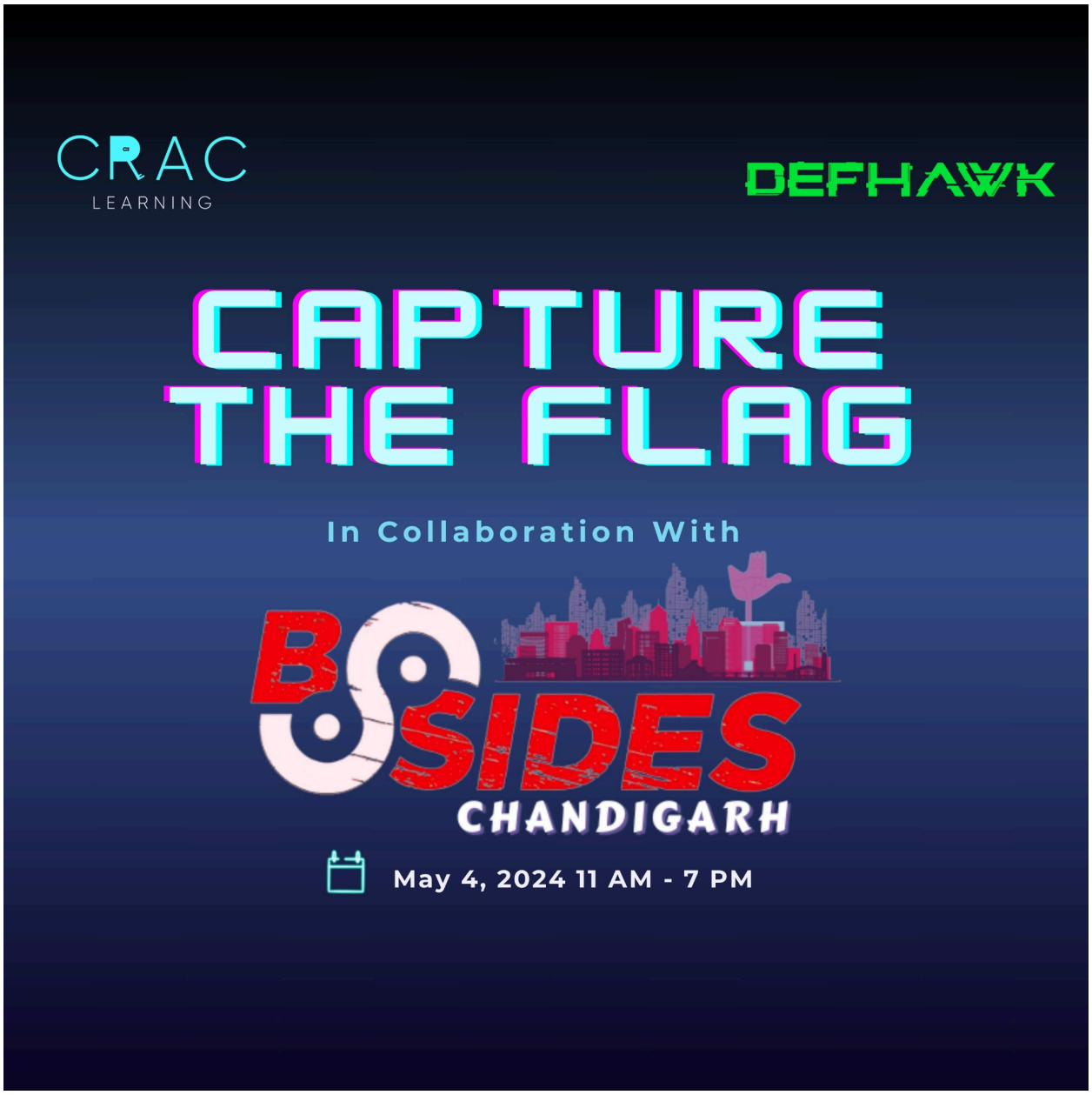
Bsides Chandigarh CTF 2024



GitHub: https://github.com/saitejagvt

Adhoc / MYBANK

How can I know more about my bank and verify its number? Read the directory and drop the concatenated positions (starting with 0 and containing 1) of valid BNs as flag. E.g., CRAC{index1index2..}

Attatchment: list_isbn.txt

https://defhawk-assets-prod.s3.ap-south-1.amazonaws.com/challenge/assets/663564540024f01cf7abf273/list_isbn.txt

We can say from the description and attachment that we have to validate ISBN codes and print indices starting from 0, and 1 in index.

```
def is_valid(isbn):
    isbn = (isbn or "").replace('-', '')
    if not re.match(r'^\d{9}[\dX]$', isbn):
        return False
    digits = [d == 'X' and 10 or int(d) for d in isbn]
    summ = sum(d * (10 - i) for i, d in enumerate(digits))
    return summ % 11 == 0
import requests, re
r =
requests.get("https://defhawk-assets-prod.s3.ap-south-1.amazonaws.com/challenge/assets/663564540024f01cf7abf273/list_isbn.txt")
l = re.findall(r"[0-9]{1,3}: ([0-9]{1}-[0-9]{3}-[0-9]{5}-[0-9]{1})",r.text)
out = ""
for i in range(len(l)):
        if is_valid1(l[i]):
                if "1" in str(i):
                        out+=str(i)
print(f"CRAC{{{out}}}")
```

References: https://exercism.org/tracks/python/exercises/isbn-verifier

https://exercism.org/tracks/python/exercises/isbn-verifier/solutions

Digital Forensics / fixmypdf

Can you fix my pdf and save it from this hidden bug?

https://defhawk-assets-prod.s3.ap-south-1.amazonaws.com/challenge/assets/66354b430024f01cf7abef53/final_fav.pdf

Fix magic number of pdf using hexedit.

