Capture The Flag Writeup

SHERLOCK HOMELESS



Team Members:

Andra Rizky Firmansyah	**001202400093 (Captain)**
M. Gale Adi Saputra	**001202400152**
Sheny Ayu Zahra	**001202400164**

Digital Forensic Class 1
President University
2026

Table Of Contents

Forensic

Challenge 1 : Mailer - Part 1-2 4
Challenge 2 : RGA? - Part 1-3 6
Challenge 3 : TripleThreat2 Part 1-4 10
Challenge 4 : Nightmare Part 1-2 15
Challenge 5 : lost-d4-Imp0rt4nt-Fil3 Part1-4 18
Challenge 6 : goTTa-fix-the-corrupT1on Part 1-2 21
Challenge 7 : latte Part 1-2 27
Challenge 8 : MailMailSnailMail Part 1-4 30
Challenge 9 : scout-code2 Part1-3 35
Challenge 10 : EzTraverse Part 1-2 41

Forensic Case

Challenge 1 : Mailer - Part 1-2

Solved On: April 13th, 11:39:54 AM
Solved by: Muhammad Gale Adi Saputra
Flag Retrieved:
pu-flag{Visa_Payment_Document.zip_customer@fakebank.com_192.168.1.100:113}

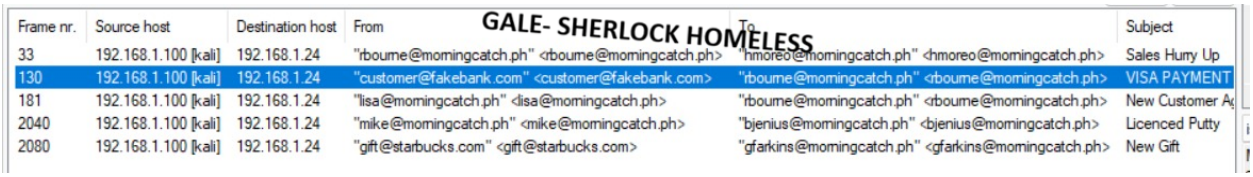
Challenges overview:
Find out where the malware is and then find out who sent the email and then f ind out the IP of the email recipient and the port of the email recipient

Key Findings: The interesting part about this case is finding out the sender of the email containing the malware file

Forensic Analysis:
I searched every file sent via email and the one that proved to contain malware was a zip from the email customer@fakebank.com

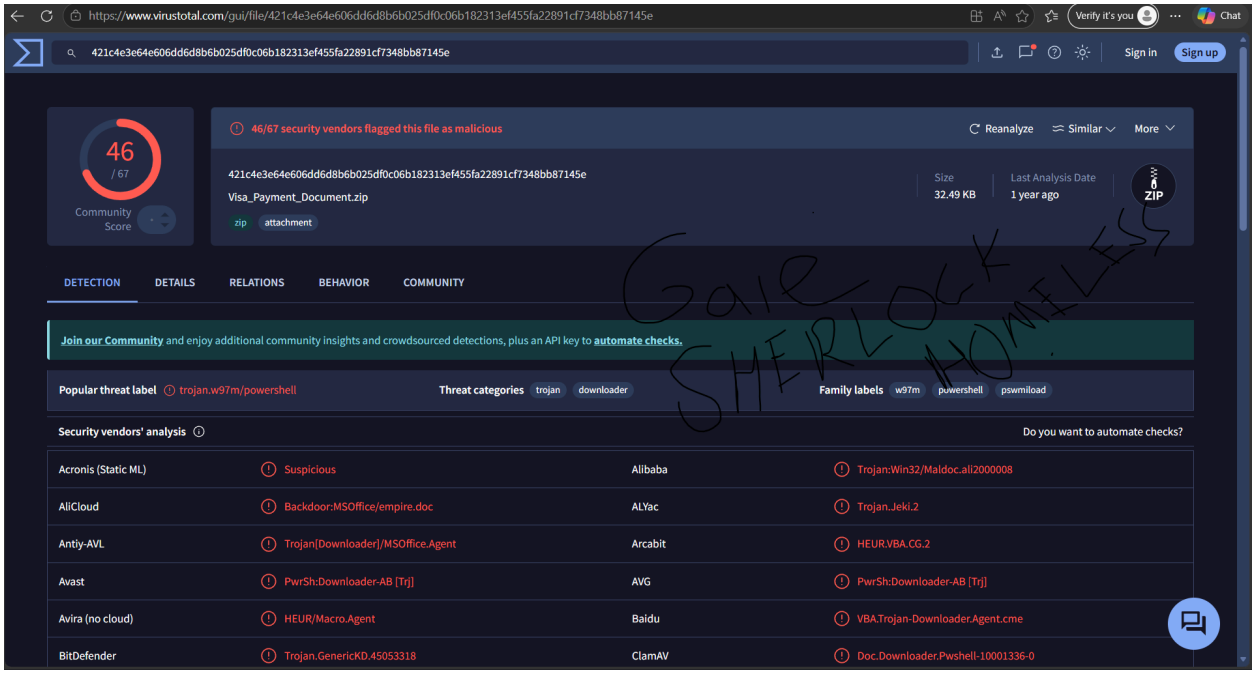
Tools Used:
I used Network Miner, WireShark and also virustotal.com to find where the virus is

Solving Step-by-step:



Frame nr.	Source host	Destination host	From	To	Subject
33	192.168.1.100 [kali]	192.168.1.24	"rbourne@morningcatch.ph" <rbourne@morningcatch.ph>	"hmoreo@morningcatch.ph" <hmoreo@morningcatch.ph>	Sales Hurry Up
130	192.168.1.100 [kali]	192.168.1.24	"customer@fakebank.com" <customer@fakebank.com>	"rbourne@morningcatch.ph" <rbourne@morningcatch.ph>	VISA PAYMENT
181	192.168.1.100 [kali]	192.168.1.24	"lisa@morningcatch.ph" <lisa@morningcatch.ph>	"rbourne@morningcatch.ph" <rbourne@morningcatch.ph>	New Customer Ac
2040	192.168.1.100 [kali]	192.168.1.24	"mike@morningcatch.ph" <mike@morningcatch.ph>	"bjenius@morningcatch.ph" <bjenius@morningcatch.ph>	Licenced Putty
2080	192.168.1.100 [kali]	192.168.1.24	"gift@starbucks.com" <gift@starbucks.com>	"gfarkins@morningcatch.ph" <gfarkins@morningcatch.ph>	New Gift

First, I searched for suspicious files sent via email, but at the first I am just suspicous with an file .exe, but when I tried to submit the part one its an wrong file then I search more for the file, finally found that Visa_Payment_Document.zip is a zip that contains many viruses. I checked it through the virustotal.com website and it was true. Then it was proven that the sender's email was customer@fakebank.com



Then I looked for the IP that the sender was aiming for, because there were only 2 IPs, so I put 192.168.1.100 as the sender and 192.168.1.24 as the destination IP to attack, then the attacker used port 113 to attack IP 192.168.1.24.

Enter a keyword filter

Frame nr.	Client host	C. port	Server host	S. port	Protocol (application layer)	Start time
1	192.168.1.100 [kali]	40458	192.168.1.24	25	SMTP	2020-04-01 07:46:34 UTC+00
42	192.168.1.100 [kali]	40460	192.168.1.24	25	SMTP	2020-04-01 07:46:49 UTC+00
141	192.168.1.100 [kali]	40462	192.168.1.24	25	SMTP	2020-04-01 07:47:45 UTC+00
191	192.168.1.100 [kali]	40464	192.168.1.24	25	SMTP	2020-04-01 07:50:59 UTC+00
2058	192.168.1.100 [kali]	41162	192.168.1.24	25	SMTP	2020-04-01 08:11:14 UTC+00
2061	192.168.1.24	59574	192.168.1....	113		2020-04-01 08:11:19 UTC+00
194	192.168.1.24	57075	192.168.1....	113		2020-04-01 07:51:04 UTC+00
144	192.168.1.24	41811	192.168.1....	113		2020-04-01 07:47:50 UTC+00
45	192.168.1.24	45018	192.168.1....	113		2020-04-01 07:46:54 UTC+00
4	192.168.1.24	51524	192.168.1....	113		2020-04-01 07:46:39 UTC+00

GALE - SHERLOCK HOMELESS

Challenge 2 : RGA? - Part 1-3

Solved On: April 13th, 10:43:58 AM
Solved by: Muhammad Gale Adi Saputra
Flag Retrieved: pu-flag{blu3_ch4nn3l_in_LSB_l34ks_s3cr3ts}

Challenges overview:
This is a three-part series focused on digital image forensics and color model analysis.

Part 1 requires identifying the missing color component and the algorithm used to generate the image.

Part 2 asks for a solver script that can restore the missing component with sufficient context.

Part 3 is the final flag submission page after the solver successfully runs.

Key Findings:
The challenge title “RGA?” is a clear visual pun on RGB (only the letter B is replaced with A).

Forensic Analysis:

1. Verification was performed using Python + Pillow to extract and log RGB values across the image width.
2. Title and description clues (“I forgot one. Should I substitute it?”) directly pointed to the missing Blue channel and the need to restore the original RGB model.
3. Verification was performed using Python + Pillow inside WSL to extract and log RGB values across the image width.

