
root@localhost:~$ echo “b477l3 0f l337”

# BATTLE_OF_1337

## CTF 2022

# BATTLE OF 1337 OFFICIAL WRITEUP

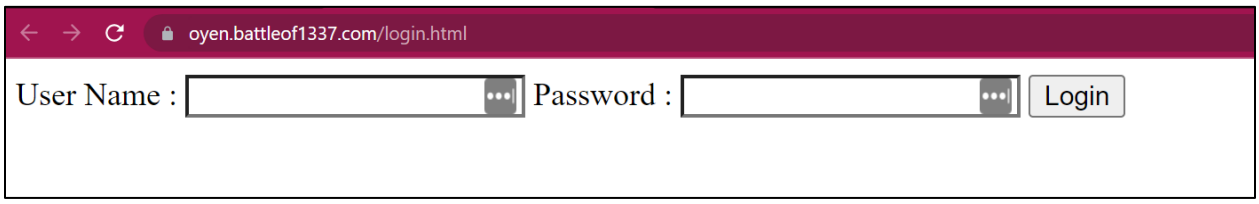
*Writeup By : Cr0wnz*

# TABLE OF CONTENT

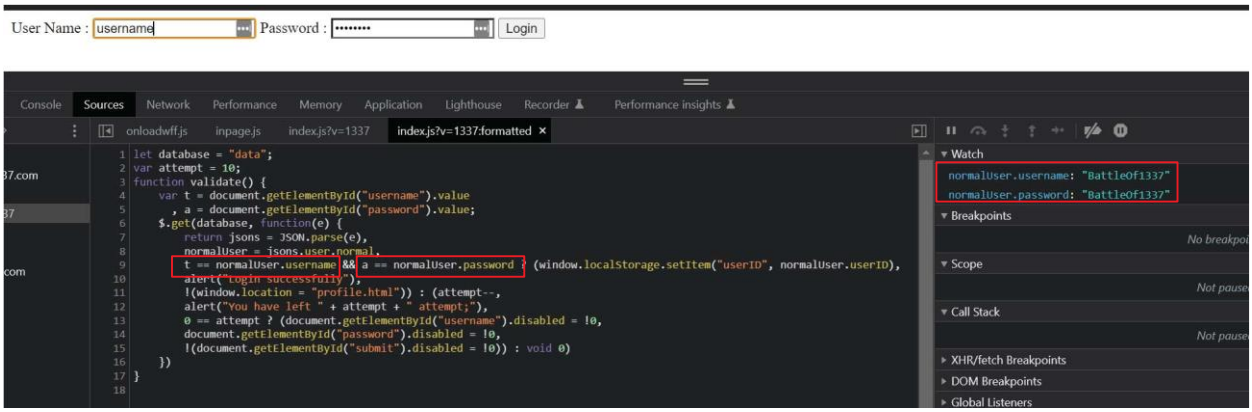
[WEB] - BREAK THE STORAGE ..... 3
[WEB] – CAT-DALMANTION ..... 4
[WEB] – ALICEINWONDERLAND ..... 5
[MISC] – RAYQUAZA ..... 7
[MISC] – HEIHAWRU ..... 9
[MISC] – SHENG XIAO ..... 11
[MISC] – REDPOINT ..... 12
[NET] – SEMERAH PADI ..... 13
[NET] – STREAMLINE ..... 15
[OSINT] – BACK TO THE FUTURE ..... 17
[OSINT] – 1GRAM ..... 18
[OSINT] – SNAP ..... 20
[RE] – SIMPLIFY ..... 22

# [WEB] - BREAK THE STORAGE

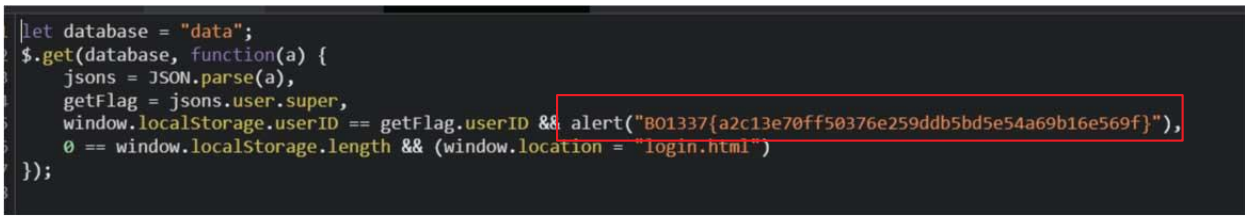

Browse the link inside the challenge, and we can see the username and password input box.




Try inserting any username and password, and this will not get us the flag. Looking at the page source, there is a javascript called. 2 variables can be seen were used to compare with our inputs. Put both variables in the watchlist, and we will get the correct username and password.