Ref: https://en.wikipedia.org/wiki/List_of_file_signatures

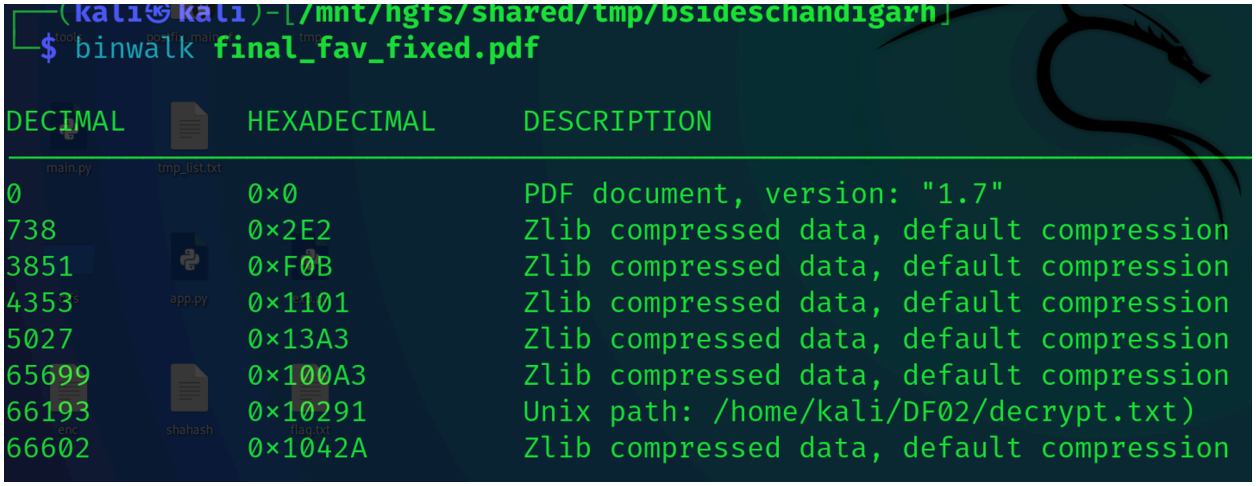
```
┌──(kali㉿kali)-[/mnt/hgfs/shared/tmp/bsideschandigarh]
└─$ xxd final_fav.pdf | head
00000000: 1951 4341 2031 2e37 0a25 c0ff eefa bada  .QCA 1.7.%......
00000010: 0a31 2030 206f 626a 0a3c 3c20 2f4d 6574  .1 0 obj.<< /Met
00000020: 6164 6174 6120 3335 2030 2052 0a2f 4e61  adata 35 0 R./Na
00000030: 6d65 7320 3c3c 202f 456d 6265 6464 6564  mes << /Embedded
00000040: 4669 6c65 7320 3c3c 202f 4e61 6d65 7320  Files << /Names
00000050: 5b20 3c32 6636 3836 6636 6436 3532 6636  [ <2f686f6d652f6
00000060: 6236 3136 6336 3932 6634 3434 3633 3033  b616c692f4446303
00000070: 3232 6636 3436 3536 3337 3237 3937 3037  22f6465637279707
00000080: 3432 6537 3437 3837 343e 2033 3920 3020  42e747874> 39 0
00000090: 5220 5d20 3e3e 203e 3e0a 2f4d 6172 6b49  R ] >> >>./MarkI
```

To

```
┌──(kali㉿kali)-[/mnt/hgfs/shared/tmp/bsideschandigarh]
└─$ xxd final_fav_fixed.pdf | head
00000000: 2550 4446 2d31 2e37 0a25 c0ff eefa bada  %PDF-1.7.%......
00000010: 0a31 2030 206f 626a 0a3c 3c20 2f4d 6574  .1 0 obj.<< /Met
00000020: 6164 6174 6120 3335 2030 2052 0a2f 4e61  adata 35 0 R./Na
00000030: 6d65 7320 3c3c 202f 456d 6265 6464 6564  mes << /Embedded
00000040: 4669 6c65 7320 3c3c 202f 4e61 6d65 7320  Files << /Names
00000050: 5b20 3c32 6636 3836 6636 6436 3532 6636  [ <2f686f6d652f6
00000060: 6236 3136 6336 3932 6634 3434 3633 3033  b616c692f4446303
00000070: 3232 6636 3436 3536 3337 3237 3937 3037  22f6465637279707
00000080: 3432 6537 3437 3837 343e 2033 3920 3020  42e747874> 39 0
00000090: 5220 5d20 3e3e 203e 3e0a 2f4d 6172 6b49  R ] >> >>./MarkI
```

We get a cipher text after opening fixed pdf. Seems to be AES but no key given.

Checked with binwalk.



Extract using “binwalk final_fav_fixed.pdf -e”

We get a script when we look those files.

```
import base64


from Crypto.Cipher import AES

from Crypto.Util.Padding import pad, unpad

from Crypto.Random import get_random_bytes


def decrypt(ciphertext, key):

    # Extract the IV from the ciphertext

    iv = ciphertext[:AES.block_size]


    # Create the AES cipher object

    cipher = AES.new(key, AES.MODE_CBC, iv)


    # Decrypt the ciphertext
```

```
    decrypted_text = cipher.decrypt(ciphertext[AES.block_size:])

    # Unpad the decrypted plaintext
    plaintext = unpad(decrypted_text, AES.block_size)

    return plaintext

# Example usage
key = b'CanY0uGuessKey?$12345678' + b'\x00' * 8  # Pad key to 32 bytes
ciphertext = base64.b64decode("AM0YtFJuiTzG8PCtLKmZjB/Mfg3ixoGjyaLMMgKEcHoWWav/CujQd+RXpKh6mPo8AyHGANnuQwigiBWrBV05lA==")
decrypted_text = decrypt(ciphertext, key)
print("Decrypted Text:", decrypted_text)
```

Just run this and we get flag.

Digital Forensics / Louder

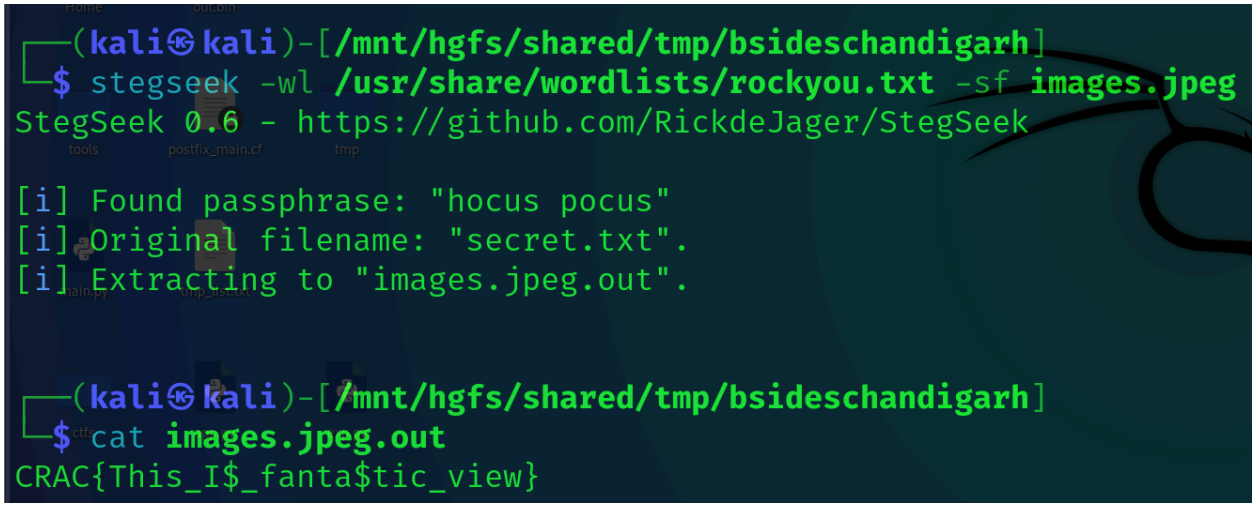
wget https://defhawk-assets-prod.s3.ap-south-1.amazonaws.com/challenge/assets/66079818b0899ad074096667/images.jpeg

Download the image.

