18
- VPNs, 22
 - transparent firewalls, 196-197
 - virtual firewalls, 17-18
 - VMs (virtual machines), 801-806
 - external firewalls*, 802-803
 - virtual firewall appliances*, 803-806
 - VPNs (virtual private networks)
 - IPsec*, 812-828
 - SSL VPNs*, 828-841
- flags (ASA), TCP connections, 265-267
- Flexible Netflow, 105112
- flexible packet matching, 448-453
- flow accounting, NAT (Network Address Translation), 353-355
- flow caches, Netflow, displaying, 101-102
- flow collection, Netflow, IOS, 10090
- Flow Label field (IPv6 header), 722
- formats, IPv6 headers, 722-724
- Forwarding Plane (network traffic), 200
- Fragment Offset field (Fragmentation header), 779
- fragmentation, IPv6, 778-784
- fragmented packets, ACLs, 90
- FTP (File Transport Protocol)
 - active FTP
 - client-side PAT*, 517518
 - downloading files*, 513
 - server-side Static NAT*, 513-518
 - uploading files*, 513
 - file types, filtering
 - passive FTP, 523
 - downloading*, 514
 - server-side Static NAT*, 515-516
 - FTP connections, inspecting, CBAC (Context Based Access Control), 755-757
 - FTP inspection
 - ASA (Adaptive Security Appliances), 512-523
 - ZFW (Zone Policy Firewall), 481-485
 - FTP over IPv6, ZFW (Zone Policy Firewall), 764-766
 - FTP request commands, filtering, 521-522
 - functional planes, network traffic, 200
 - FWSM (Firewall Services Module), 20, 36-38
 - clock synchronization, 74-77
 - configuring, 55-60
 - DHCP services, 82-86
 - HTTPS access, 63-65
 - IOS baseline configuration, 67-70
 - IP addresses, obtaining, 77
 - packet classification, 235-236
 - remote management access, 60-65
 - SSH access, 62-63
 - syslog messages, 94
 - Telnet access, 61-62
 - virtual contexts, interconnecting, 234-238
 - virtualized modules, 242-246

G

- gatekeepers, H.323, 561
 - calls, 567-572
- gateway registration, MGCP (Media Gateway Control Protocol), 586-589
- gateways, H.323, 561
- global unicast addresses
 - ASA (Adaptive Security Appliances), configuring on, 734-736
 - IOS, configuring, 729
- graphical interfaces, CLI (command line interface), correlation, 1154118

GRE tunnel, site-to-site IPsec, 822-823

Guidelines Security Policy, 3

H

H.323, 560-572

CUCM (Cisco Unified Communication Manager), 562

direct calls, 563-567

gatekeepers, 561

calls, 567-572

gateways, 561

MCUs (Multipoint Control Units), 561

terminals, 561

hardware models, ASA (Adaptive Security Algorithm) appliances, 32-34

headers, IPv6, format, 722-724

Hooker, Richard, 715

Hop Limit field (IPv6 header), 722

HTTP (Hyper-Text Transfer Protocol)