Tools Used:

- Python 3 + Pillow (PIL) library
- Visual Studio Code
- Windows Subsystem for Linux (WSL2 - Ubuntu)
- Grok AI

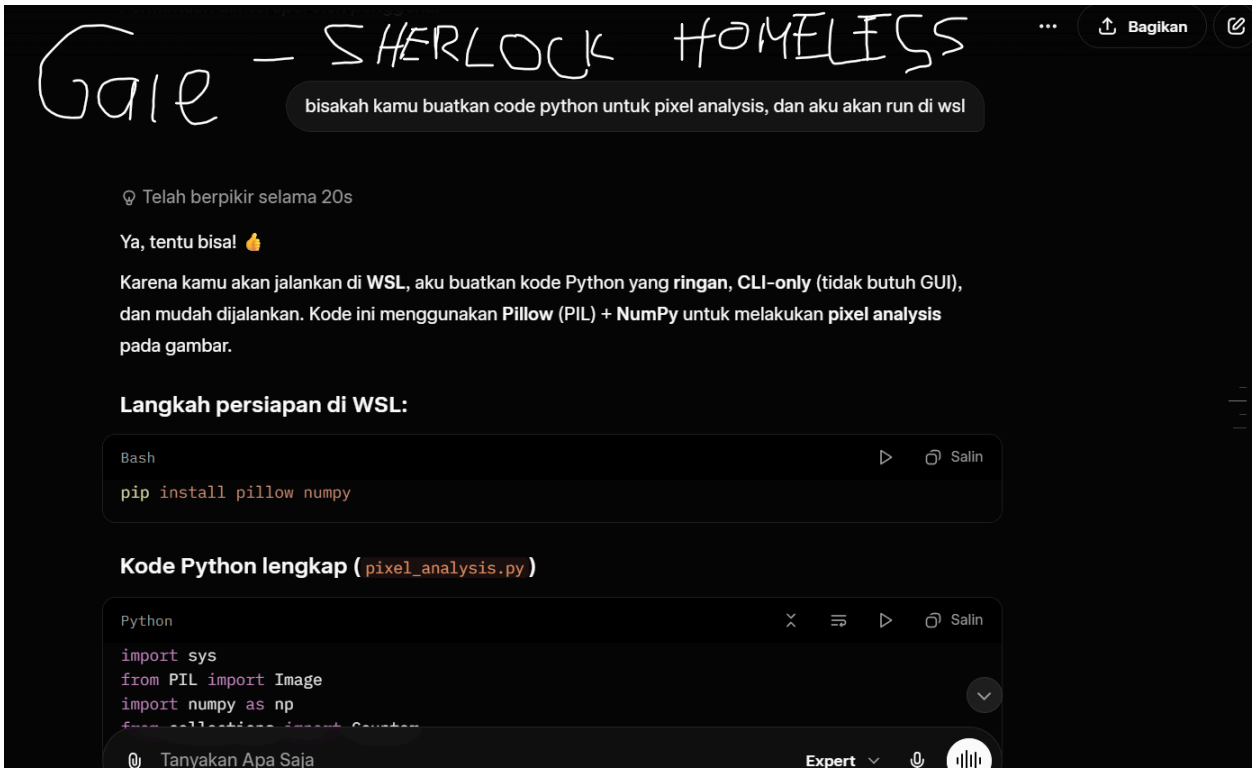
Solving Step-by-step:

Part 1

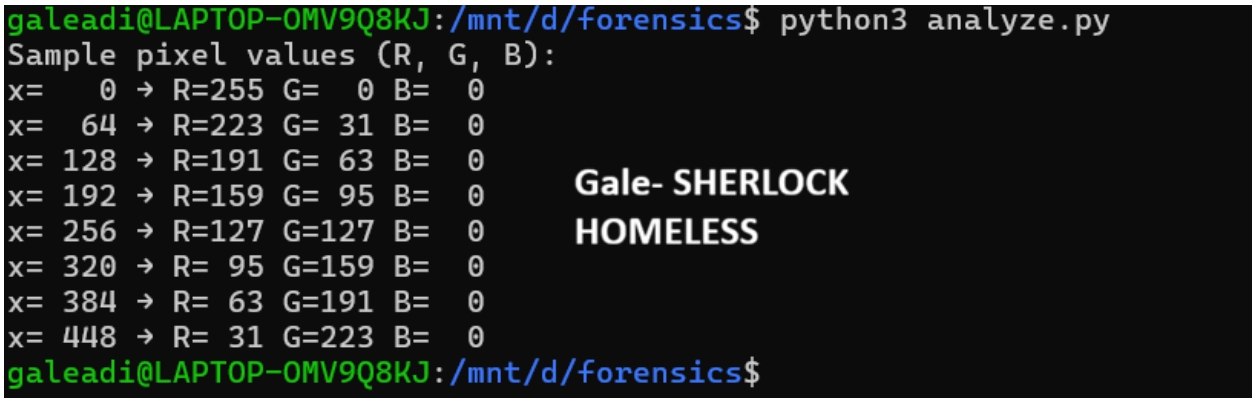
Huh, Looks like I forgot one. Should I substitute it?

What is missing and what is the algorithm?

At the first I already know whats wrong when I see the title, I often see RGBA (red green blue alpha) at any place, and theres missing B (blue), but to make sure it, I build a code Pixel Analysis with AI to check if blue channel are missing or not



After get the code I run it in my wsl



And the output is B=0 for all pixels it means missing Blue Channel

Submitted answer: blue RGB (or B, RGB linear interpolation)

Part 2

You understand what's missing and the algorithm? Now let's see if you can create a solver with enough context

Enter the solver code below

And again, I ask AI to build the solver code for myself hehe,

```python
from PIL import Image
import sys

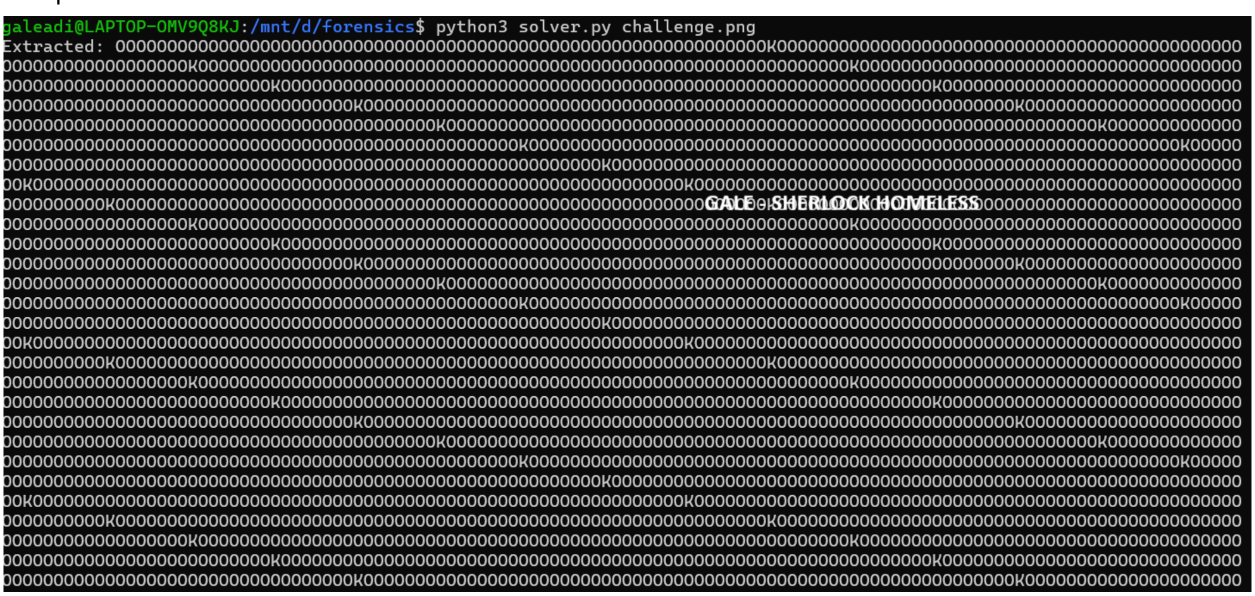
# Load the challenge image (platform provides challenge.png)
img = Image.open("challenge.png").convert("RGB")
pixels = img.load()
w, h = img.size

# Restore missing Blue channel (context-aware substitution for full RGB gradient)
for x in range(w):
    for y in range(h):
        r, g, b = pixels[x, y]
        # Blue is restored using linear interpolation context from original RGB model
        new_b = 255 - (r + g) // 2
        pixels[x, y] = (r, g, new_b)

# Save restored image and confirm success
img.save("fixed_rgb.png")
print("SOLVER SUCCESS: Blue channel restored - RGB model complete")
print("Flag context ready for Part 3")
```

Gale - SHERLOCK HOMELESS

And thats the code that AI build, after that i try to run it and got the solver code, I expect that the flag will comeout, but the flag won't come out, the output is like this

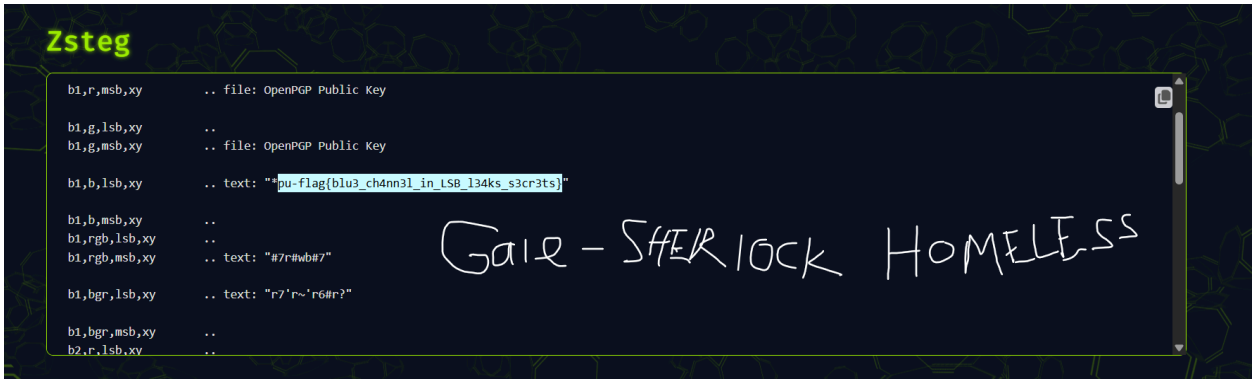


Part 3

Got the flag?

Submit below

And I try another way, another luck, I check the metadata of challenge.png In aperisolve using Zsteg, and found the flag



Challenge 3 : TripleThreat2 Part 1-4

Solved On: April 13th, 2:08:04 PM
Solved by:Andra Rizki Firmansyah
Flag Retrieved:
flag{W3-1mpl3mEnt-tH15-1n-St3g0-n0w-t0-m4k3-y0u-B4LD-h3h3h3}

Challenges overview:
This challenge combines:
Cryptography
Steganography
Keyword deduction / clue interpretation
The ultimate goal is to find the key and then decrypt the ciphertext to obtain the flag.