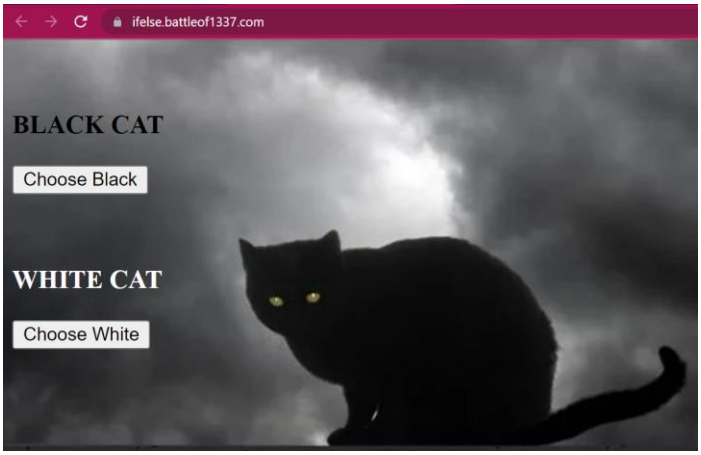



Login and view *profile.js* to get the flag.



## [WEB] – CAT-DALMANTION

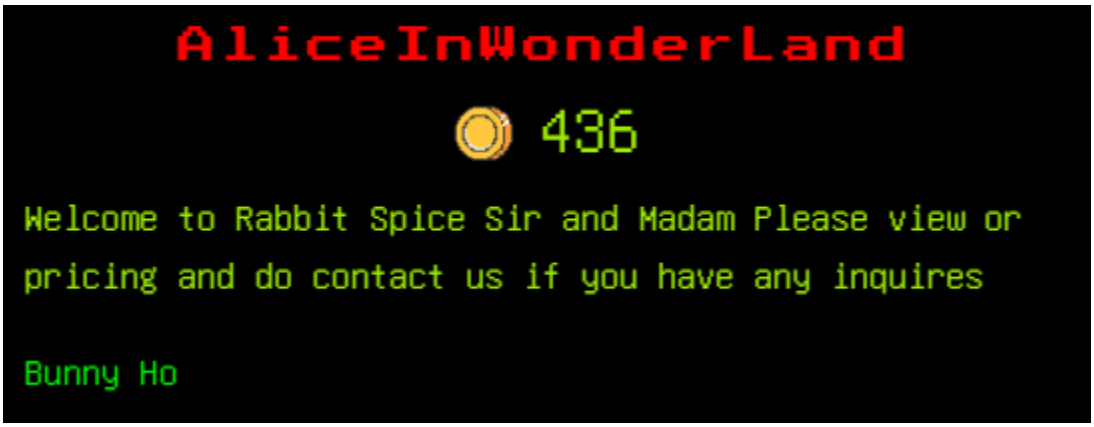

Browse the link inside the challenge, we will see 2 button either to choose “Black” or “White”



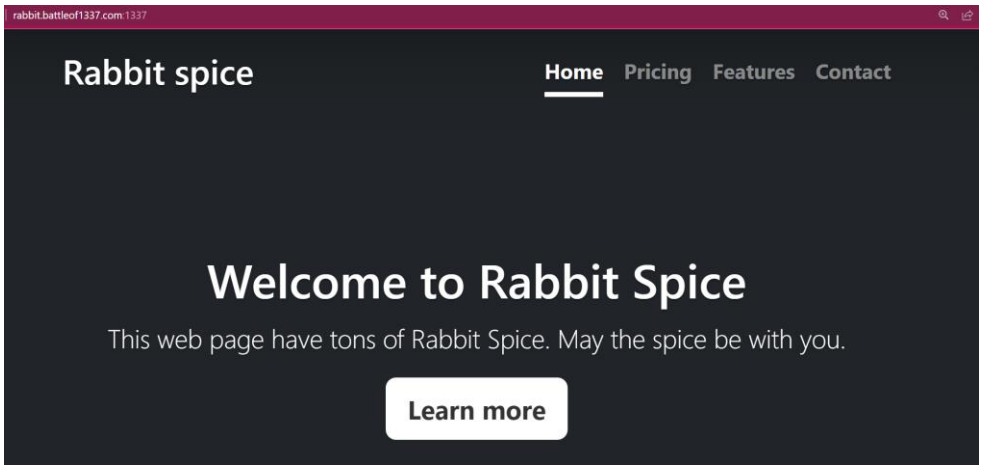

View the header of the response using *curl* command and we can see extra header “flag” with the flag for us to submit.

```
└─▶ curl -I https://ifelse.battleof1337.com/
HTTP/2 200
server: nginx-rc
date: Mon, 18 Jul 2022 11:28:09 GMT
content-type: text/html;charset=UTF-8
vary: Accept-Encoding
flag: BO1337{kuc1n6_5374n}
strict-transport-security: max-age=31536000
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
```

# [WEB] – ALICEINWONDERLAND




Browse the link inside the challenge, we can see few tabs inside the page.