connections, ZFW (Zone Policy Firewall), 762

IOS devices, remote management access, 73-74

HTTP inspection

ASA (Adaptive Security Appliances), 525-534

ZFW (Zone Policy Firewall), 487-493

HTTP Listener, Cut-Through Proxy, 632-634

HTTPS

access, ASDM (Adaptive Security Device Manager), 63-65

IOS devices, remote management access, 73-74

classic IOS firewalls, 328-331

inbound ping, 257

outbound ping, 255-257

through ASA without NAT, 258-260

ZFW (Zone Policy Firewall), 370-373, 758-760

Huxley, Aldous, 361

I-L

ICMP (Internet Control Message Protocol), 14

connections, 257

DAD (Duplicate Address Detection), 725

inspecting, CBAC (Context Based Access Control), 754-755

Layer 2 Address Resolution (ARP Replacement), 724

ND (Neighbor Discovery) messages, 725

Router Advertisements (RA) messages, 724

Router Solicitation (RS) messages, 724

Identification field (Fragmentation header), 779

identity services, Integrated Services Routers (ISR), 19

identity, 617

identity awareness, stateful firewalls, 15

Identity NAT, 296-298

inbound NAT analysis, 313-314

identity services, CBAC (Context Based Access Control), 325

IGP (Interior Gateway Protocols), 139

IM inspection, ZFW (Zone Policy Firewall), 494-496

IM traffic, ASA (Adaptive Security Appliances), inspection, 534-536

implicit rules, ASA (Adaptive Security Appliances), 249-250

inbound access, ASA (Adaptive Security Appliances), 248

inbound NAT analysis, 311-314

inbound ping, ICMP connections, 257

inline SLB, 796

instant messengers, blocking, 494-496
Integrated Routing and Bridging (IRB), 194-195

Integrated Services Routers (ISR), 19-20, 38

integrity, 3

Intelligent Networks, 1

interconnecting, 238-241

virtual contexts, 232-241

ASA (Adaptive Security Appliances), 238-241

external router, 233

FWSM (Firewall Services Module), 234-238

unshared interfaces, 233

interface allocation, ASA (Adaptive Security Appliances), 224

interfaces

ASA (Adaptive Security Appliances),
viewing information, 51

downstream interfaces, 672-674

IOS routers, configuring, 69-70

RPF (Reverse Path Forwarding)
interface, 674-676

upstream interfaces, 672-674

Interior Gateway Protocol (IGP), 139

internal departments, isolating, 254

Internet Access, topology, 254

intrazone firewall policies, ZFW (Zone Policy Firewall), 410-413

intrusion prevention, stateful firewalls, 23-24

intrusion prevention systems (IPS), 788-793

IOS (Internetwork Operating System)

ACLs, 746

editing, 747-748

handling, 743-750

antispoofing, uRPF (unicast Reverse Path Forwarding), 417-420

command authorization, 657-658

debug options, 97-98

global unicast addresses, configuring, 729

membership awareness, 645-649

Packet Capture, embedding on, 128-129

site-to-site IPsec, 813-818

uRPF (unicast Reverse Path Forwarding), antispoofing, 776-778
virtual fragment assembly, 783-784