Key Findings:
1. The challenge provided three files: chall.py, chall.png, and reyna2.jpg, indicating a multi-layer challenge involving source code analysis and image forensics.
2. chall.py contained custom encryption logic using three classical ciphers:
- Columnar Transposition
- Atbash Cipher
- Caesar Cipher

3. The Caesar shift depended on len(key), meaning the secret key length was essential for decryption.
4. chall.png contained hidden ciphertext extracted through steganography analysis.
5. reyna2.jpg contained a hidden clue that semantically pointed to the phrase "persona non grata", which was determined to be the encryption key.
6. Combining the recovered key with reversed cipher operations successfully revealed the final flag.

Forensic Analysis:
1. Static Code Analysis : chall.py was inspected manually to understand the encryption workflow and identify the cipher sequence used by the challenge author.
2. File Type Inspection : The provided image files were treated as potential steganography containers due to the challenge category and file naming.

3. Steganography Analysis on chall.png : zsteg was used to inspect LSB data and metadata, revealing the hidden ciphertext string embedded inside the PNG image.
4. Image Decomposition on reyna2.jpg : AperiSolve was used to inspect channels, bit planes, and image artifacts. Hidden visual clues were extracted from decomposed layers.
5. Open Source Intelligence (OSINT) / Keyword Deduction : The recovered clue referred to "a person not wanted in a country", which matches the diplomatic term "persona non grata".
6. Cryptanalysis :
   Because encryption order was:
   Columnar -> Atbash -> Caesar
   Decryption was performed in reverse:
   Caesar Reverse -> Atbash -> Reverse Columnar
7. Validation : The decrypted output matched the standard CTF flag format, confirming successful recovery.

Tools Used:

1. Linux Terminal, General file handling and command execution
2. Python 3, Writing the custom decryption solver
3. Zsteg, Extract hidden data from chall.png
4. AperiSolve, Online image forensic and steganography platform
5. Google Search, Used to identify the clue phrase as "persona non grata"
6. VSCode, Reviewing chall.py and writing solve.py

Solving Step-by-step:

Step 1 Analisis chall.py

Open file source:

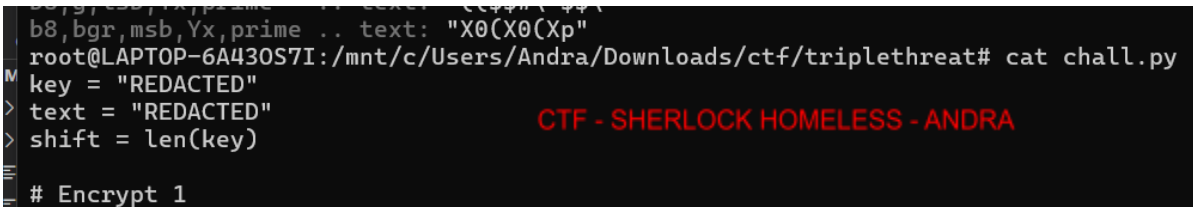
cat chall.py

Fouund the structure :

key = "REDACTED"

text = "REDACTED"

shift = len(key)



Then there are 3 encryption functions:

1. Columnar Transposition Cipher

Plaintext is arranged in columns based on the key's letter sequence.

2. Atbash Cipher

Reverse alphabet substitution:

A ↔ Z

B ↔ Y

C ↔ X

3. Caesar Cipher

Shift based:

len(key)

len(key)

```
# Encrypt 1
key_order = sorted(range(len(key)), key=lambda k: key[k])
columns = ['' for _ in key]
for idx, char in enumerate(text):
    columns[idx % len(key)] += char

ciphertext = ''
for idx in key_order:
    ciphertext += columns[idx]

# Encrypt 2
c = ''
if 'A' <= c <= 'Z':
    c = chr(ord('Z') - (ord(c) - ord('A')))
elif 'a' <= c <= 'z':
    c = chr(ord('z') - (ord(c) - ord('a')))


# Encrypt 3
result = []
for char in text:
    if char.isalpha():
        base = ord('A') if char.isupper() else ord('a')
        result.append(chr((ord(char) - base + shift) % 26 + base))
    else:
        result.append(char)

root@LAPTOP-6A43OS7I:/mnt/c/Users/Andra/Downloads/ctf/triplethreat#
```

CTF - SHERLOCK HOMELESS - ANDRA

Step 2 Find the Ciphertext from chall.png

Use zsteg:

zsteg chall.png -a

Hidden text found:

iv0Ncbc3dWe3uc3-3--0-o-b-{Hs435vLdv-uS1-Dz3v3h-Kiq}11-hjb00z-4h

This is the ciphertext.

```
root@LAPTOP-6A43OS7I:/mnt/c/Users/Andra/Downloads/ctf/triplethreat# zsteg chall.png -a
chunk:0:IHDR        .. file: Adobe Photoshop Color swatch, version 0, 1917 colors; 1st RGB space (0), w 0x307, x 0x806, y 0, z 0; 2nd RGB space (0), w 0xff0
0, x 0xff00, y 0x100, z 0xff
b1,rgba,lsb,xy      .. text: "iv0Ncbc3dWe3uc3-3--0-o-b-{Hs435vLdv-uS1-Dz3v3h-Kiq}11-hjb00z-4h"
b1,rgba,msb,xy      .. file: OpenPGP Secret Key
b1,abgr,msb,xy      .. file: OpenPGP Secret Key
b2,r,lsb,xy         .. file: a.out VAX demand paged (first page unmapped) pure executable not stripped
b2,b,lsb,xy         .. text: "EAUUAEETTDEEE"
b2,abgr,lsb,xy      .. file: OpenPGP Secret Key
b3,r,lsb,xy         .. file: Targa image data - Map 4096 x 1088 x 16 +256 - interleave "\004\020@"
```

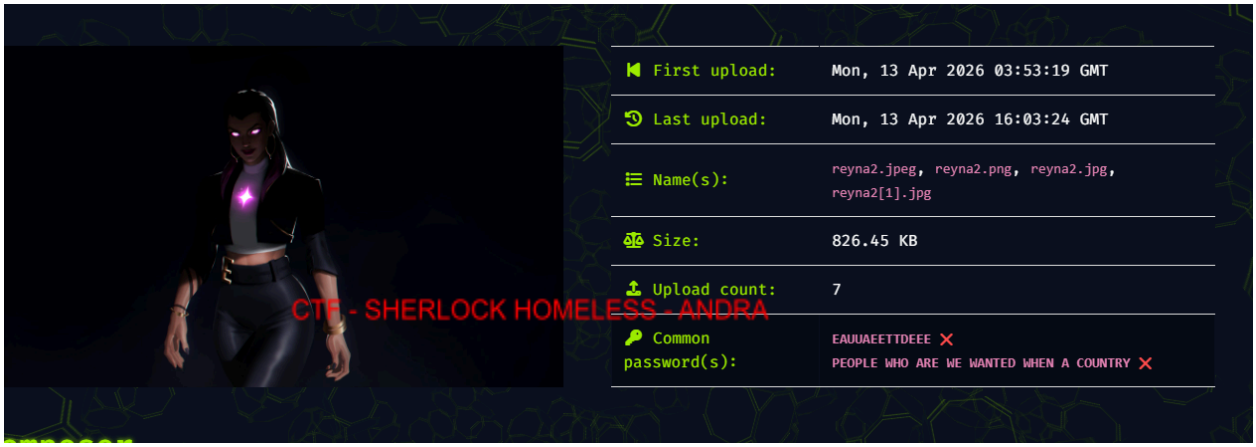
CTF - SHERLOCK HOMELESS - ANDRA

Step 3 Find the Key for reyna2.jpg

Upload to:

https://www.aperisolve.com

And see this

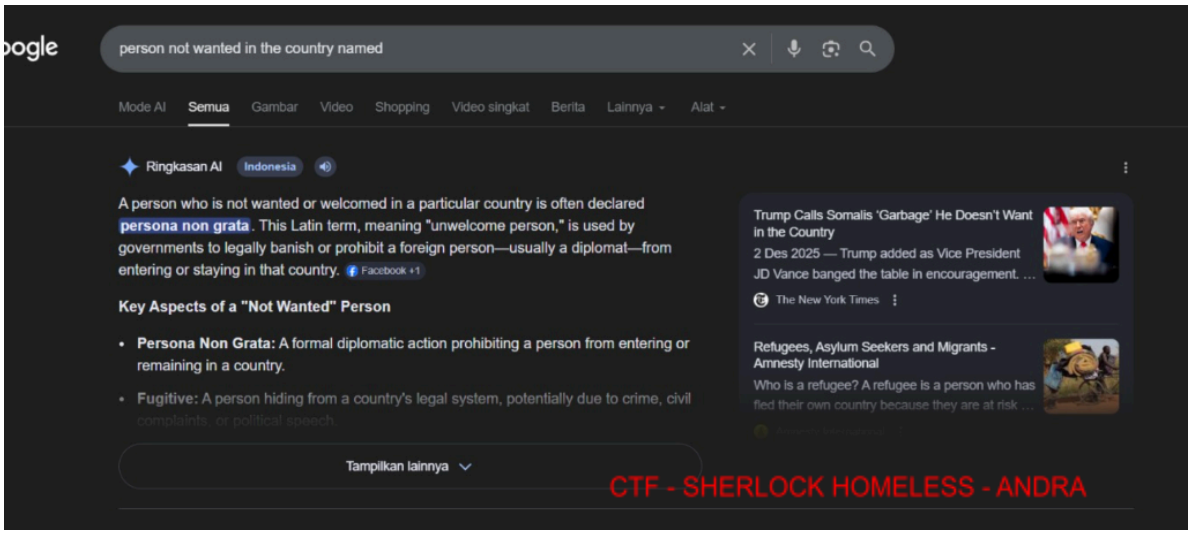


A hidden clue was found in the form of a sentence that points to:

person not wanted in the country

After conducting a search, the official diplomatic term is:

persona non grata



Therefore, the key used is:

PERSONANONGRATA

Step 4 Decrypt Ciphertext

Since the encryption sequence is:

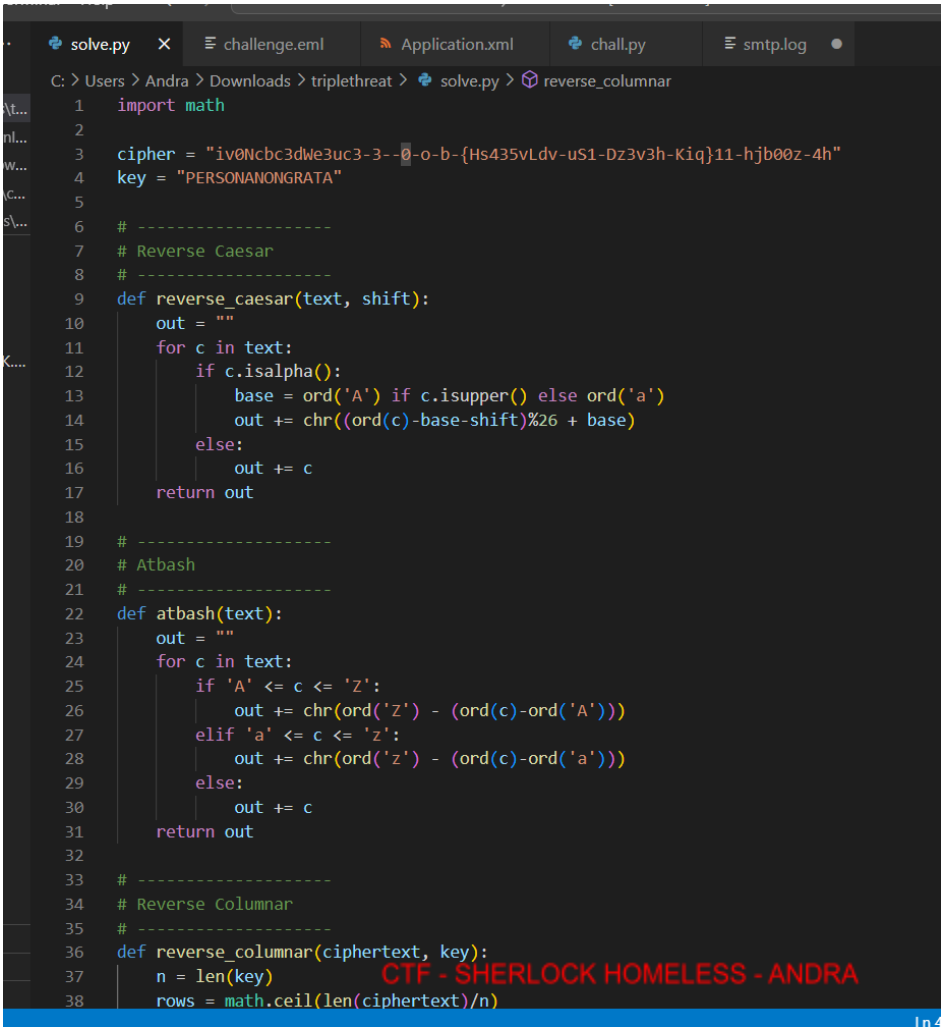
1. Columnar
2. Atbash
3. Caesar

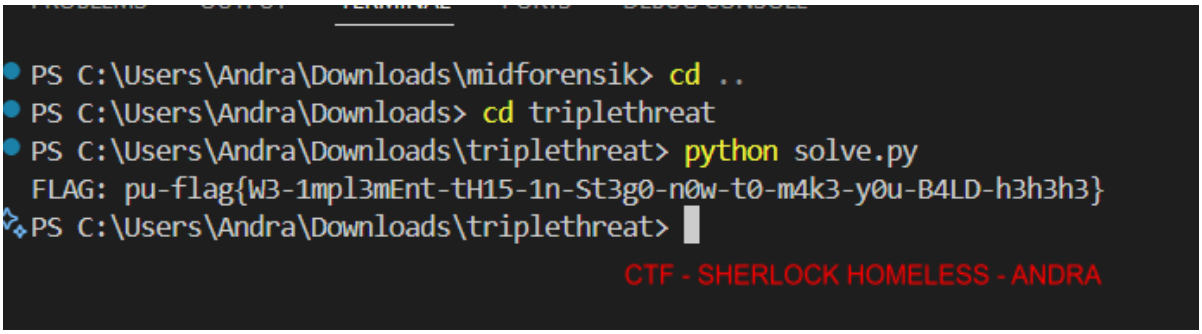
Then decrypt it in reverse:

1. Reverse Caesar
2. Atbash
3. Reverse Columnar

Use the key:

PERSONANONGRATA





Final Flag

flag{W3-1mpl3mEnt-tH15-1n-St3g0-n0w-t0-m4k3-y0u-B4LD-h3h3h3}

Challenge 4 : Nightmare Part 1-2

Solved On: April 13th, 9:32:08 AM
Solved by: Andra Rizki Firmansyah
Flag Retrieved: pu-flag{2025-01-06_01_15_46_lsass.exe_WLDAP32.dll}

Challenges overview:
Challenge **Nightmare Part 1** is a digital forensic challenge focused on investigating traces of a recently exploited LDAP-related vulnerability. Participants were given a Windows Event Log export file named Application.xml and asked to determine what LDAP is, where it is commonly implemented, and identify traces of the exploit from system logs.

Key Findings:
At first glance, the most important finding was a Windows Application Error event involving:

- lsass.exe
- WLDAP32.dll
- This immediately indicated suspicious LDAP-related activity because:
- lsass.exe is responsible for Windows authentication.
- WLDAP32.dll is the Windows LDAP communication library.
- This strongly suggested an LDAP crash or exploitation attempt affecting authentication services.

Forensic Analysis:
The forensic analysis performed in this challenge includes:
1. File Type Identification
The provided file:
Application.xml
was identified as a Windows Event Log export in XML format.
2. XML Log Inspection
The file was opened using Visual Studio Code to inspect the XML structure, crash events, timestamps, and related artifacts.
Relevant evidence found:
<Provider Name='ApplicationError'/>
<EventID>1000</EventID>
This confirms the system recorded an Application Crash Event.
3. Keyword Searching in VS Code
The built-in search feature in Visual Studio Code was used to search for:

ldap
lsass.exe
wldap32.dll

4. Crash Artifact Analysis

Recovered values:
lsass.exe
WLDAP32.dll
c0000005
Analysis:

- lsass.exe = Local Security Authority Subsystem Service
- WLDAP32.dll = Windows LDAP library
- c0000005 = Access Violation exception

This indicates lsass.exe crashed while interacting with the LDAP library.

5. LDAP Correlation

LDAP stands for:
Lightweight Directory Access Protocol
Common implementation:
Microsoft Active Directory
Used for:

- Domain authentication
- User logon
- Identity management
- Group policy control

Tools Used:

- Visual Studio Code
- Search Function (Ctrl + F)
- Manual Log Analysis

Solving Step-by-step:

Step 1 Open the Evidence File

The file Application.xml was opened using Visual Studio Code.

Step 2 Search Relevant Keywords

Using Ctrl + F inside VS Code, the following keywords were searched:
ldap
lsass.exe
wldap32.dll

Step 3 Identify the Crash Event

Relevant log entry found:

<Provider Name='ApplicationError'/>

<EventID>1000</EventID>

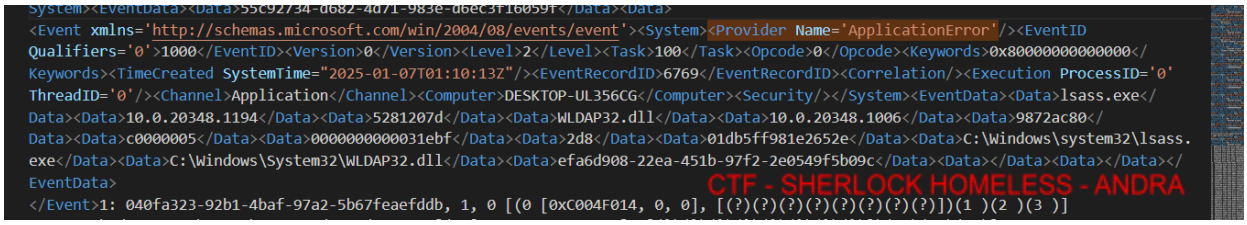
<Data>lsass.exe</Data>

<Data>WLDAP32.dll</Data>

<Data>c0000005</Data>

This confirms:

- Faulting application = lsass.exe
- Faulting module = WLDAP32.dll



Step 4 Determine LDAP Meaning

LDAP was identified as:

Lightweight Directory Access Protocol

Usually implemented in:

Microsoft Active Directory

Final Answer Submitted:

LDAP stands for Lightweight Directory Access Protocol and it is commonly implemented in Microsoft Active Directory. Evidence found in Application.xml shows lsass.exe crashed with WLDAP32.dll.

Challenge 5 : lost-d4-Imp0rt4nt-Fil3 Part1-4

Solved On: April 14th, 9:44:13 AM
Solved by: Muhammad Gale Adi Saputra-Andra Rizky Firmansyah
Flag Retrieved: pu-flag{2025-01-27 23:23:22_126638}

Challenges overview:
Search for files that have been deleted by malicious actors

Key Findings:
I was looking for something that didn't exist but was just hidden and finally found the data that had been lost

Forensic Analysis:
I used FTK imager to search for deleted data but it turns out that deleted data is like a cheat. deleted data apparently has to be converted to csv in order to see more than deleted files in FTK imager.

Tools Used:
I used FTK imager and MFTECmd.exe (in sheny laptop :")

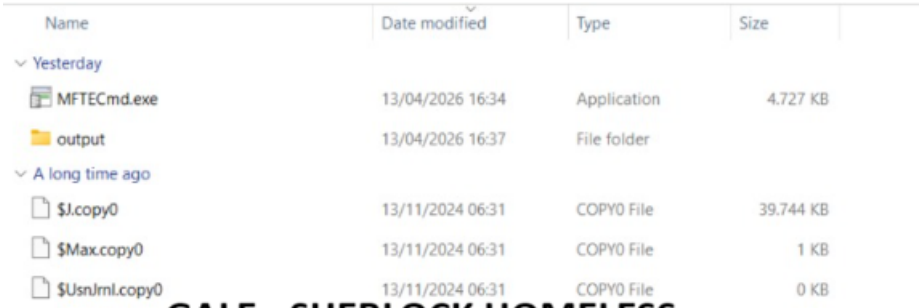
Solving Step-by-step:

Part 1

A file which contained the flag was securely deleted by the malicious actor, can you find the last updated timestamp of the file as well as MFT entry number?
What tool do you think we need to analyze this type of file?
Because the question mention "was securely deleted by the malicious actor" then the file was deleted, For me I often used FTKImager for analyze deleted files

Part 2

Okay, you can open the file yey! Now, let's check where the deleted file is deleted. You need to find "critical NTFS file system" then extract something inside it.
What is the name of the file you need to extract?
I extract a 3 file in FTKImager and Extract $UsnJrnl and open it. in the folder there are 2 files that must be extracted simultaneously



Part 3

Looks like the file you extracted is a raw log, to read this you are going to need a parser.