Ran exiftool, binwalk, etc, but didnt get any info.

Used stegseek and got flag.

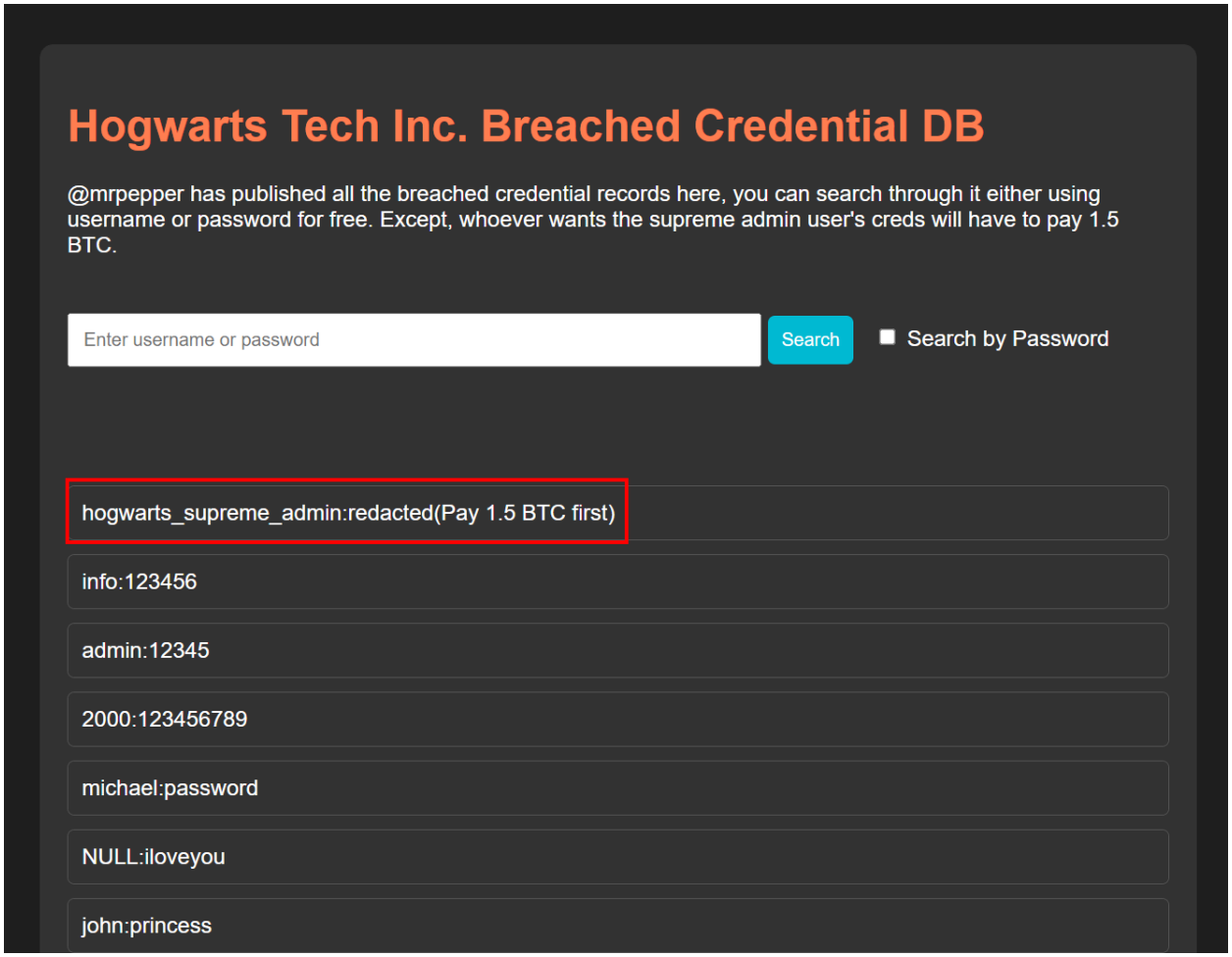
stegseek -wl /usr/share/wordlists/rockyou.txt -sf images.jpeg



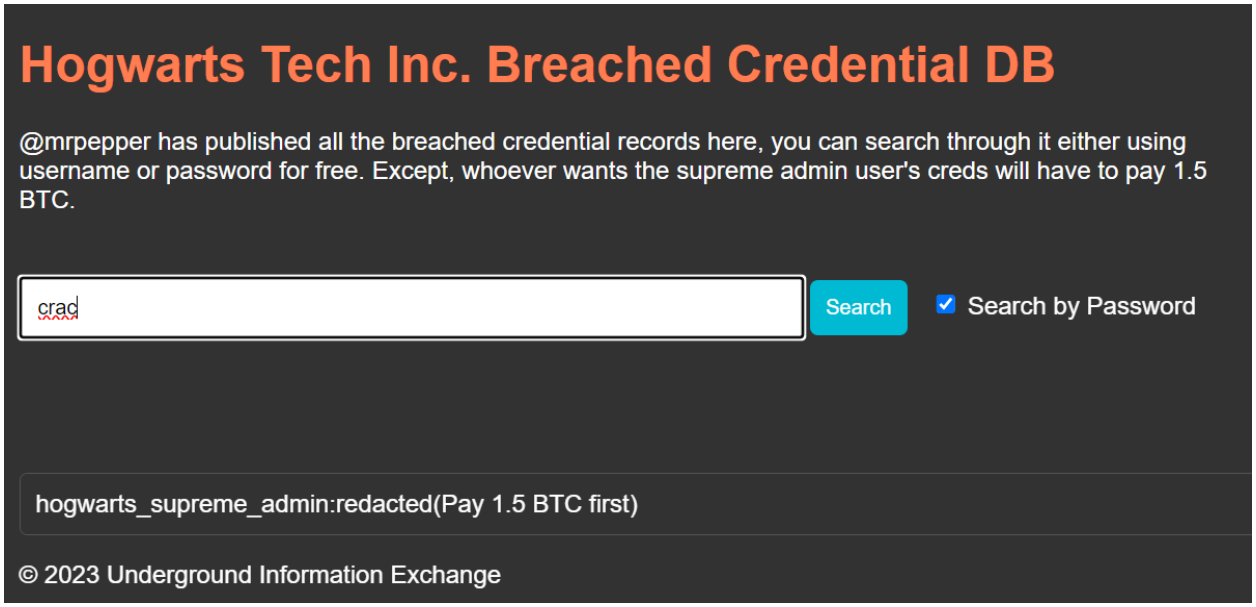
Scripting / breached

Explore the given website.
We get http://web.ctf.defhawk.com:5000/hogwartstechinc-breached-20231231 where we can check breached passwords.