IOS CLI, baseline configuration, 67-70

IOS devices, remote management access, 70-74

HTTP/HTTPS, 73-74

SSH, 71-73

Telnet, 70-71

IOS firewalls, 323-324

ACLs, handling, 338-343

administrative access control, 654-662

application inspection, 474-478

CBAC (Context Based Access Control), 324-325-327, 355-359

Dual NAT, 351

Dynamic NAT, 349-350

ICMP connections, 328-331

IPv6, 751-757

NAT (Network Address Translation), 343-359

flow accounting, 353-355

Policy NAT, 350-351

TCP connections, 334-338

UDP connections, 331-334

user-level control, Auth-Proxy, 634-645

IOS flow collection, Netflow, 100-90

IOS router-based firewalls, 38

ASR (Aggregation Services Routers), 39-41

Integrated Services Routers (ISR), 38

IOS routers

interfaces, configuring, 69-70

IPv6, enabling on, 727-728

- show version command, 68
- summary boot sequence, 67
- IOS syslog messages, 93**
- IOS traceroute, through ASA without NAT, 261-265**
- IOS transparent bridging, 191-193**
- IOS ZFW (Zone Policy Firewall), 361-370, 414**
 - ACLs, 379-391
 - application inspection, 478-479
 - DNS inspection, 479-480*
 - FTP inspection, 481-485*
 - HTTP inspection, 487-493*
 - IM inspection, 494-496*
 - Auth-Proxy, 650-653
 - connection limits, defining, 403-406
 - FTP over IPv6, 764-766
 - HTTP connections, 762
 - ICMP connections, 370-373, 758-760
 - intrazone firewall policies, 410-413
 - IPv6, 757-766
 - ACLs, 762-764*
 - membership awareness, 645-649
 - NAT (Network Address Translation), 391-400
 - parameter, maps, 758
 - router traffic, inspecting, 407-410
 - TCP connections, 377-379
 - Telnet connections, 761-762
 - transparent mode, 400-403
 - UDP connections, 373-377, 760
- IP fragmentation, 439-448**
 - stateless filtering in IOS, 443-445
 - virtual reassembly on ASA, 446-448
 - virtual reassembly on IOS, 445-446
- IP multicasting, 669**
 - multicast addressing, 670
 - multicast forwarding, 671-676
 - multicast routing, 671-676
- IP options**
 - drop on ASA, 437-438
 - drop on IOS, 437-438
- handling, 430-439**
- stateless filtering on IOS, 434-437**
- IP phones, digital certificates, 593-596**
- IP precedence value, packet distribution, 90**
- IP routing, 134-135**
- IP spoofing. *See* antispoofing**
- IPS (intrusion prevention systems), 788-793**
- IPsec**
 - site-to-site
 - GRE tunnel, 822-823*
 - IOS, 813-818*
 - VTIs (Virtual Tunneling Interfaces), 818-821*
 - tunnels, NAT (Network Address Translation), 823-826
 - VPNs (virtual private networks), 812-828
- IPv6, 716-717, 785**
 - ACLs, 743-744, 768
 - object, groups, 769*
 - addresses, obtaining, 77
 - addressing, 717-721-722
 - antispoofing, 776-778
 - ASA, stateful inspection, 770-773
 - ASA (Adaptive Security Appliances), enabling on, 733
 - changes to, 716-717
 - classic IOS firewalls, 751-757
 - connection limits, 774-775
 - connectivity, 724-743
 - firewalls, tunneling mechanisms, 806-812
 - fragmentation, 778-784
 - header format, 722-724

- IOS ACLs (Access Control Lists)
 - editing*, 747-748
 - handling*, 743-750
- IOS routers, enabling on, 727-728
- L2 addresses, resolving, 736-737
- netflow, 739-742
- packets, filtering, 745
- static routes, configuring, 737-739
- traffic statistics, viewing, 742-743
- ZFW (Zone Policy Firewall), 757-766
 - ACLs*, 762-764
- IRB (Integrated Routing and Bridging), 194-195
- ISR (Integrated Services Router), 19-20, 38
- Java applets, HTTP response body, removing from, 531-532
- Jung, Carl, 787
- L2 addresses, resolving, 736-737
- LANs, virtual LANs (VLANs), 17
- Layer 2 Address Resolution (ARP Replacement), ICMP, 724
- Link State, 139
- load balancing, firewalls, 798
- locally-defined ACL, Cut-Through Proxy, 627-628
- logging levels, syslog messages, 95-96
- LSDB (Link State Database), OSPF (Open Shortest Path First), 174-175

M

- M bit field (Fragmentation header), 779
- maintenance tasks, ACLs, 281-284
- malware database, BTF (botnet traffic filtering), 538-600
- management access, virtual contexts, 225-227
- Management Plane (network traffic), 200
- management tasks, virtual contexts, 218-220

- matching-set*, cookie response headers, 528
- MCUs (Multipoint Control Units), H.323, 561
- Media Gateway Control Protocol (MGCP). *See* MGCP (Media Gateway Control Protocol)
- membership awareness, IOS, 645-649
- messages, ICMP, 724
- metric parameters, EIGRP, 154-158, 169-187
- MGCP (Media Gateway Control Protocol), 549, 584-592
 - commands, 591-592
 - CUCM (Cisco Unified Communication Manager), 585
 - gateway registration, 586-589
- mobility, 845
- ModifyConnection (MDCX) command (MGCP), 592
- monitoring
 - EIGRP, 152-166
 - OSPF (Open Shortest Path First), 169-187
 - RIP (Routing Information Protocol), 142-146
- MPLS networks, firewalls, 841-845
- multicast addressing, 670, 721
- multicast forwarding, 671-676
 - ASA (Adaptive Security Appliances), rules, 712-714
- multicast routing, 671-676
 - ASA (Adaptive Security Appliances), inserting in, 697-712
 - PIM (Protocol Independent Multicast), 676-697
- multicasting, 669
 - multicast addressing, 670
 - multicast forwarding, 671-676
 - multicast routing, 671-676
 - ASA (*Adaptive Security Appliances*), 697-712
 - PIM (*Protocol Independent Multicast*), 676-678, 697