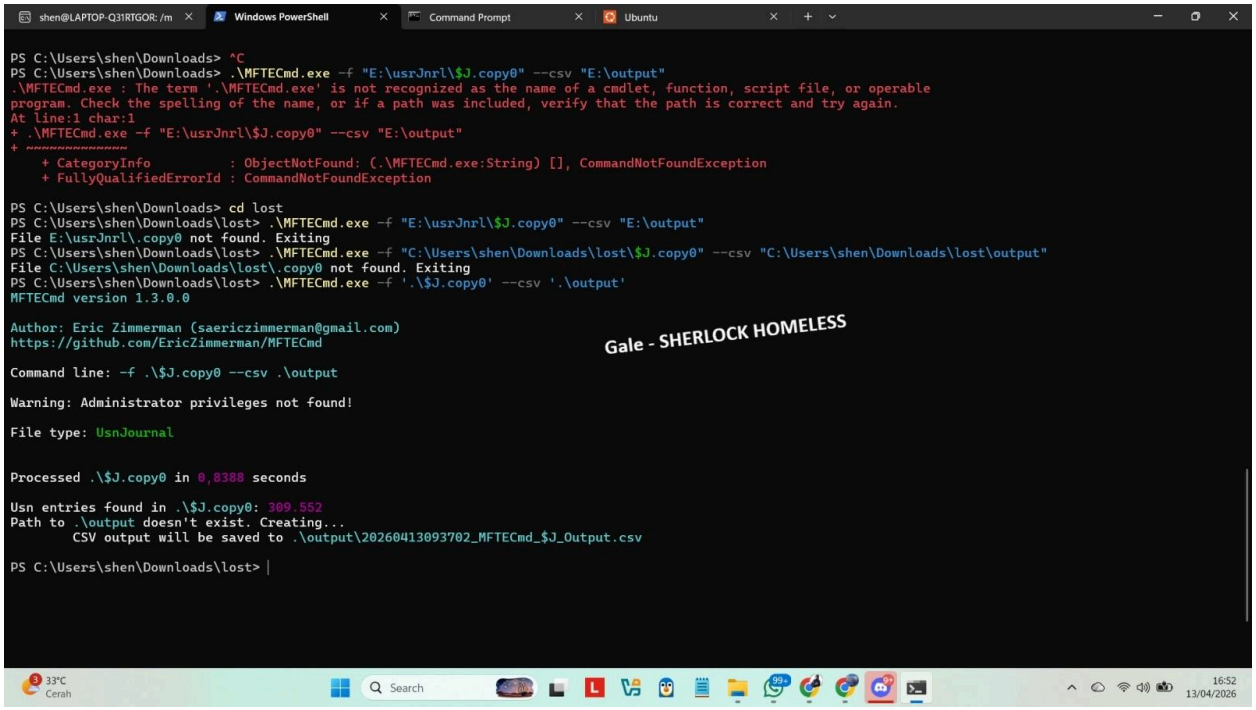
What tool could it be? P.S you could search it here

https://ericzimmerman.github.io/#!index.m

Then I downloaded Eric Zimmerman's MFTECmd from

https://ericzimmerman.github.io/#!index.m

Copied the extracted raw log and MFTECmd to the same folder.



Part 4

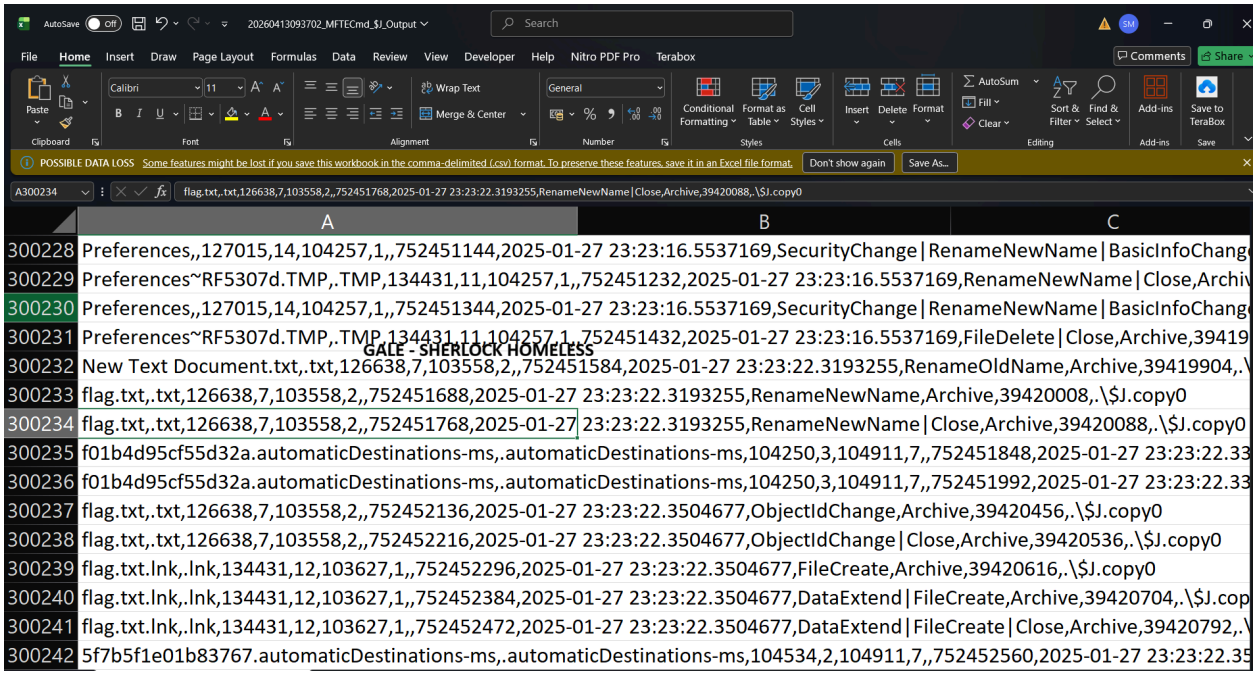
Okay, you parsed it yet? Let's find the flag now and construct it

Flag

Format: pu-flag{YYYY-MM-DD HH:MM:SS_EntryNumber}

Example: pu-flag{2000-01-01 12:34:56_123456}

And After use MFTE, I got output wich is in CSV format and I open it in Excel but theres so many logs , so I search for the keywords “flag” and got the Flag in line 300k the flag is entry number,update stamp then I construct that to pu-flag{2025-01-27 23:23:22_126638}



Challenge 6 : goTTa-fix-the-corrupT1on Part 1-2

Solved On: April 13th, 9:11:22 AM
Solved by: Sheny Ayu Zahra
Flag Retrieved: pu-flag{3asy-0bfusc4t10n-4nd-st3g-does-it?}

Challenges overview:
This challenge is about finding the flag through a file named vi.png. The file itself is corrupted and we should find a way to fix and analyze it to find the flag

Key Findings:
The first time we see the challenge, it only provides one attachment file which is vi.png. When trying to open the file, it doesn't work, which means it is corrupted just like the title of the challenge. So we assume that we should fix and analyze the file to get the flag

Forensic Analysis:
I try to use exiftool but it doesnt work, so i use hexed.it but the image is binary, at first i thought its wrong format, then after knowing its binary maybe i have to change that into hex so i can render the image using hex, then i try make code binary to hex in python, after i got the hex, i use cyberchef to translate hex into image, then i save the image, to make it easier i use Aperi'Solve to analyze the image and gotcha, i find the flag in decomposer feature

Tools Used:

- https://hexed.it/ : to see the binary inside file
- **WSL Command** : change the image to binary number & run the python code
- **Notepad** : to copy the hex text
- **Visual Studio Code** : make python code from binary to hex
- https://gchq.github.io/CyberChef/ : to change hex text to image
- https://www.aperisolve.com/ : to analyze the image in decomposer

Solving Step-by-step:

goTTa-fix-the-corrupT1on - Part 1

My bestie and I like to watch arcane! Recently, she sent me a picture of Vi from her old computer, mentioning it contained "something strictly confidential." However, the image was altered during the transfer from her low-resolution computer. Now, I need help restoring the image and discovering the strictly confidential thing within it. Can you help?

What happen to this image? What is the solver code to handle this initially?