After some inputs, when we pass empty input, we get list of all usernames and passwords breached but except one password.



We need to find the password which might be the flag.



Check the password checkbox, and when we search “crac”(Flag Format) we will get the username again. And Flag format is CRAC{} but here it’s crac because CRAC didnt work.

When we append { to crac and search, we also get username but when we search other characters, we wont get any results.
Now we can use this as success response to bruteforce the characters of flag.
I wrote a python script for that.

```
import requests
import string
flag='crac{'
url = 'http://web.ctf.defhawk.com:5000/search-db'
headers={"Content-Type": "application/x-www-form-urlencoded"}
chars = string.ascii_lowercase+string.ascii_uppercase+string.digits+'_{}}!$@'
while True:
    for ch in chars:
        data=f'password={flag+ch}'
        print(data)
        r = requests.post(url,data=data,headers=headers).text
        print(r)
        if 'hogwarts_supreme_admin' in r:
            flag=flag+ch
            break
    if "}" in flag:
        break
print(flag)
```

It’s slow because of server response time and number of possible characters.

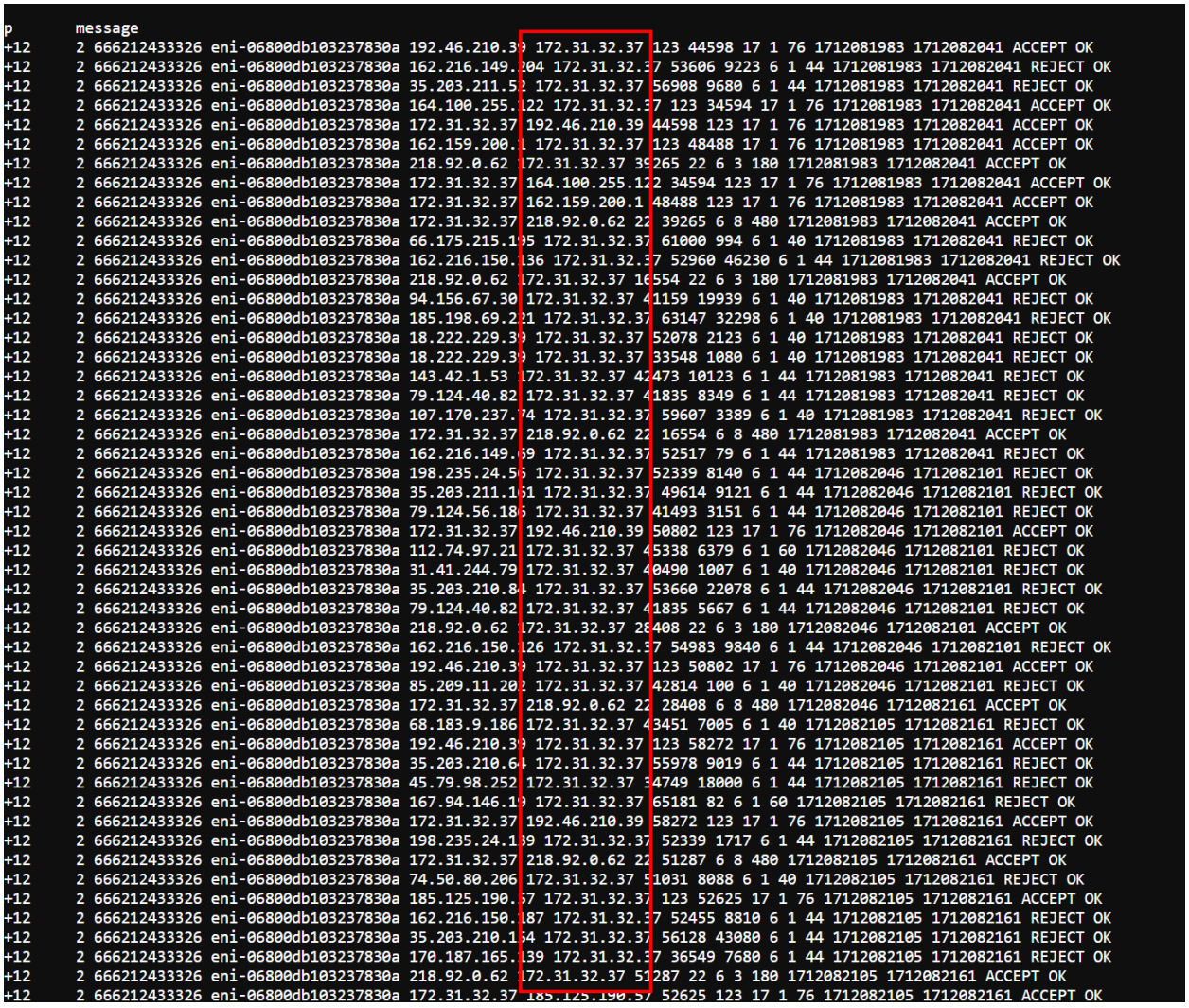
```
{"search_results":[]}

password=crac{{hohch0c73747d0e0e0wwissag{
{"search_results":[]}

password=crac{{hohch0c73747d0e0e0wwissag}
{"search_results":["hogwarts_supreme_admin:redacted(Pay 1.5 BTC first)"]}

crac{{hohch0c73747d0e0e0wwissag}
[Finished in 1525.6s]
```

At last we get the flag.