multiple mode operation, virtual contexts, 214

multiprotocol routing

CBAC (Context Based Access Control), 324

Integrated Services Routers (ISR), 19

N

NASes (Network Access Servers), 618, 654

NAT (Network Address Translation), 8-10, 716-717

ASA (Adaptive Security Appliances), ACL changes, 849

classic IOS firewalls, 343, 355-359

Dual NAT, 315-316

Dynamic NAT, 291-293

classic IOS firewalls, 349-350

exemption, 303-304

address publishing, 310-311

inbound NAT analysis, 314

flow accounting, 353-355

Identity NAT, 296-298

inbound NAT analysis, 313-314

IPsec tunnels, 823-826

Policy NAT, 299-303

precedence rules, 304-308

Static NAT, 298-299

classic IOS firewalls, 346-344

inbound NAT analysis, 314

through ASA with NAT, 287-288

Address Publishing, 308-311

inbound NAT analysis, 311-314

NAT, control model, 288-290

ZFW (Zone Policy Firewall), 391-400

ND (Neighbor Discovery) messages, 725

Netflow, 98-100, 131

ASA (Adaptive Security Appliances), enabling, 112-114

Flexible Netflow, 105-112

flow caches, displaying, 101-102

flow collection, IOS, 100-90

flow expert, 104

NSEL (Netflow Security Event Logging), 112-114

source IP traceback, 102

traditional Netflow, 100-105

traditional records, 99

v9, 105-112

Network Access Servers (NASes), 618

Network Address Translation (NAT). See NAT (Network Address Translation)

Network Device Groups (NGDs), 654

network object, groups, 275

network segmentation, stateful firewalls, 17-18

network topologies

firewalls, 133-134

insertion, 6-10

IP routing, 134-135

RIP (Routing Information Protocol), 140-150

routing protocols, 138-140

static routing, 135-138

stateful firewalls, 20

network traffic, functional planes, 200

Newton, Isaac, 473

Next Header field (Fragmentation header), 779

Next Header field (IPv6 header), 722

NGDs (Network Device Groups), 654

NotificationRequest (RQNT) command (MGCP), 592

Notify (NTFY) command (MGCP), 592

NSEL (Netflow Security Event Logging), 112-114

NTP, clock synchronization, 74-77

O

object, groups

ASA (Adaptive Security Appliances),
766-770

IPv6 ACLs, 769

one-arm SLB, 797

OSPF (Open Shortest Path First), 167-187

area ranges, 181-182

authentication, 189

border routers, 186

broadcast networks, adjacency, 172

configuring, 169-187

AS, external LSAs, 184-185

LSA type-3 filtering, 182-183

LSDB (Link State Database), 174-175

point-to-point networks, adjacency,
173-174

router LSAs, 175-176

routing table, 174-175

summary LSAs, 180

transparent firewalls, 196

VRFs (Virtual Routing and
Forwarding), 207-208

outbound access, ASA (Adaptive Security Appliances), 248

outbound IOS traceroute, through ASA without NAT, 261-265

outbound NAT analysis, 290-308

outbound ping, ICMP connections, 255-257

output filters, CLI (command line interface), 48

P-Q

Packet Capture, 118, 122-130

ASA (Adaptive Security Appliances),
embedding on, 123-128

defining captures, 126-127

IOS, embedding on, 128-129

Packet Tracer, integrating, 125-126

packet classification, FWSM (Firewall
Services Module), 235-236

packet filters, 10

Packet Tracer, 131

Packet Capture, 125-126

Packet Tracker, Dual NAT, 316-317

packets

distribution, IP precedence value, 90

IPv6, filtering, 745

parameter-maps, ZFW (Zone Policy
Firewall), 758

partitioning, virtualization, 201

passive FTP, server-side Static NAT,
515-516

PAT (Port Address Translation), 810

Dynamic PAT, 293-296

inbound NAT analysis, 311-313

Payload Length field (IPv6 header), 722

performance monitoring, ASDM
(Adaptive Security Device Manager),
114-115

performance parameters, firewalls,
ASA (Adaptive Security Appliances),
29-30

phone registration, CUCM (Cisco
Unified Communication Manager),
610-611

PIM (Protocol Independent Multicast)