A lot of rabbit holes we will encounter inside this challenge. After a few hints are released, viewing robots.txt will get us a lot of directories with “Disallow.”

```
curl -ks http://rabbit.battleof1337.com:1337/robots.txt | grep -i Disallow
| grep -Ev "\.txt|\.php" | sed 's/Disallow: \///g' >  word.txt
```

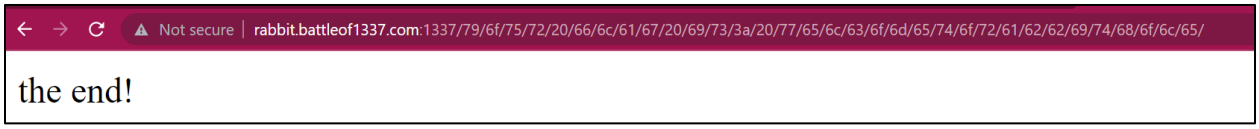
We can try get all the directory names using this command:

Use this command to recurse directory fuzzing:

```
ffuf -ic -u 'http://rabbit.battleof1337.com:1337/FUZZ' -w word.txt:FUZZ -
recursion
```

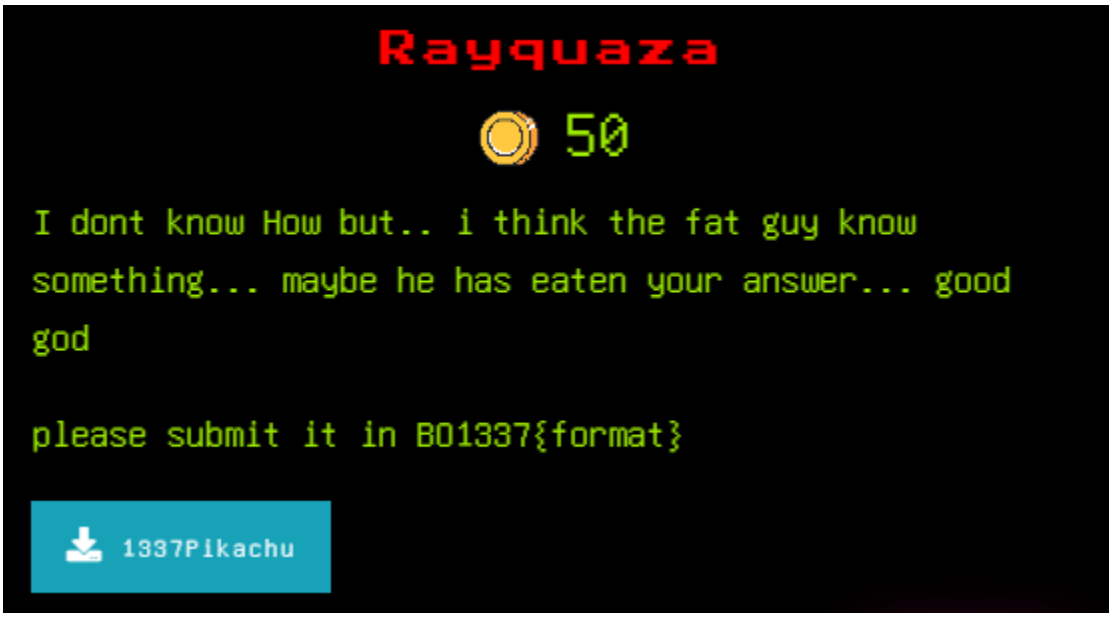
```
[INFO] Starting queued job on target: http://rabbit.battleof1337.com:1337/79/6f/75/72/20/66/6c/61/67/20/69/73/3a/20/77/65/6c/63/6f/6d/65/74/6f/72/61/62/62/69/74/68/6f/
6c/65/FUZZ

:: Progress: [256/256] :: Job [305/305] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::
```

Browse to the URL we found will not get us the flag yet.




To get the flag, we will need to convert all values inside the URL from hex.

# [MISC] – RAYQUAZA



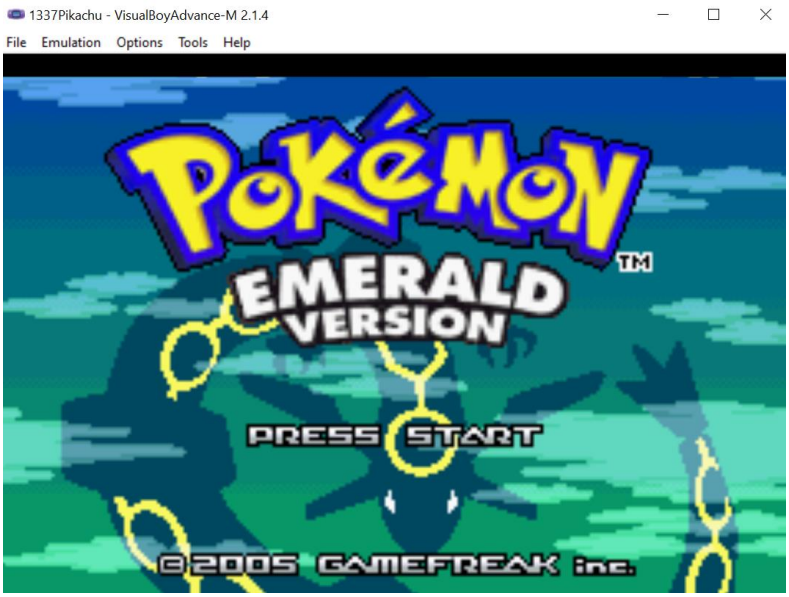

Looking at the information of the file, it is a GBA ROM image.

```
└─▶ file 1337Pikachu
1337Pikachu: Game Boy Advance ROM image: "POKEMON EMER" (BPEE01, Rev.00)
```

We need an emulator to run this GBA image. Here is the link to download the emulator:

- HTTPS://GITHUB.COM/VISUALBOYADVANCE-M/VISUALBOYADVANCE-M/RELEASES/TAG/V2.1.4

As we can see, we are able to load the GBA image using this emulator.