Cloud Security / FlowLogs

Attachment:
https://defhawk-assets-prod.s3.ap-south-1.amazonaws.com/challenge/assets/660c539ab0899ad0740984d3/flow-flogs.txt

Our aim is to find victim's IP address who got attacked.
After just having a glance we can tell that huge number of ips are trying to connect to single IP address: 172.31.32.37



```
p       message
+12     2 666212433326 eni-06800db103237830a 192.46.210.39 172.31.32.37 123 44598 17 1 76 1712081983 1712082041 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 162.216.149.204 172.31.32.37 53606 9223 6 1 44 1712081983 1712082041 REJECT OK
+12     2 666212433326 eni-06800db103237830a 35.203.211.52 172.31.32.37 56908 9680 6 1 44 1712081983 1712082041 REJECT OK
+12     2 666212433326 eni-06800db103237830a 164.100.255.122 172.31.32.37 123 34594 17 1 76 1712081983 1712082041 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 172.31.32.37 192.46.210.39 44598 123 17 1 76 1712081983 1712082041 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 162.159.200.1 172.31.32.37 123 48488 17 1 76 1712081983 1712082041 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 218.92.0.62 172.31.32.37 39265 22 6 3 180 1712081983 1712082041 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 172.31.32.37 164.100.255.122 34594 123 17 1 76 1712081983 1712082041 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 172.31.32.37 162.159.200.1 48488 123 17 1 76 1712081983 1712082041 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 172.31.32.37 218.92.0.62 22 39265 6 8 480 1712081983 1712082041 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 66.175.215.195 172.31.32.37 61000 994 6 1 40 1712081983 1712082041 REJECT OK
+12     2 666212433326 eni-06800db103237830a 162.216.150.136 172.31.32.37 52960 46230 6 1 44 1712081983 1712082041 REJECT OK
+12     2 666212433326 eni-06800db103237830a 218.92.0.62 172.31.32.37 16554 22 6 3 180 1712081983 1712082041 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 94.156.67.30 172.31.32.37 41159 19939 6 1 40 1712081983 1712082041 REJECT OK
+12     2 666212433326 eni-06800db103237830a 185.198.69.221 172.31.32.37 63147 32298 6 1 40 1712081983 1712082041 REJECT OK
+12     2 666212433326 eni-06800db103237830a 18.222.229.39 172.31.32.37 52078 2123 6 1 40 1712081983 1712082041 REJECT OK
+12     2 666212433326 eni-06800db103237830a 18.222.229.39 172.31.32.37 33548 1080 6 1 40 1712081983 1712082041 REJECT OK
+12     2 666212433326 eni-06800db103237830a 143.42.1.53 172.31.32.37 42473 10123 6 1 44 1712081983 1712082041 REJECT OK
+12     2 666212433326 eni-06800db103237830a 79.124.40.82 172.31.32.37 41835 8349 6 1 44 1712081983 1712082041 REJECT OK
+12     2 666212433326 eni-06800db103237830a 107.170.237.74 172.31.32.37 59607 3389 6 1 40 1712081983 1712082041 REJECT OK
+12     2 666212433326 eni-06800db103237830a 172.31.32.37 218.92.0.62 22 16554 6 8 480 1712081983 1712082041 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 162.216.149.69 172.31.32.37 52517 79 6 1 44 1712081983 1712082041 REJECT OK
+12     2 666212433326 eni-06800db103237830a 198.235.24.56 172.31.32.37 52339 8140 6 1 44 1712082046 1712082101 REJECT OK
+12     2 666212433326 eni-06800db103237830a 35.203.211.161 172.31.32.37 49614 9121 6 1 44 1712082046 1712082101 REJECT OK
+12     2 666212433326 eni-06800db103237830a 79.124.56.186 172.31.32.37 41493 3151 6 1 44 1712082046 1712082101 REJECT OK
+12     2 666212433326 eni-06800db103237830a 172.31.32.37 192.46.210.39 50802 123 17 1 76 1712082046 1712082101 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 112.74.97.21 172.31.32.37 45338 6379 6 1 60 1712082046 1712082101 REJECT OK
+12     2 666212433326 eni-06800db103237830a 31.41.244.79 172.31.32.37 40490 1007 6 1 40 1712082046 1712082101 REJECT OK
+12     2 666212433326 eni-06800db103237830a 35.203.210.84 172.31.32.37 53660 22078 6 1 44 1712082046 1712082101 REJECT OK
+12     2 666212433326 eni-06800db103237830a 79.124.40.82 172.31.32.37 41835 5667 6 1 44 1712082046 1712082101 REJECT OK
+12     2 666212433326 eni-06800db103237830a 218.92.0.62 172.31.32.37 28408 22 6 3 180 1712082046 1712082101 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 162.216.150.126 172.31.32.37 54983 9840 6 1 44 1712082046 1712082101 REJECT OK
+12     2 666212433326 eni-06800db103237830a 192.46.210.39 172.31.32.37 123 50802 17 1 76 1712082046 1712082101 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 85.209.11.202 172.31.32.37 42814 100 6 1 40 1712082046 1712082101 REJECT OK
+12     2 666212433326 eni-06800db103237830a 172.31.32.37 218.92.0.62 22 28408 6 8 480 1712082046 1712082161 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 68.183.9.186 172.31.32.37 43451 7005 6 1 40 1712082105 1712082161 REJECT OK
+12     2 666212433326 eni-06800db103237830a 192.46.210.39 172.31.32.37 123 58272 17 1 76 1712082105 1712082161 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 35.203.210.64 172.31.32.37 55978 9019 6 1 44 1712082105 1712082161 REJECT OK
+12     2 666212433326 eni-06800db103237830a 45.79.98.252 172.31.32.37 34749 18000 6 1 44 1712082105 1712082161 REJECT OK
+12     2 666212433326 eni-06800db103237830a 167.94.146.19 172.31.32.37 65181 82 6 1 60 1712082105 1712082161 REJECT OK
+12     2 666212433326 eni-06800db103237830a 172.31.32.37 192.46.210.39 58272 123 17 1 76 1712082105 1712082161 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 198.235.24.139 172.31.32.37 52339 1717 6 1 44 1712082105 1712082161 REJECT OK
+12     2 666212433326 eni-06800db103237830a 172.31.32.37 218.92.0.62 22 51287 6 8 480 1712082105 1712082161 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 74.50.80.206 172.31.32.37 51031 8088 6 1 40 1712082105 1712082161 REJECT OK
+12     2 666212433326 eni-06800db103237830a 185.125.190.57 172.31.32.37 123 52625 17 1 76 1712082105 1712082161 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 162.216.150.187 172.31.32.37 52455 8810 6 1 44 1712082105 1712082161 REJECT OK
+12     2 666212433326 eni-06800db103237830a 35.203.210.154 172.31.32.37 56128 43080 6 1 44 1712082105 1712082161 REJECT OK
+12     2 666212433326 eni-06800db103237830a 170.187.165.139 172.31.32.37 36549 7680 6 1 44 1712082105 1712082161 REJECT OK
+12     2 666212433326 eni-06800db103237830a 218.92.0.62 172.31.32.37 51287 22 6 3 180 1712082105 1712082161 ACCEPT OK
+12     2 666212433326 eni-06800db103237830a 172.31.32.37 185.125.190.57 52625 123 17 1 76 1712082105 1712082161 ACCEPT OK
```