Cisco routers, enabling on, 677-678

multicast routing, 676-697

PIM, DM, 678-680

PIM, SM, 680-689

PIM, DM, 678-680

PIM, SM, 680-689

join process, 683-684

rendezvous points, 690-697

source registering, 686

PIX firewall, 326

PKI (Public Key Infrastructure),
CUCM (Cisco Unified Communication
Manager), 593-596

point-to-point networks, adjacency, OSPF (Open Shortest Path First), 173-174

Policies (Security), firewalls, 2-3

policy management, borderless networks, 846

Policy NAT, 299303

- classic IOS firewalls, 350-351

pooling, 201

Port Address Translation (PAT), 810

port redirection, address publishing, 309-310

positioning, ASA (Adaptive Security Algorithm) appliances, 28-29

post-decryption filtering, ASA (Adaptive Security Appliances), 826-828

PPP session negotiation, 80

PPPoE, baseline configuration, 78-79

PPPoE clients, IP addresses, obtaining, 77-78

precedence rules, NAT (Network Address Translation), 304-308

prefix advertisement, 731-733

presence servers, SIP (Session Initiation Protocol), 575

Private VLANs (PVLANS), 794-796

privileged mode, ASA (Adaptive Security Appliances), 47

Procedures Security Policy, 4

protocols, routing protocols, 138-140

proxy servers, SIP (Session Initiation Protocol), 575

PSTN (Public Switched Telephone Network), 549

Public Key Infrastructure (PKI), CUCM (Cisco Unified Communication Manager), 593-596

Public Switched Telephone Network (PSTN), 549

PVLANS (Private VLANs), 794

QoS (Quality of Service), firewalls, 793

QUERY messages, EIGRP, 162163

R

RADIUS, versus TACACS+, 620621

Real-Time Transport Protocol (RTP), 549

records, Netflow, 99

redirect servers, SIP (Session Initiation Protocol), 575

redistributing routes, EIGRP, 158161

reference netflow, IPv6, 739-742

Registrar Server, SIP (Session Initiation Protocol), 575

registration process, SCCP (Skinny Client Control Protocol), 553-554

regular users, 618

remote management access, 60-65

- IOS devices, 70-74
- HTTP/HTTPS*, 73-74
- SSH*, 71-73
- Telnet*, 70-71

rendezvous points, PIM, SM, 690-697

REPLY messages, EIGRP, 162163

reported distance, feasible distance, calculating, 157-158

reports, BTF (botnet traffic filtering), 543-544

Replier, Agnes, 669

Res field (Fragmentation header), 779

Reserved field (Fragmentation header), 779

resource classes, 230-232

resource types, configurable, 229

resources, virtual contexts, allocating to, 228-232

RestartInProgress (RSIP) command (MGCP), 592

Reverse Path Forwarding) RPF interface, 674-676

RFCs, SIP (Session Initiation Protocol), 573-574

RIP (Routing Information Protocol), 139, 140-150

- authentication, 188
- configuring, 142-146
- database, 143
- discontinuous subnets, 146
- distribute-list, defining and applying, 149
- monitoring, 142-146
- originating default route, 148
- routing table, 143
- summary routes, generating, 148
- timers, 144
- updates, 143
- version control, 144
- VLSM, 145
- VRFs (Virtual Routing and Forwarding), 210-211

Routed mode, versus Transparent mode, 7

Router Advertisements (RA) messages, 724

router LSAs, OSPF (Open Shortest Path First), 175-176

Router Solicitation (RS) messages, 724

router traffic, ZFW (Zone Policy Firewall), 407-410

router-based firewalls, 18-20

router-based stateful firewalls, 38

- ASR (Aggregation Services Routers), 39-41

- Integrated Services Routers (ISR), 38

routers

- ASR (Aggregation Services Routers), 39-41

- ISR (Integrated Services Routers), 38

- PIM (Protocol Independent Multicast), enabling on, 677-678

routes, redistributing, EIGRP, 158-161

Routing Information Protocol (RIP).