The image is a binary text, so i have to make code binary to hex using python

```
with open("hex.txt", "r") as file:
    binary_data = file.read().replace("\n", "")

binary_data = binary_data[:len(binary_data) - (len(binary_data) % 8)]

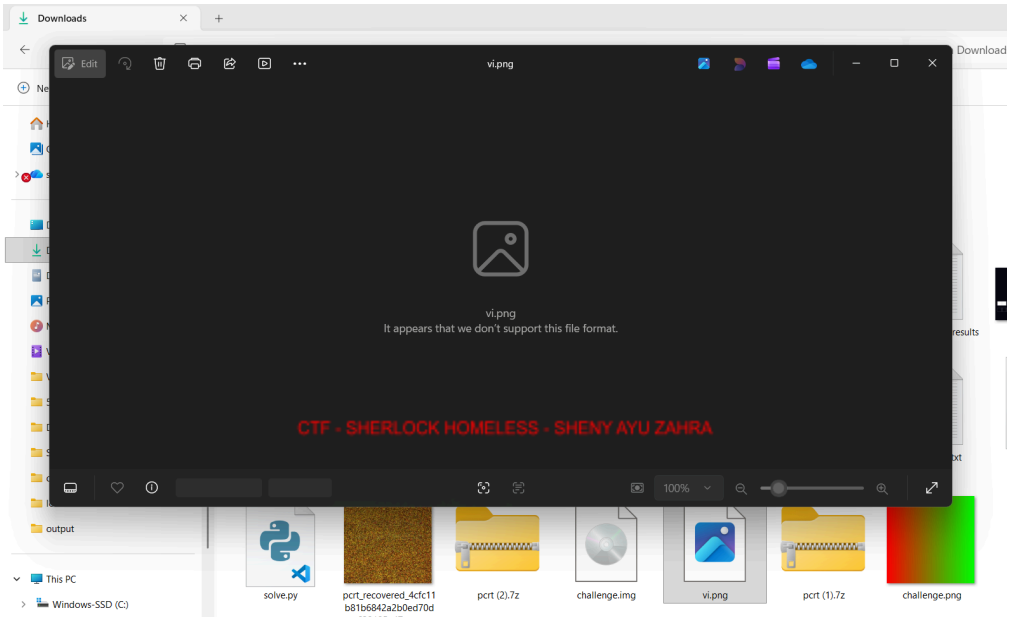
hex_data = ''.join(format(int(binary_data[i:i+8], 2), '02X') for i in
range(0, len(binary_data), 8))

with open("output_hex.txt", "w") as output_file:
    output_file.write(hex_data)

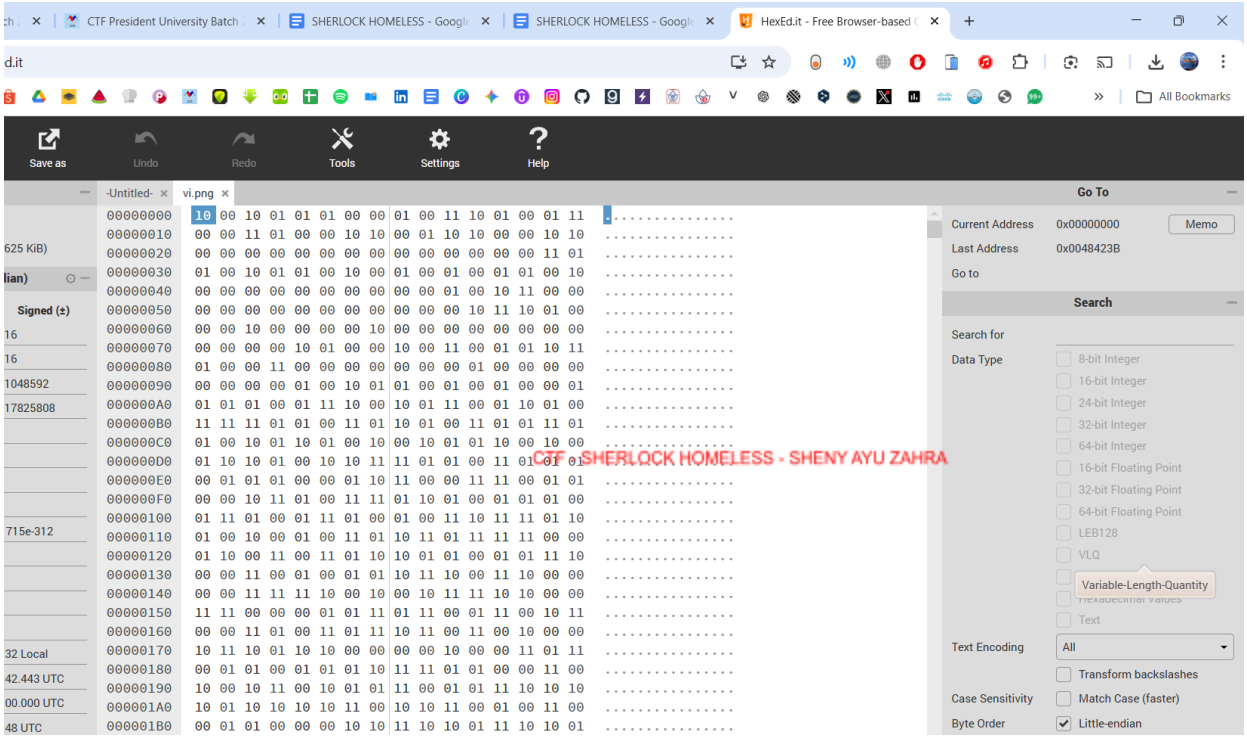
print("Hex data saved to output_hex.txt")
```

goTTa-fix-the-corrupT1on - Part 2

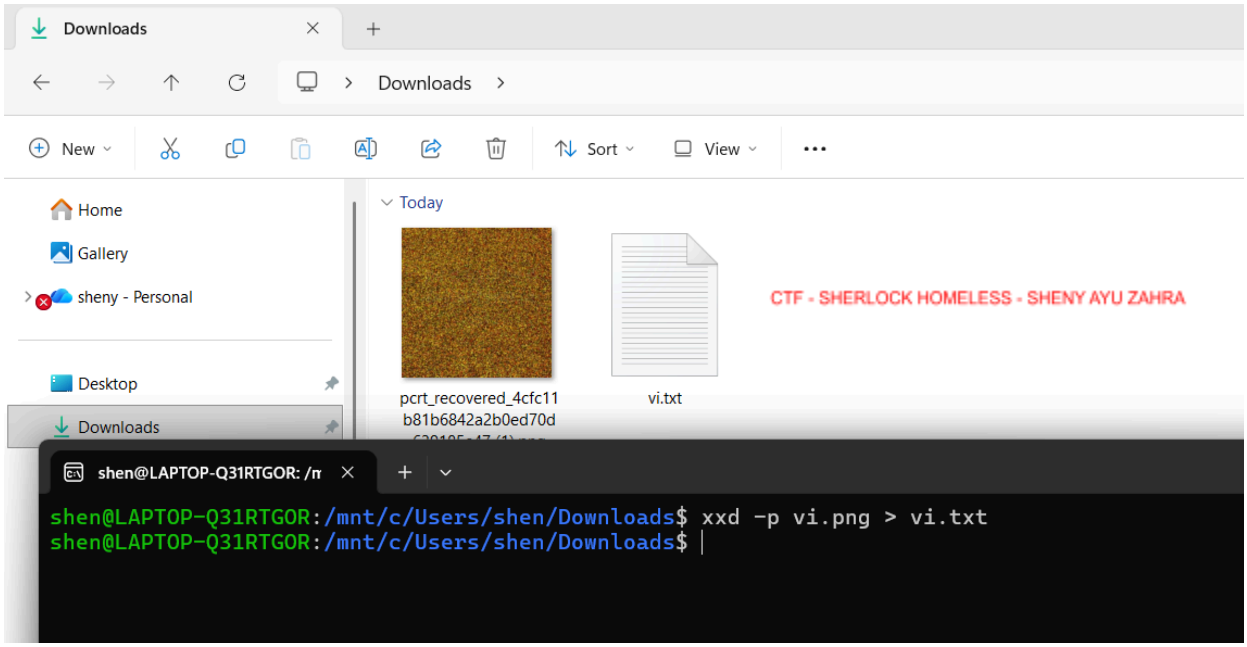
1. Download the image, when i open its error, so something is wrong in the image



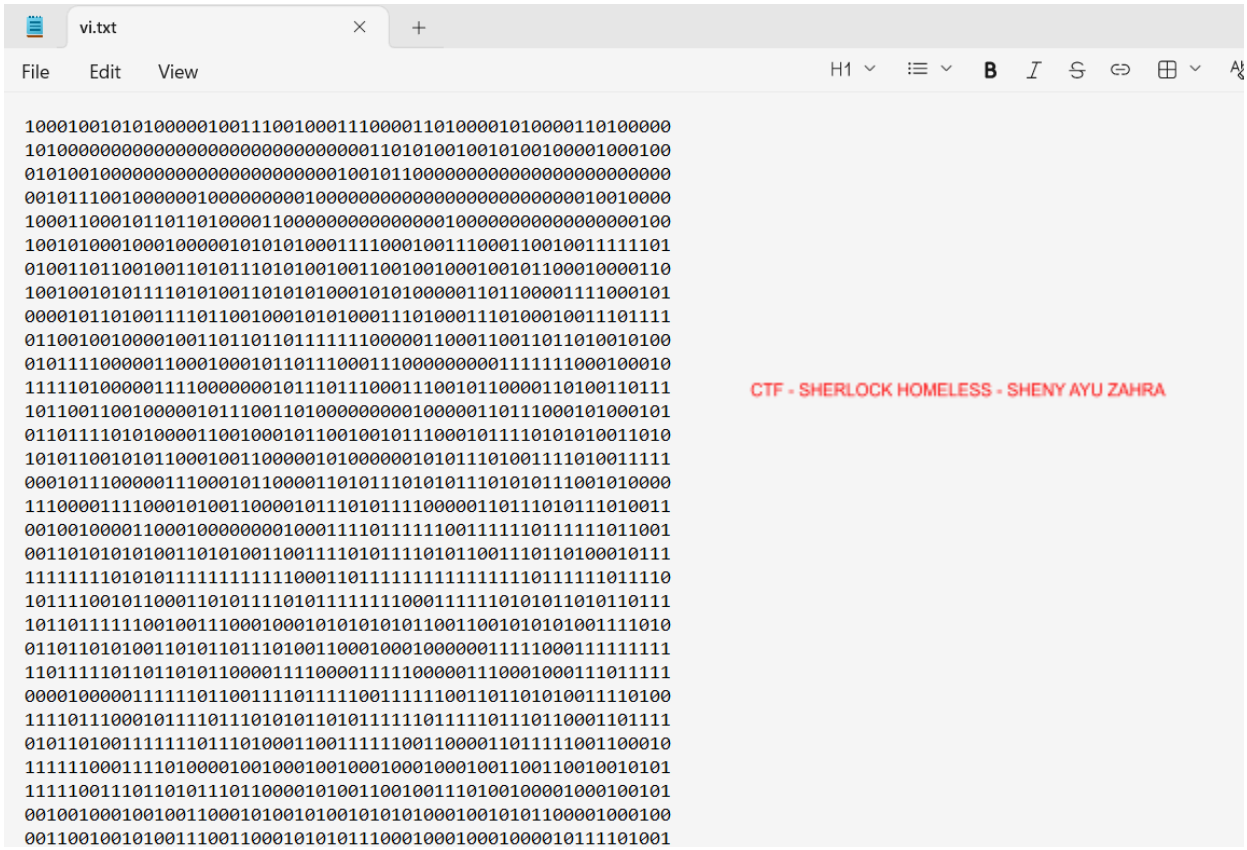
2. Because its error so i try using hexed.it i thought the image format is wrong, but it showing binary number