Looking at the description of the challenge, we will need to find a fat guy inside the game to get our flag. As soon as we inside the game, please finish the first part of the game and go out of the house to meet this fat guy.


We will get the flag on the final conversation.

## [MISC] – HEIHAWRU

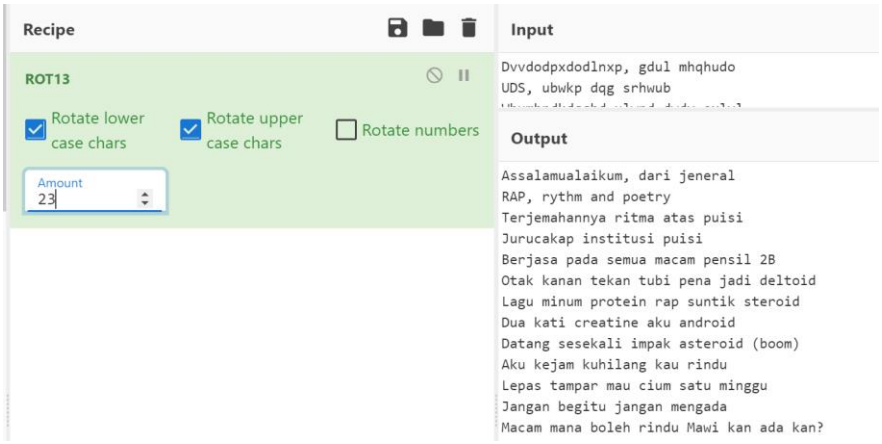

Inside the file, a lot of lines with non-English words can be found.

```
Dvvdodpxdodlnxp, gdul mhqhudo
UDS, ubwkp dqg srhwub
Whumhpdkdqqbd ulwpd dwdv sxlvl
Mxuxfdnds lqvwlwxvl sxlvl
Ehumdvd sdgd vhpxd pdfdp shqvlo 2E
Rwdn ndqdq whndq wxel shqd mdgl ghowrlg
Odjx plqxp surwhlq uds vxqwln vwhurlg
Gxd ndwl fuhdwlqh dnx dqgurlg
Gdwdqj vhvhndol lpsdn dvwhurlg (errp)
Dnx nhmdp nxklodqj ndx ulqgx
Ohsdv wdpsdu pdx flxp vdwx plqjjx
Mdqjdq ehjlwx mdqjdq phqjdgd
Pdfdp pdqd erohk ulqgx Pdzl ndq dgd ndq?
(Dvvdodpxdodlnxp)
Uhdolwlqbd lql 083
Elqwdqj uhdolwl nxmdglndq vdudsdq
Pdndq olpd pdqjnxn wdqbd  pdqd vdudsdqnx?
```

```
5:6:2 6:1:1 31:3:1 15:3:3 15:3:3 43:4:1 27:2:1 32:3:1 33:1:1 41:3:1 38:3:4
24:2:2 10:5:4 41:5:3 45:6:3 35:1:1 15:3:3 1:3:3 36:2:2 34:1:1 45:2:3 21:2:2
17:1:2 11:4:2
```

At the end of the file, there is a weird code given.

After throw the text file inside the Cyberchef, we can see the correct sentence with ROT13.




| [Row]:[Column of word]:[Index of word] |
|---|

The code can be used to find out the correct flag.

I've made a simple script to get the flag. Replace <SNIP> according to what we have.

```
texts = """Assalamualaikum, dari jeneral
RAP, rythm and poetry
<SNIP>
Tolong ceraikan buntut dari kerusi
Bangun"""

codes="""5:6:2
6:1:1
<SNIP>
17:1:2
11:4:2"""

for i in codes.split("\n"):
      a,b,c=i.split(":")
      for index,lines in enumerate(texts.split("\n")):
            if index == (int(a)-1):
                  print(lines.split()[int(b)-1][int(c)-1],end="")
```

Run the script and we will get the flag.

# [MISC] – SHENG XIAO


This challenge is related to “Zodiac Killer”.

To get the flag we just need follow all the symbols above and put it in this website:

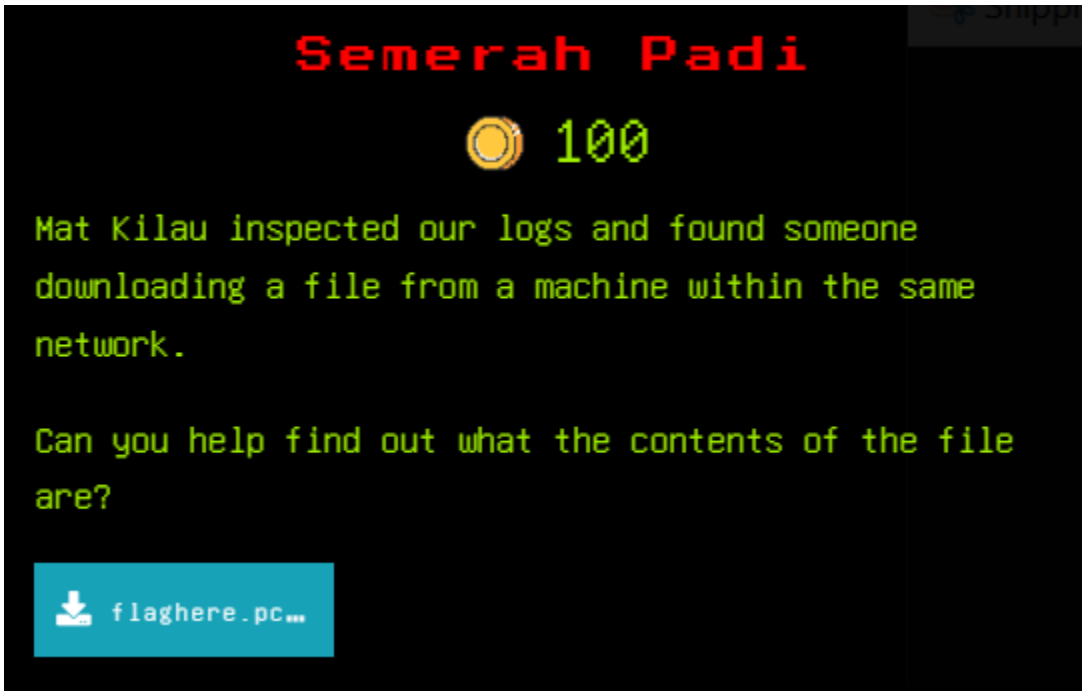
- HTTP://ZODIACKILLERCIPHERS.COM/TYPEWRITER/

# [MISC] – REDPOINT


We receive one PNG image file as below.


After trying different tools and techniques for several hours, the flag is very simple. We just need to wrap “screwdriver” inside BO1337{<FLAG>}

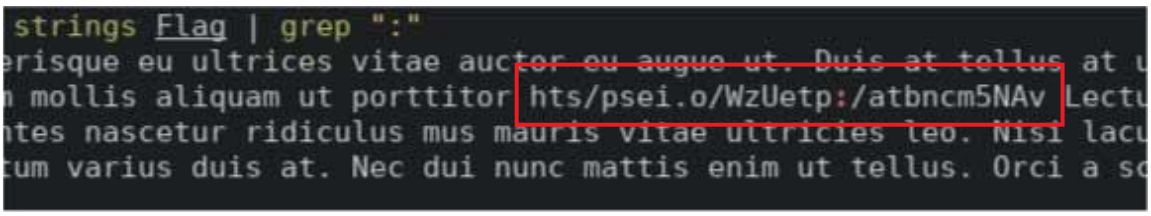
# [NET] – SEMERAH PADI




```
tshark -nr flaghere.pcap --export-objects http
```

It is a pcap file. We can try extract HTTP object using tshark command:

We found 1 file called “Flag”, which will contain a lot of text. Looking around in that file, we found a weird strings.




After looking around, it does contain all characters to build up “https://”. Understanding the string will get us the correct URL which is (Look left and right, that’s how you could build this full URL):

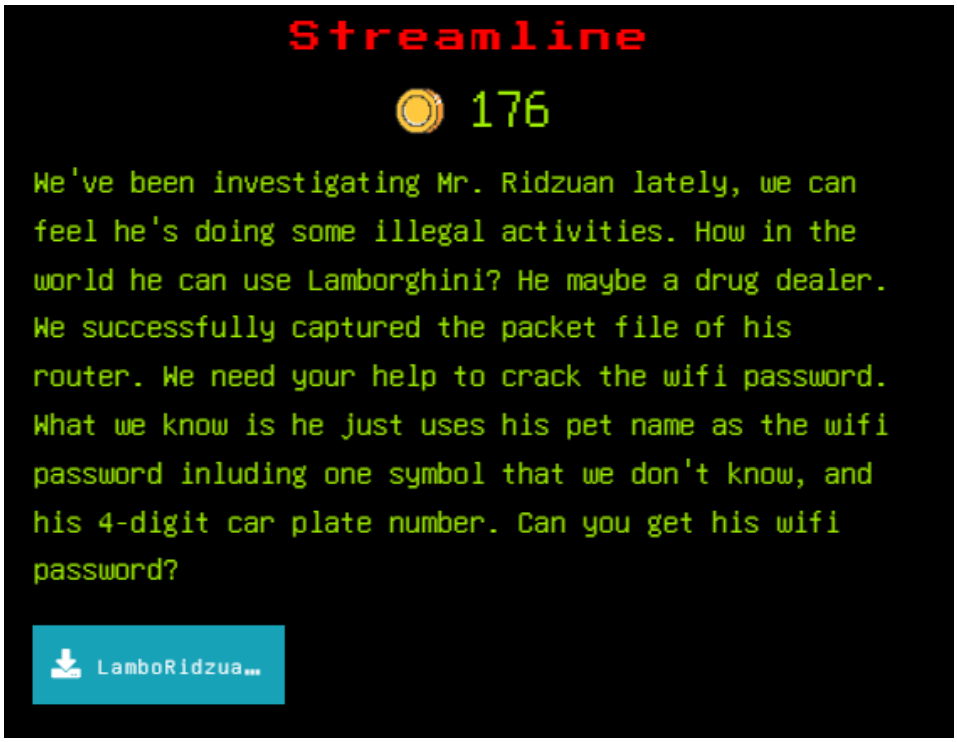
- HTTPS://PASTEBIN.COM/5WNZAUVE

Inside the pastebin page, we will get another 2 downloadable files.