See **RIP (Routing Information Protocol), 139**

routing protocols, 138-140

- authentication, configuring for, 187-190

- EIGRP, 150-170

- OSPF (Open Shortest Path First), 167-187

- RIP (Routing Information Protocol), 142-150

routing tables

OSPF (Open Shortest Path First), 174-175

RIP (Routing Information Protocol), 143

RP definition, ASA (Adaptive Security Appliances), 708

RPF (Reverse Path Forwarding) interface, 674-676

RTP (Real-Time Transport Protocol), 549

rules, firewalls, 2-5

S

same security access, through ASA without NAT, 272-273

Santayana, George, 324

scanning engines, borderless networks, 846

SCCP (Skinny Client Control Protocol), 549, 550-560

- ASA (Adaptive Security Algorithm) appliances, 556-560

- baseline configuration, 552

- registration process, 553-554

security

- ACLs, time-based ACLs, 453-458

- antispoofing, 416-424

- ASA (Adaptive Security Appliances)

- connection limits, 458-463*

- TCP normalization, 463-466*

- threat detection, 466-470*

- flexible packet matching, 448-453

- IP Options, handling, 430-439

- TCP flags, filtering, 425-429
- TTL (Time-to-Live) value, filtering on, 429-430
- security access, ASA (Adaptive Security Appliances), 248
- security contexts, 241
- security intelligence services, borderless networks, 846
- security levels, ASA (Adaptive Security Appliances), 253-254
- Security Policies, firewalls, 2-3
- Security Wheel, 4-5
- sequence number randomization, TCP connections, 267-272
- Server Load Balancers (SLBs). *See* SLBs (Server Load Balancers)
- server-side Static NAT
 - active FTP, 517-518
 - passive FTP, 515-516
- service object, groups, 275, 276
- Session Initiation Protocol (SIP). *See* SIP (Session Initiation Protocol)
- session negotiation (PPP), 80
- set-cookie response headers, matching, 528
- Shakespeare, William, 415
- shell command authorization sets, 654
- show command, 665-666
- show version command
 - ASA (Adaptive Security Appliances), 46
 - IOS routers, 68
- single mode operation, virtual contexts, 213
- SIP (Session Initiation Protocol), 549, 573-583
 - B2BUA (Back-to-Back User Agent), 575
 - presence servers, 575
 - proxy servers, 575
 - redirect servers, 575
 - Registrar Server, 575
 - RFCs, 573-574
 - user agents, 575
- site-to-site IPsec
 - GRE tunnel, 822-823
 - IOS, 813-818
 - VTIs (Virtual Tunneling Interfaces), 818-821
- Skinnny Client Control Protocol (SCCP). *See* SCCP (Skinnny Client Control Protocol)
- SLBs (Server Load Balancers), 796-801
 - inline SLB, 796
 - one-arm SLB, 797
- SMR (Stub Multicast Routing), ASA (Adaptive Security Appliances), 702-707
- Source and Destination Addresses field (IPv6 header), 722
- source IP traceback, Netflow, 102
- specialized security appliances, stateful firewalls, 25
- SSH (Secure Shell)
 - access, 62-63
 - IOS devices, remote management access, 71-73
- SSL VPNs, 828-841
 - client-based access, 836-841
 - clientless access, 829-836
- Standards Security Policy, 3
- stateful firewalls, 13-14
 - application awareness, 14-15
 - dedicated appliances, 18
 - identity awareness, 15
 - intrusion prevention, 23-24
 - IOS router-based firewalls, 38
 - ASR (Aggregation Services Routers), 39-41
 - ISR (Integrated Services Router), 38
 - network segmentation, 17-18
 - router-based firewalls, 18-20
 - routing table for protection tasks, 16-17

- security design, 21-25
- specialized security appliances, 25
- switch-based firewalls, 20
- topologies, 20
- unicast Reverse Path Forwarding (uRPF), 16-17
- virtualization, 17-18
- VPNs, 22
- stateful inspection**
 - CBAC (Context Based Access Control), 325
 - IPv6, ASA (Adaptive Security Algorithm), 770-773
- stateless autoconfiguration, 731-733**
- stateless filtering on IOS**
 - IP fragmentation, 443-445
 - IP options, 434-437
- static command, address publishing, 308-309**
- Static NAT, 298-299**
 - classic IOS firewalls, 346-344
 - inbound NAT analysis, 314
- static routes**
 - configuring, 737-739
 - VRFs (Virtual Routing and Forwarding), 205-206
- static routing, 135-138**
- statistics, BTF (botnet traffic filtering), 543-544**
- Stub Multicast Routing (SMR), ASA (Adaptive Security Appliances), 702-707**
- stub operation, EIGRP, 164-166**
- summary boot sequence**
 - ASA (Adaptive Security Appliances), 44
 - IOS routers, 67
- summary LSAs, OSPF (Open Shortest Path First), 180**
- summary routes, EIGRP, 161-162**
- switch-based firewalls, 20**
- syslog messages**
 - ASA (Adaptive Security Appliances), 93-94