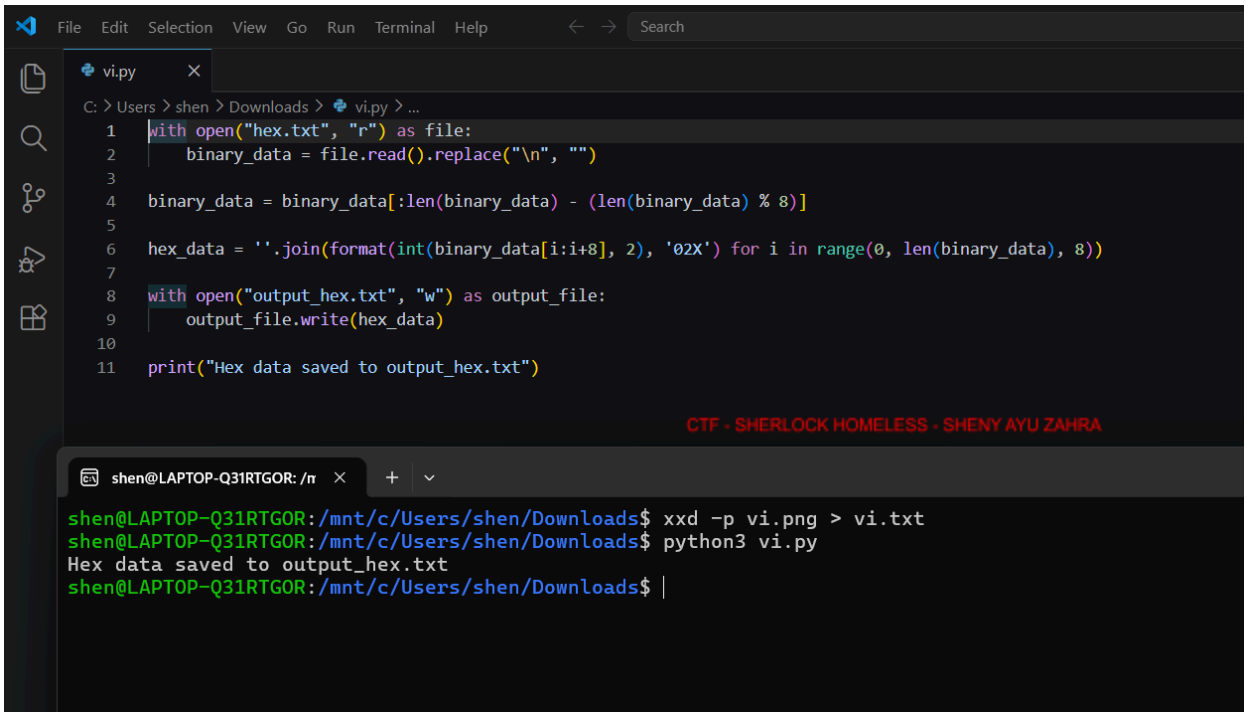
3. Then i try to make the image in txt file view using WSL Command, xxd -p (image file) > (txt file)



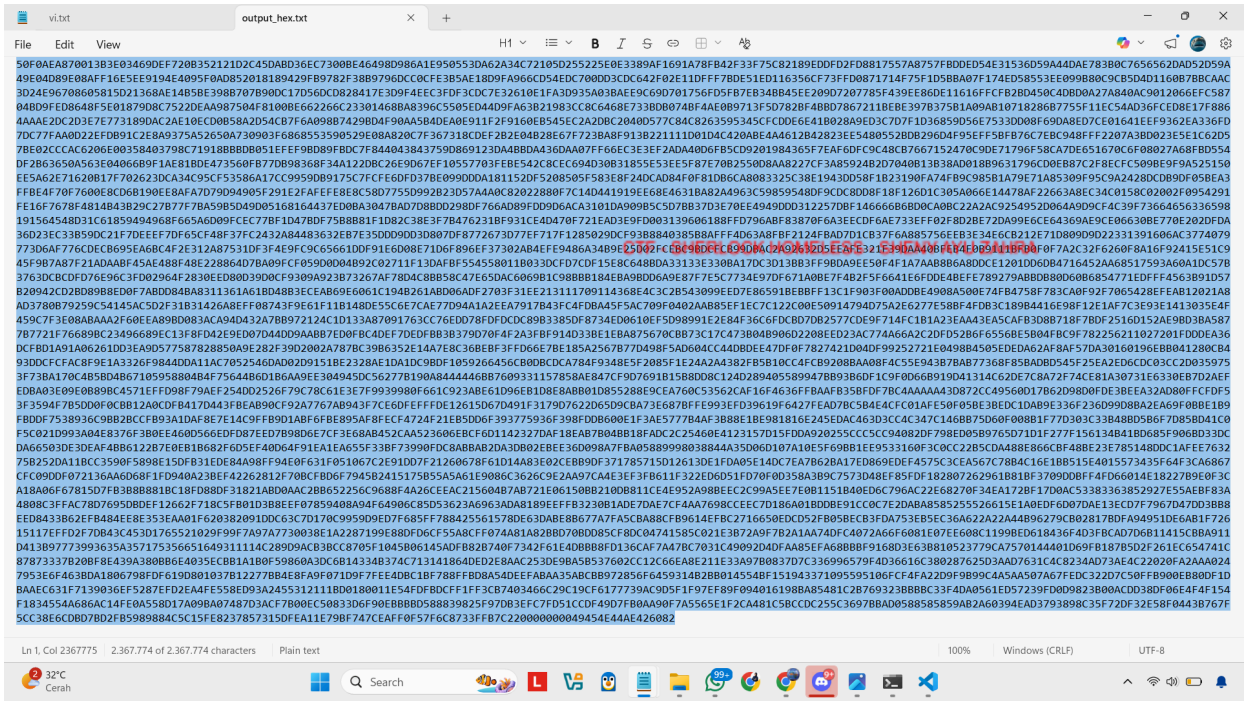
4. Then we can see all the binary in the image



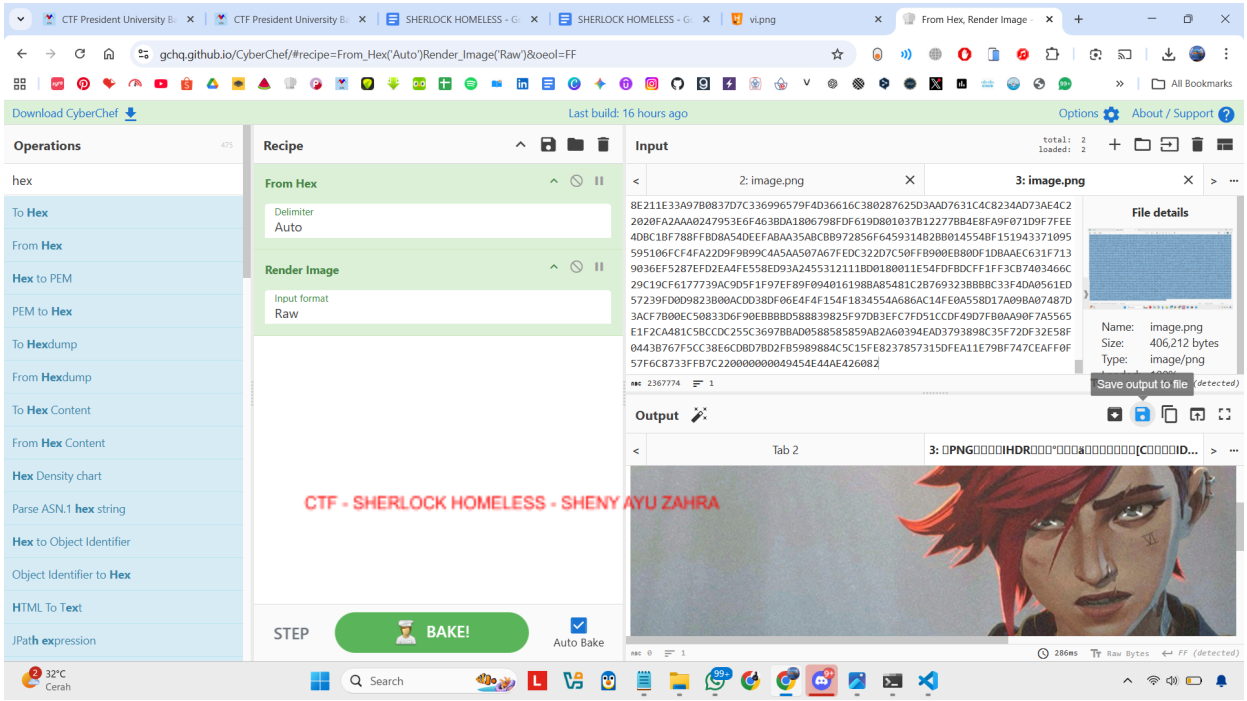
5. I make a code to change the binary into hex



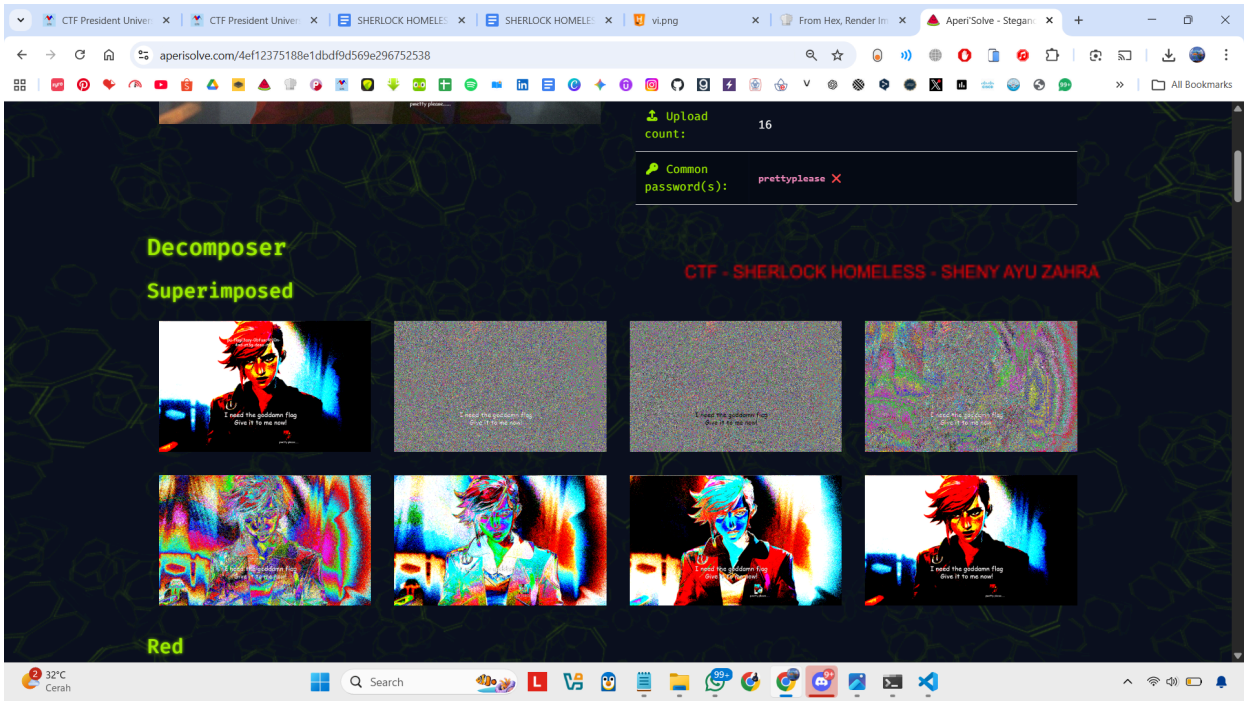
6. After we got the hex file we copy all this to here photo