- HTTPS://UFILE.IO/BO0CUBRA
- HTTPS://UFILE.IO/G6XQRBL2

To get the flag, we need to open the mp3 file using “sonic-visualizer”. Import the wav file that we received and pressed (Shift+G) to Add spectrogram.

## [NET] – STREAMLINE




```
steghide extract -sf LamboRidzuan.jpg
```

We received an image file. Using “steghide”, we could extract another pcap file.

The pcap file does look encrypted so we can’t view the correct traffic within it. We need a password to decrypt it. First, let’s get the hash from this pcap using hashcat tool.

```
/usr/lib/hashcat-utils/cap2hccapx.bin handshake.cap handshake.hccapx
```

```
└─> /usr/lib/hashcat-utils/cap2hccapx.bin handshake.cap handshake.hccapx
Networks detected: 1

[*] BSSID=ae:0b:fb:d7:e5:a2 ESSID=Ridzuan Wifi (Length: 12)
 --> STA=a8:9c:ed:1c:53:05, Message Pair=0, Replay Counter=1
 --> STA=a8:9c:ed:1c:53:05, Message Pair=2, Replay Counter=1

Written 2 WPA Handshakes to: handshake.hccapx
```

To craft our password wordlist, we need to understand the challenge’s description. The password will be “[petname]+[1 symbol]+[4 digit car plate number]”. From this information, we can use hashcat command to crack it.

```
hashcat -a 3 -m 2500 -1 ?d -2 ?s handshake.hccapx oyen?2?1?1?1?1
```

```
└─▶ hashcat -a 3 -m 2500 -1 ?d -2 ?s handshake.hccapx oyen?2?1?1?1?1 --show
ae0bfbd7e5a2:a89ced1c5305:Ridzuan Wifi:oyen@9367
```

The petname is a bit guessy which you can see the orange cat above the car in the image.

# [OSINT] – BACK TO THE FUTURE


The flag can be seen using wayback URL:

- HTTPS://WEB.ARCHIVE.ORG/WEB/20220704083124/HTTPS://B2F.BATTLEOF1337.COM/

View the page source and find the flag format “BO1337”.

```
      <li class="d-flex">
        <i class="mdi mdi-email mr-1"></i>
        <span>omgjasonbourne@jmail.com</span>
      </li>
      <li class="d-flex">
        <i class="mdi mdi-phone mr-1"></i>
        <span>BO1337{aHR0cHM6Ly9veW0uY2F0bWUuY2Y=}</span>
      </li>
    </ul>
  </div>
</a>
div>
v>
```

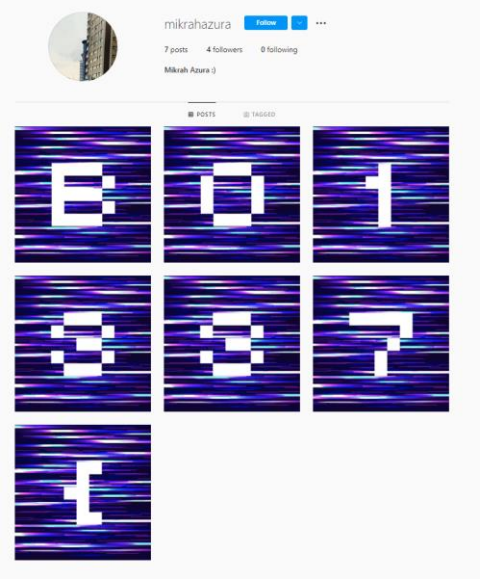
# [OSINT] – 1GRAM


Looking at the same URL we received, we found a clue in the page source:

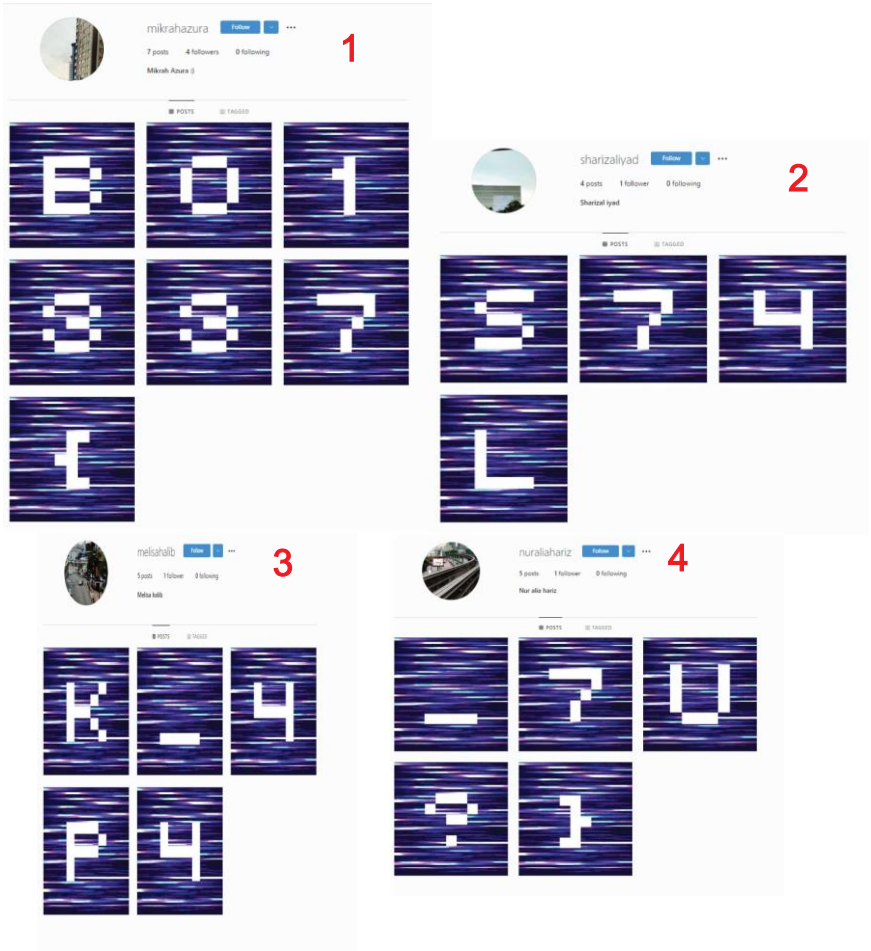
- HTTPS://B2F.BATTLEOF1337.COM

```
  <div class="col-md-6">
    <div class="contact-info px-4">
      <h4 class="mb-1">Contact Details</h4>
      <p class="text-dark font-weight-medium pt-4 mb-2">Email address</p>
      <p>mikrahazura@gmail.com</p>
      <p class="text-dark font-weight-medium pt-4 mb-2">Phone Number</p>
      <p>+60189*906*6</p>
      <p class="text-dark font-weight-medium pt-4 mb-2">Birthday</p>
      <p>Nov 15, 1990</p>
      <p class="text-dark font-weight-medium pt-4 mb-2">Event</p>
      <p>Im the clue you looking for please follow me at instagram @mikrahazura</p>
    </div>
  </div>
 </div>
/div>
```

The user @mikrahazura have the flag posted. But it's not the full flag yet.

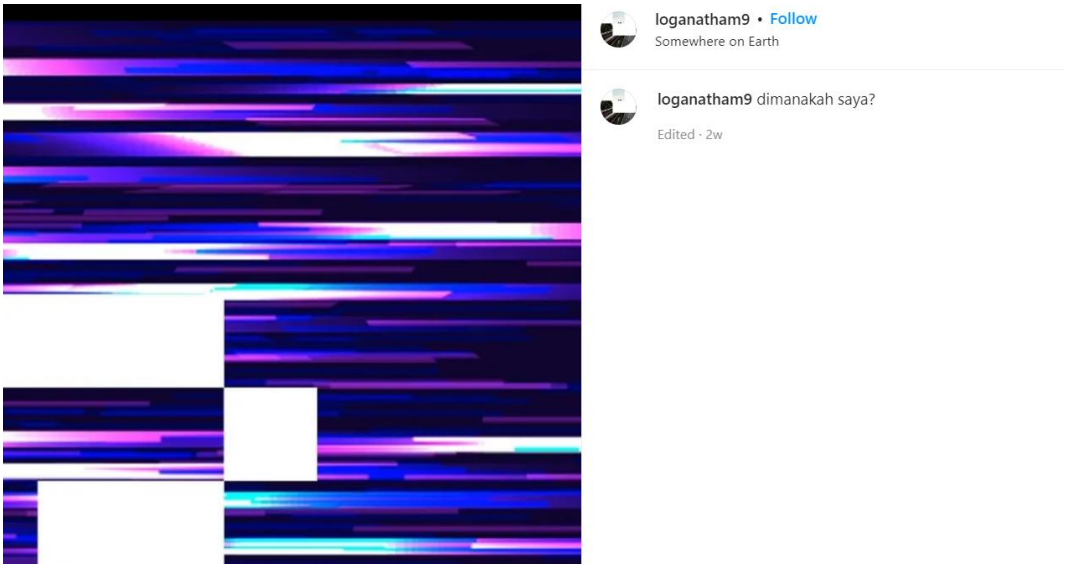



Looking at one of the pictures, we can see other users tagged. One of the users tagged, also contains the same picture as above. Below are the users with the flag pictures.

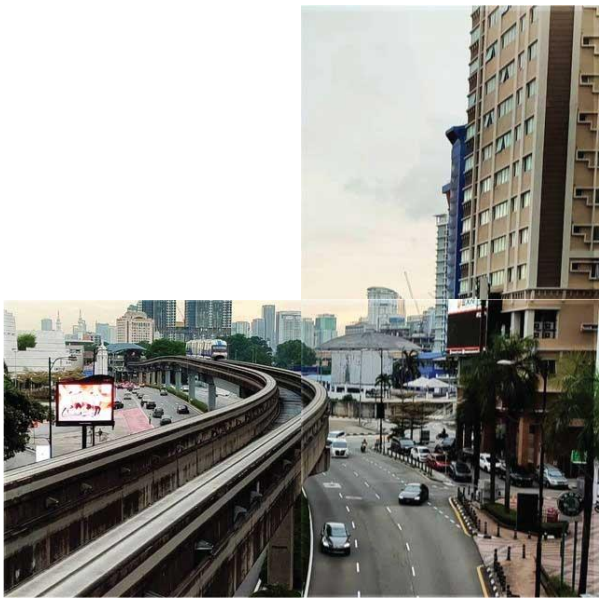


# [OSINT] – SNAP


We can also see another user posted an image with a sentence “dimanakah saya?”.