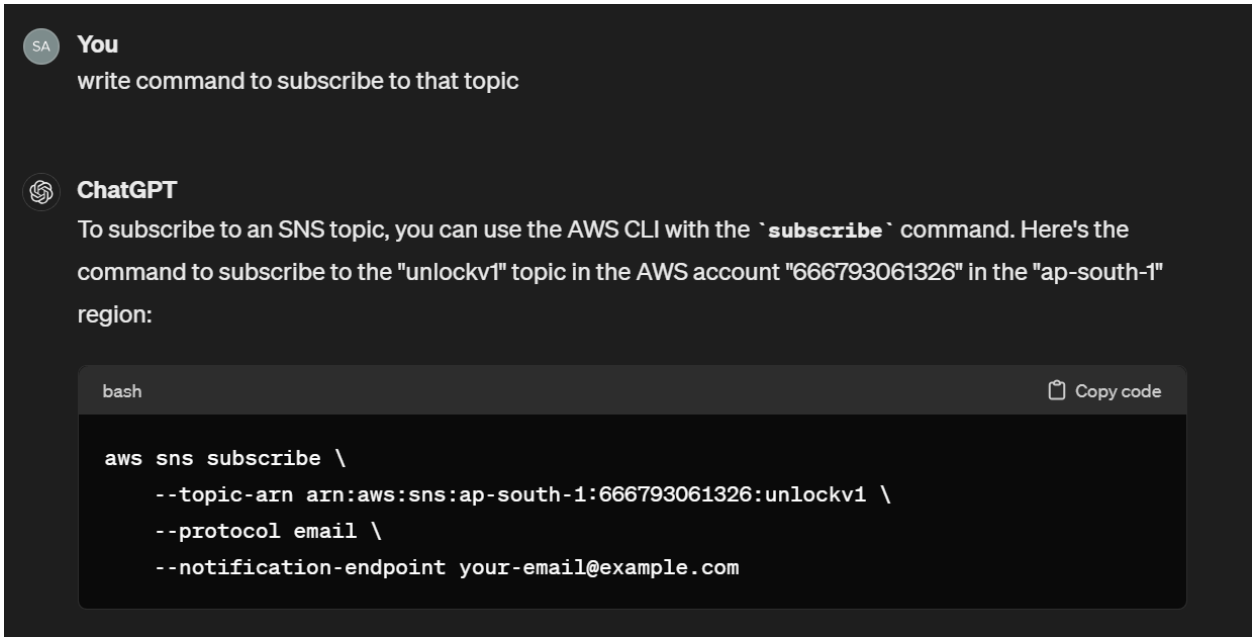
Flag: CRAC{172.31.32.37}

Cloud Security / Follow Us

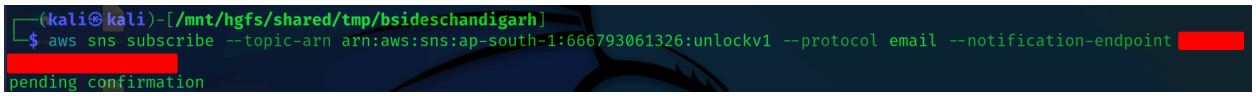
Do you want to hear about latest security updates? Follow or topic - unlockv1 in ap-south-1 region of our account 666793061326

We have,
Region: ap-south-1
TopicName: unlockv1
Account ID: 666793061326

After some research on aws docs, and chatGPT, I got the final command.

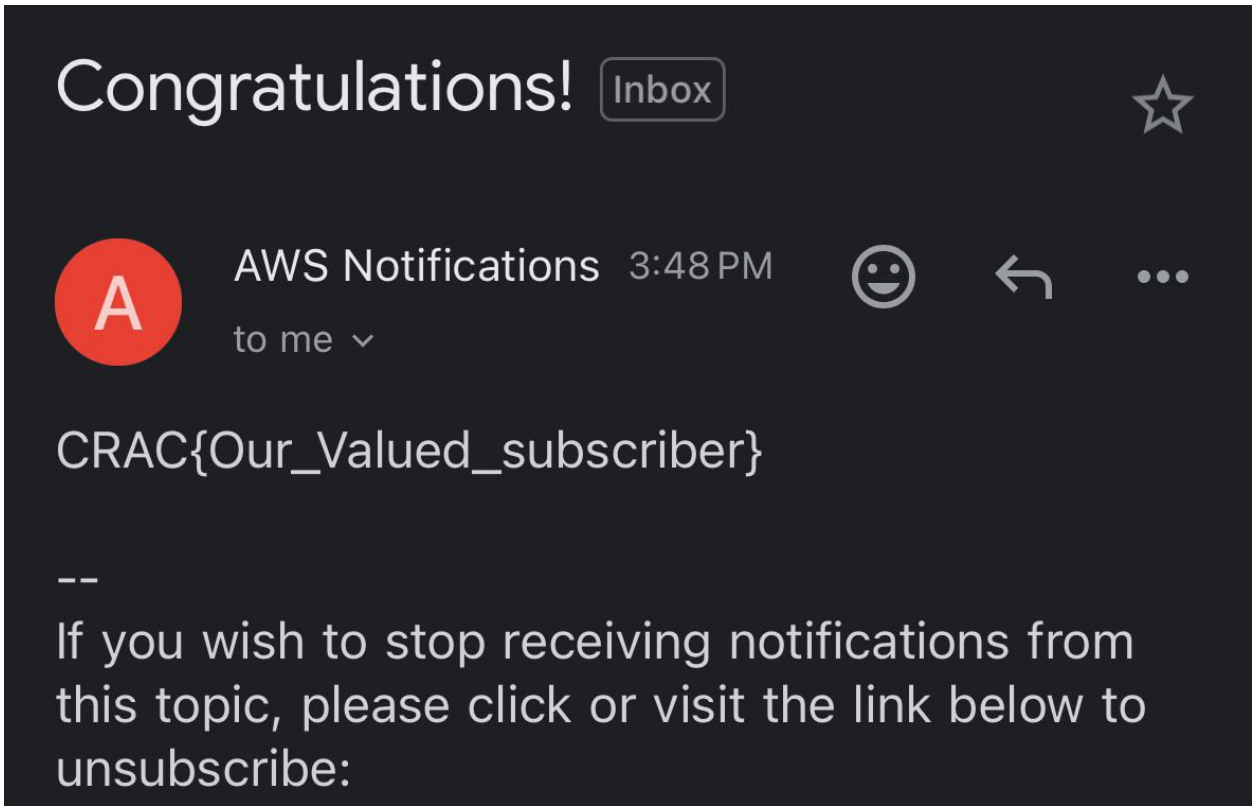


aws sns subscribe --topic-arn arn:aws:sns:ap-south-1:666793061326:unlockv1 --protocol email --notification-endpoint email@gmail.com



Before this, you have to configure your AWS account using aws configure command.