- destinations, 95-96
- FWSM (Firewall Services Module), 94
- IOS, 93
- logging levels, 95-96

T

- TACACS+, versus RADIUS, 620-621**
- TCP (Transport Control Protocol), 14**
- TCP connections**
 - classic IOS firewalls, 334-338
 - inspecting, CBAC (Context Based Access Control), 751-753
 - sequence number randomization, 267-272
 - through ASA without NAT, 265-272
 - ZPF (Zone Policy Firewall), 377-379
- TCP flags, filtering, 425-429**
- TCP normalization, ASA (Adaptive Security Algorithm) appliances, 463-466**
- TCP sequence number randomization, disabling, 317-318**
- Telnet**
 - access, 61-62
 - connections, ZFW (Zone Policy Firewall), 761-762
 - IOS devices, remote management access, 70-71
- Template FlowSet (Netflow), 105**
- template IDs (Netflow), 106**
- template records (Netflow), 105**
- terminals, H.323, 561**
- threat detection, ASA (Adaptive Security Algorithm) appliances, 466-470**
- through ASA with NAT, 287-288, 321**
 - Address Publishing, 308-311
 - connection limits, defining with NAT rules, 318-320
 - Dual NAT, 315-317
 - inbound NAT analysis, 311-314

- NAT, control model, 288-290
- outbound NAT analysis, 290-308
- TCP sequence number randomization, 317-318
- through ASA without NAT, 247-248, 285
 - access types, 248-252
 - ACLs, handling, 274-284
 - ICMP connections, 258-260
 - same security access, 272-273
 - security levels, 253-254
 - TCP connections, 265-272
 - UDP connections, 260-265
 - Windows Traceroute, 258-260
- time-based ACLs, 453-458
- timers, RIP (Routing Information Protocol), 144
- topologies
 - extranet, 254
 - firewalls, 133-134
 - IP routing*, 134-135
 - RIP (Routing Information Protocol)*, 140-150
 - routing protocols*, 138-140
 - static routing*, 135-138
 - Internet Access, 254
 - stateful firewalls, 20
- traditional Netflow, 100-105
- traffic
 - blocking, BTF (botnet traffic filtering), 543
 - classifying, BTF (botnet traffic filtering), 541-542
 - functional planes, 200
 - classic IOS firewalls*, 331-334
 - inspecting, CBAC (Context Based Access Control)*, 751-753
 - through ASA without NAT*, 260-265
 - ZFW (Zone Policy Firewall)*, 373-377, 760

- Traffic Class field (IPv6 header), 722
- traffic statistics, IPv6, viewing, 742-743
- transparent firewalls, 196-197
- Transparent Mode, versus Routed mode, 7
- transparent mode, ZFW (Zone Policy Firewall), 400-403
- transparent virtual contexts, 221-224
- Transport Control Protocol (TCP), 14
- TTL (Time-to-Live) value, filtering on, 429-430
- tunneling mechanisms, firewalls, IPv6, 806-812
- tunneling traffic, ASA (Adaptive Security Appliances), inspection, 534-536
- tunnels, IPsec, NAT (Network Address Translation), 823-826