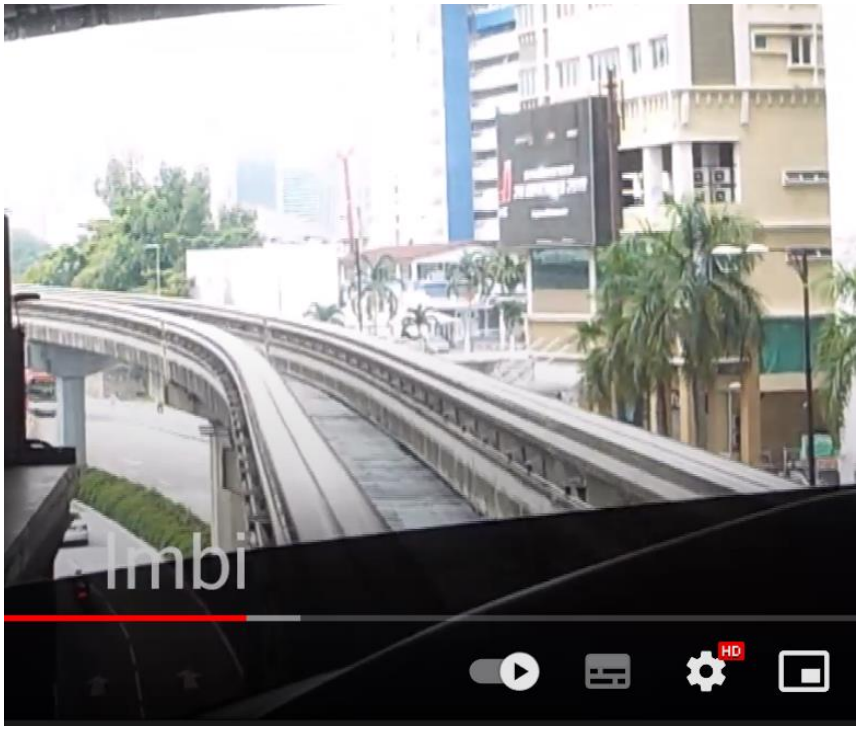



After observing every user Instagram that we found. We can see their profile pictures can make into one image as below.



I'm not very good with geolocation. But we know for sure it's near to a monorail station. After looking at the monorail walkthrough in youtube, we can find out that the flag is BO1337{Imbi}

- https://www.youtube.com/watch?v=ff7SFBCpynQ



## [RE] – SIMPLIFY


Open the binary in Ghidra. We can see there is only 2 different output, which either “Correct code!” or “Wrong code..”.

```
undefined8 main(undefined8 param_1,undefined8 param_2)

{
  undefined8 in_R9;
  long in_FS_OFFSET;
  int local_20;
  uint local_1c;
  uint local_18;
  uint local_14;
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  printf("Enter code: ");
  __isoc99_scanf("%i-%i-%i-%i",&local_20,&local_1c,&local_18,&local_14,in_R9,param_2);
  if ((((local_20 + local_14 * 3 == 0x467c) && (local_1c * local_18 * 3 == 0x4ef3ae)) &&
      (local_20 == 0x3f2)) && (local_18 * 0xc0d3 + local_14 == 0x3cbbd6c)) {
    printf("Correct code! The flag is %i-%i-%i-%i\n",0x3f2,(ulong)local_1c,(ulong)local_18,
           (ulong)local_14);
  }
  else {
    puts("Wrong code..");
  }
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return 0;
}
```

Based on this information, let's try make a simple angr script to get our flag!

```
import angr
import sys

def main(argv):
  path_to_binary = "./crackme"
  project = angr.Project(path_to_binary)
  initial_state = project.factory.entry_state()
  simulation = project.factory.simgr(initial_state)

  def is_successful(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())
    if b'Correct code! The flag is ' in stdout_output:
      return True
    else: return False

  def should_abort(state):
    stdout_output = state.posix.dumps(sys.stdout.fileno())

    if b'Wrong code..' in  stdout_output:
      return True
    else: return False

  simulation.explore(find=is_successful, avoid=should_abort)

  if simulation.found:
    solution_state = simulation.found[0]
    solution = solution_state.posix.dumps(sys.stdin.fileno())
    print("[+] Success! Solution is: {}".format(solution.decode("utf-8")))

  else:
    raise Exception('Could not find the solution')

if __name__ == '__main__':
  main(sys.argv)
```

Run "python3 solve.py" and we will get the correct input.

```
WARNING | 2022-07-19 18:47:08,965 | cle.loader | The main binary is a
[+] Success! Solution is: 0000001010-0000001337-0000001290-0000005678
```

```
Enter code: 1010-1337-1290-5678
Correct code! The flag is 1010-1337-1290-5678
```