After executing the command, You'll get mail after few seconds or minutes with flag.



Cloud Security / LOOP

Figure out what is my gateway? Located in ap-south-1 with id h2vb2gupxd on prod as collectBits

API Gateway is used in AWS Lambda services.
https://`restapi_id`.execute-api.`region`.amazonaws.com/`stage_name`/

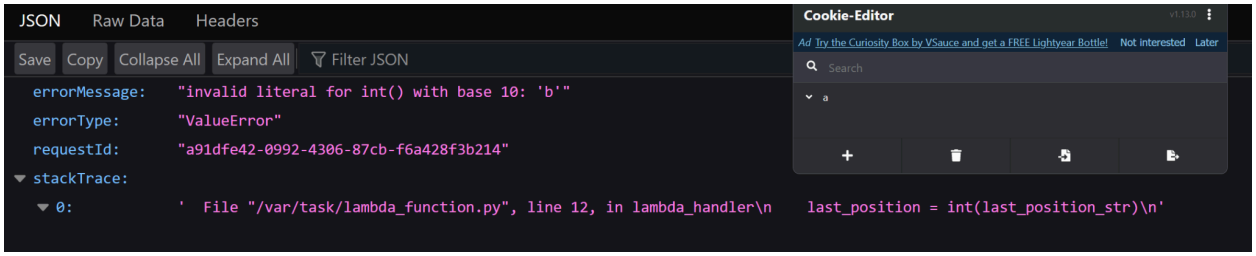
We can form url from out data.

https://h2vb2gupxd.execute-api.ap-south-1.amazonaws.com/prod/collectBits

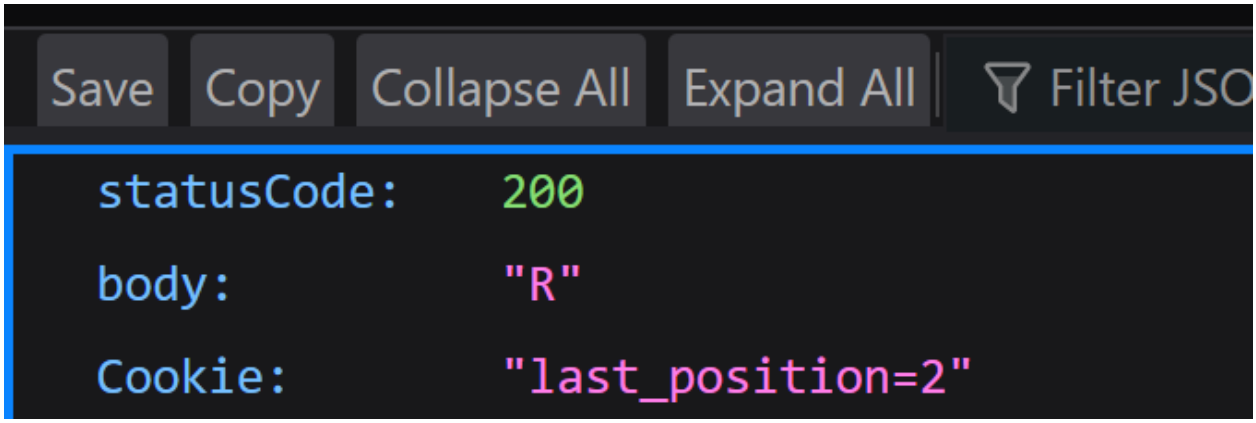
When we open it, It is expecting cookie in header.

```
{
  "errorMessage": "'cookie'",
  "errorType": "KeyError",
  "requestId": "e84855c1-4478-41a2-b055-ac82fa91f9ea",
  "stackTrace": [
    "  File \"/var/task/lambda_function.py\", line 10, in lambda_handler\n    cookie_header = event['headers']['cookie']\n"
  ]
}
```

So when we add normal cookie, It is expecting variable name: last_position as integer

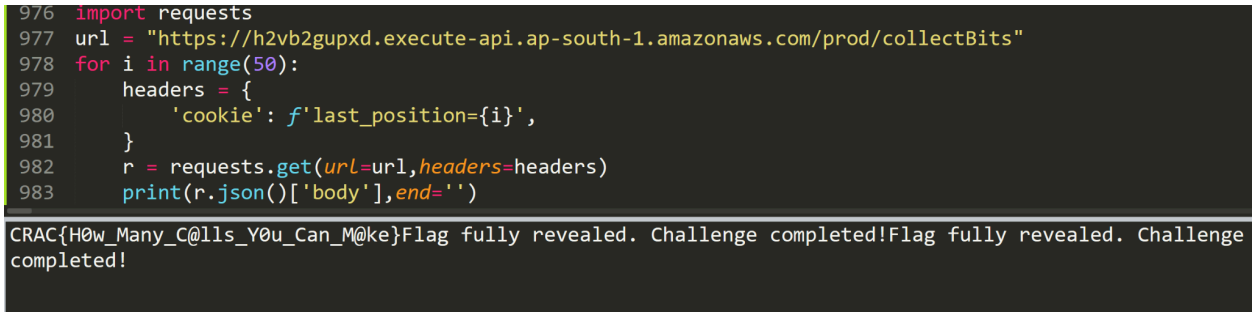


When we pass any number, the response is changing.



When i tried last_position=0, we get C.
So it might be flag values.
I wrote script for that.