U

- unicast Reverse Path Forwarding (uRPF). *See* uRPF (unicast Reverse Path Forwarding)
- updates, RIP (Routing Information Protocol), 143
- upper bound connections, ASA (Adaptive Security Appliances), setting, 774-775
- upstream interfaces, 672-674
- uRPF (unicast Reverse Path Forwarding)
 - antispoofing
 - ASA (Adaptive Security Appliances)*, 420-424, 776
 - IOS*, 417-420, 776-778
 - stateful firewalls, 16-17
- user agents, SIP (Session Initiation Protocol), 575
- user-agent matching, 529-531
- user-group membership awareness, IOS, 645-649

user-level control

AAA (Authentication, Authorization, and Accounting), Cut-Through Proxy, 621-634

IOS firewalls, Auth-Proxy, 634-645

users

admin users, 618

regular users, 618

V

version control, RIP (Routing Information Protocol), 144

Version field (IPv6 header), 722

virtual contexts

allocating resources to, 228-232

interconnecting, 232-241

ASA (Adaptive Security Appliances), 238-241

external router, 233

FWSM (Firewall Services Module), 234-238

unshared interfaces, 233

management access, 225-227

management tasks, 218-220

multiple mode operation, 214

single mode operation, 213

transparent, 221-224

VRFs (Virtual Routing and Forwarding), 212-225

virtual firewalls, 17-18

VMs (virtual machines), 803-806

virtual fragment assembly

ASA (Adaptive Security Algorithm), 783

IOS, 783-784

virtual LANs (VLANs). *See* VLANs (virtual LANs)

virtual machines (VMs). *See* VMs (virtual machines)

Virtual Routing and Forwarding (VRFs). *See* VRFs (Virtual Routing and Forwarding)

virtualization, 199-200

abstraction, 200

architecture, 242-246

partitioning, 201

pooling, 201

stateful firewalls, 1748

virtual contexts

allocating resources to, 228-232

interconnecting, 232-241

management access, 225-227

management tasks, 218-220

multiple mode operation, 214

single mode operation, 213

transparent, 221-224

VLANs (virtual LANs), Data Plane, 201-202

VRFs (Virtual Routing and Forwarding), Data Plane, 202-211

VLANs (virtual LANs), 17

Data Plane, 201-202

VLSM (Variable Length Subnet Mask), RIP (Routing Information Protocol), 145

VMs (virtual machines), firewalls, 801-806

external firewalls, 802-803

virtual firewall appliances, 803-806

Voice over IP (VoIP), 549**voice protocols, 547-548-549**

ASA Phone, Proxy, voice inspection, 603-616

ASA TLS, Proxy, voice inspection, 596-603

call signaling protocols, 548

H.323, 562-572

H.323 standard, 560-561

MGCP (Media Gateway Control Protocol), 584-592

SCCP (Skinny Client Control Protocol), 550-560

SIP (Session Initiation Protocol), 573-583

VoIP (Voice over IP), 549

vows of confidence, 6

VPNs, stateful firewalls, 22

VPNs (virtual private networks)

firewalls, IPsec, 812-828

SSL VPNs, 828-841

client-based access, 836-841

clientless access, 829-836

VRFs (Virtual Routing and Forwarding), 18, 202-205

Data Plane, 202-211

dynamic routing protocols, 207-211

interconnecting virtual contexts,
232-241

external router, 233

static routes, 205-206

virtual contexts, 212-225

allocating resources to, 228-232

management access, 225-227

VRF-aware services, 212

**VTIs (Virtual Tunneling Interfaces),
site-to-site IPsec, 818-821**

Auth-Proxy, 650-653

connection limits, defining, 403-406

FTP over IPv6, 764-766

HTTP connections, 762

ICMP connections, 370-373, 758-760

intrazone firewall policies, 410-413

IPv6, 757-766

ACLs, 762-764

membership awareness, 645-649

NAT (Network Address Translation),
391-400

parameter-maps, 758

router traffic, inspecting, 407-410

TCP connections, 377-379

Telnet connections, 761-762

transparent mode, 400-403

UDP connections, 373-377, 760

**ZPF (Zone Policy Firewall). *See* ZFW
(Zone Policy Firewall)**

W-Z

Web Security Appliance (WSA), 25

**Windows Traceroute, through ASA
without NAT, 258-260**

WSA (Web Security Appliance),

**ZFW (Zone Policy Firewall), 361-370,
414, 645**

ACLs, 379-391

application inspection, 478-479

DNS inspection, 479-480

FTP inspection, 481-485

HTTP inspection, 487-493

IM inspection, 494-496