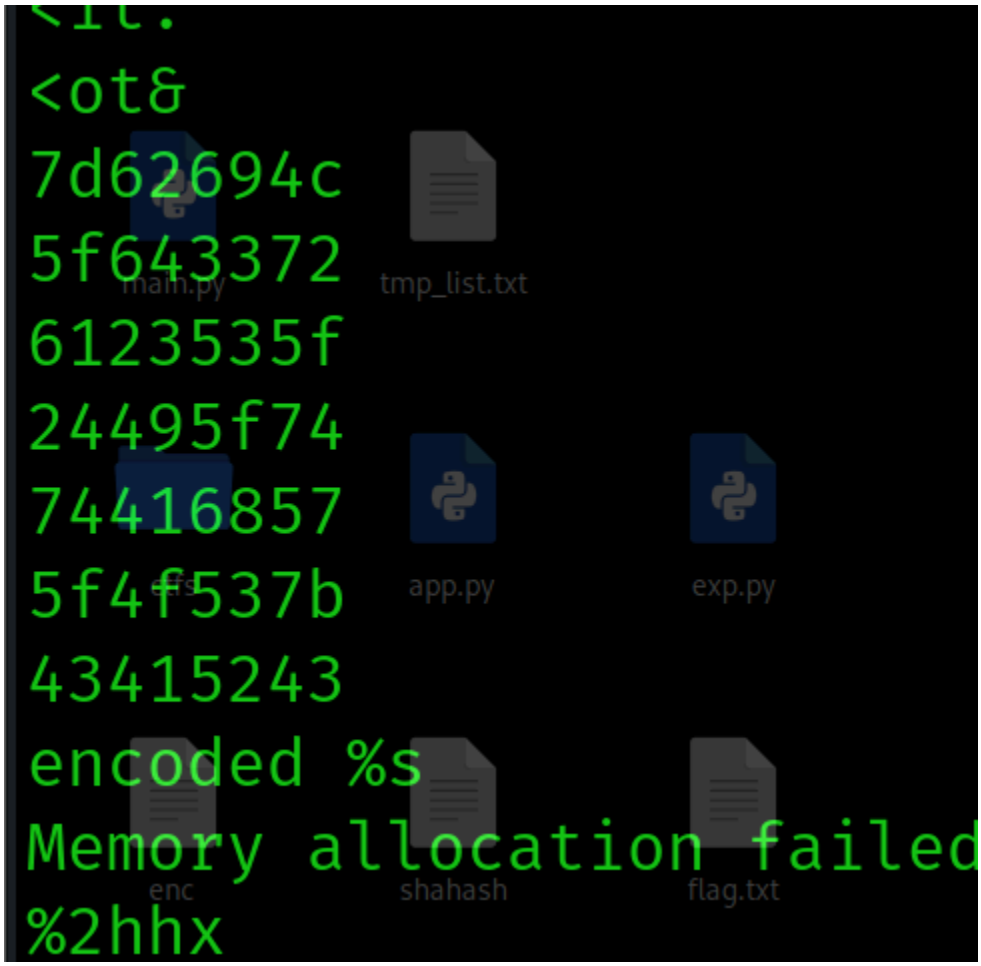
```
import requests
url = "https://h2vb2gupxd.execute-api.ap-south-1.amazonaws.com/prod/collectBits"
for i in range(50):
        headers = {
            'cookie': f'last_position={i}',
        }
        r = requests.get(url=url,headers=headers)
        print(r.json()['body'],end='')
```



Reverse Engineering / fruitslice

Attachment: libexternal.so

We get these hex values when we do strings,



Paste it in cyberchef and reverse.
https://gchq.github.io/CyberChef/#recipe=From_Hex('Auto')Reverse('Character')&input=N2Q2MjY5NGMKNWY2NDMzNzIKNjEyMzUzNWYKMjQ0OTVmNzQKNzQ0MTY4NTcKNWY0ZjUzN2IKNDM0MTUyNDM

Reverse Engineering / onecallaway

Attachment: passage

Upload in dogbolt.org

Hex-Rays

8.4.0.240320

```
153    v2[1] = 82;
154    v2[2] = 65;
155    v2[3] = 67;
156    v2[4] = 123;
157    v2[5] = 87;
158    v2[6] = 104;
159    v2[7] = 101;
160    v2[8] = 114;
161    v2[9] = 101;
162    v2[10] = 95;
163    v2[11] = 100;
164    v2[12] = 111;
165    v2[13] = 101;
166    v2[14] = 115;
167    v2[15] = 95;
168    v2[16] = 116;
169    v2[17] = 104;
170    v2[18] = 105;
171    v2[19] = 115;
172    v2[20] = 95;
173    v2[21] = 98;
174    v2[22] = 117;
175    v2[23] = 103;
176    v2[24] = 103;
177    v2[25] = 121;
178    v2[26] = 95;
179    v2[27] = 108;
180    v2[28] = 101;
181    v2[29] = 97;
182    v2[30] = 100;
183    v2[31] = 115;
184    v2[32] = 95;
185    v2[33] = 116;
```

We can decode these values using python or manually.

```
968   v1[37] = 3;"""
969
970 import re
971
972 l = re.findall(r"= ([0-9]{1,3});",a)
973 for i in l:
974     print(chr(int(i)),end='')
975
976 # b = """496e6143
```

```
CRAC{Where_does_this_buggy_leads_to}<0x00><0x04><0x05>
<0x0b><0x01>
<0x0e><0x0c><0x0f><0x14><0x15><0x13><0x18><0x16><0x17><0x12><0x19><0x1
<0x11><0x06><0x13><0x00><0x15><0x0f><0x17><0x19><0x08><0x05>%<0x14><0>
<0x0c><0x1a><0x01><0x04><0x12>  <0x10>
<0x1f> !<0x16><0x02><0x18><0x1b>"#$<0x11><0x03>[Finished in 0.4s]
```

```
a = """all those values you copied"""
import re
l = re.findall(r"= ([0-9]{1,3});",a)
for i in l:
        print(chr(int(i)),end='')
```

Cryptography / Secret

Here you go - take my secret shares! 1-567683, 2-570399, 3-115050. you can also take away this unique number 797077. Reveal the secret as CRAC{secret}

When we google secret shares, we get shamir's secret sharing.

So i made chatgpt to write script.

```
import numpy as np

shares = [(1, 567683), (2, 570399), (3, 115050)]

x = [point[0] for point in shares]
y = [point[1] for point in shares]

# Performing polynomial interpolation
```

```
coefficients = np.polyfit(x, y, len(shares) - 1)

# Creating a polynomial function
poly_func = np.poly1d(coefficients)

# Interpolating at x = 0
secret = poly_func(0)
print(secret)
```

We get output as: 106901.9999999976
We can round to 106902
Flag: CRAC{106902}

Adhoc / SanityCheck

Welcome to the CTF! Before diving into the complex challenges, let's ensure your setup is working correctly. Below is a string encoded in Base64. Decode it to reveal the flag. If you encounter any issues, double-check your decoding process or seek assistance from the organizers. REVGSEFXS3tZZXNfaXRfd29ya2VkfQ==

Decode the text from base64 to get flag.
This is most important challenge in all the CTFs.
It mostly tell us and organisers how dedicated and fast we are.
This is also a points boosting challenge sometimes.