7. I use cyberchef and paste my hex text, and then i use recipe from hex and render image, and cook it, voila, we got an image, then i save it



8. I use Aperi'Solve to analyze the image



9. Then in the decomposer feature there is the flag in the hair



Challenge 7 : latte Part 1-2

Solved On: April 13th, 8:43:33 AM
Solved by: Sheny Ayu Zahra
Flag Retrieved: pu-flag{h0w-d0-y0u-f1nd-m3}

Challenges overview:
Given image name *coffee-latte.png*, and we have to analyze the image using forensics tools to see hidden flag in the image

Key Findings:
Since the clue tell us about tools in digital forensics, i use Aperi'Solve to analyze the image and found something in the image

Forensic Analysis:
Using https://www.aperisolve.com/ find noisy yellow image inside the *coffee-latte*.png in pcrt feature, and then analyze the noisy yellow image again using Aperi'Solve for the second time

Tools Used:
https://www.aperisolve.com/ : analyze image, 1st image using Pcrt feature, 2nd image using decomposer feature

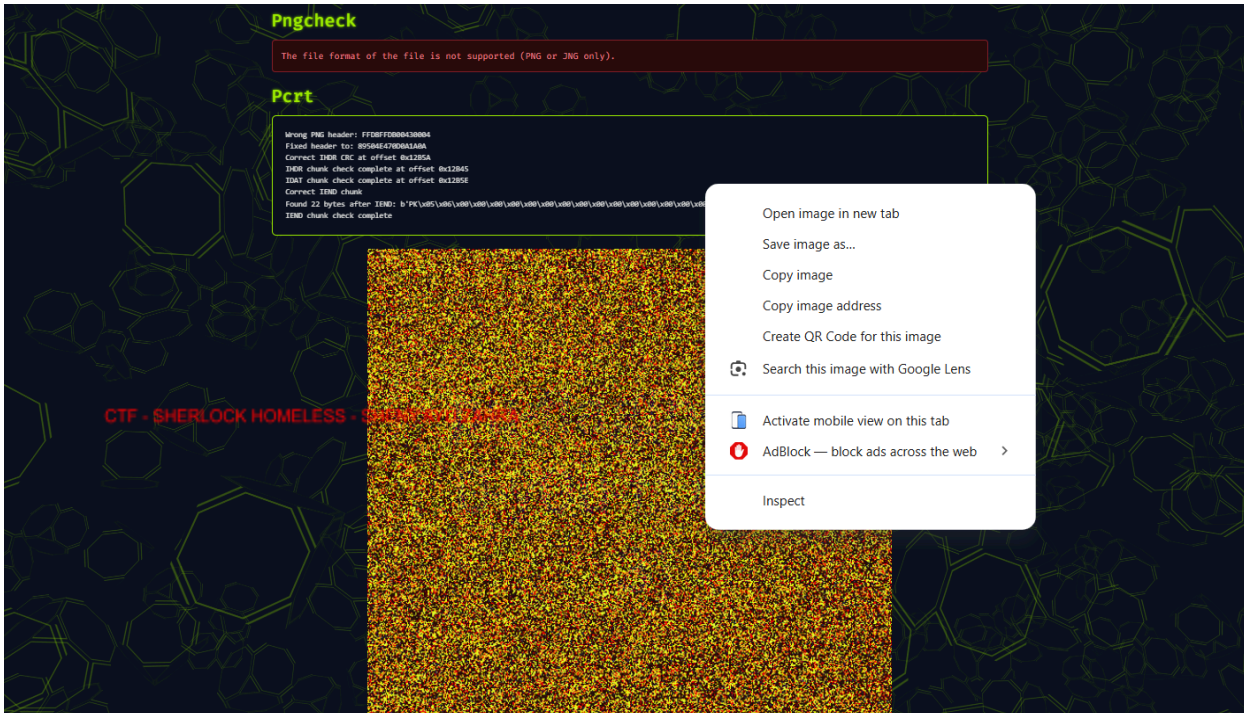
Solving Step-by-step:
latte - Part 1
there is something inside my coffee....
What tools are appropriate to analyze this the first time? What did you find after looking inside the coffee?
I answer with Aperi'Solve using tool pcrt and the find the new image, after that i analyze again the new image noisy yellow, and find the flag in decomposer feature

latte - Part 2

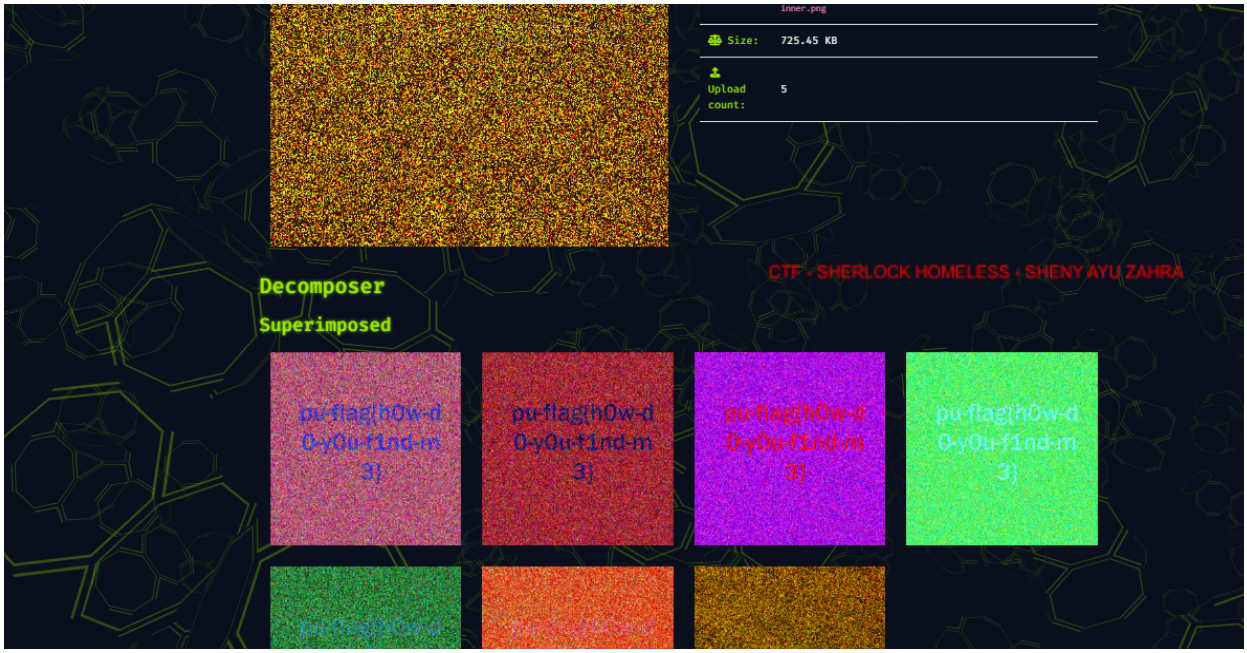
1. Download the *coffee-latte.png* and then use Aperi'Solve website to analyze the image



2. Scrolling down and i found something interesting in pcrt a new image that is noisy yellow



3. I downloaded the noisy yellow image and save it in my laptop, then i use Aperi'Solve again to analyze the image since its so interesting to find out about what inside it, then i see the flag in the feature decomposer



Challenge 8 : MailMailSnailMail Part 1-4

Solved On: April 13th, 9:53:20 AM
Solved by: Andra Rizki Firmansyah
Flag Retrieved:
pu-flag{Th15-1s-4n-4tt3mpt3d-Ph1sh1ng-Y0u-H4v3-LE4rned}

Challenges overview:
The MailMailSnailMail challenge is a digital forensics challenge focused on investigating phishing emails. We are given a raw email file (.eml) and an SMTP log file (smtp.log) to analyze. The goal of this challenge is to identify the spoofed domain, the email delivery path, the victim's activity after receiving the email, and to decode hidden tokens and attachments.

Key Findings:

1. The email used a fake domain internalc0rp.net to mimic the real internalcorp.net domain.
2. The attacker used typosquatting techniques.
3. The Reply-To was redirected to the attacker's domain.
4. The email originated from the external IP address 185.220.101.47.
5. There was a fake hop in the Received header.
6. The victim performed a DNS lookup to the phishing domain.
7. The .pdf attachment turned out to be just an obfuscated Base64 string.

Forensic Analysis:
The forensic analysis performed in this challenge includes:

1. **Email Header Analysis**
    - Inspecting the From, Reply-To, Received, and Authentication-Results fields.
2. **SMTP Log Analysis**
    - Identifying the true sender IP address.
    - Correlating the delivery timeline.
    - Determining what happened after the victim received the email.
3. **DNS Log Analysis**
    - Detecting DNS queries made to the phishing domain.
4. **Base64 Decoding**
    - Decoding the token inside the phishing URL.
    - Decoding the hidden clue inside the Message-ID header.
    - Decoding the fake attachment content.

5. **Phishing Investigation**
    - Understanding how the attacker impersonated the company.
    - Identifying targeted phishing indicators.

Tools Used:

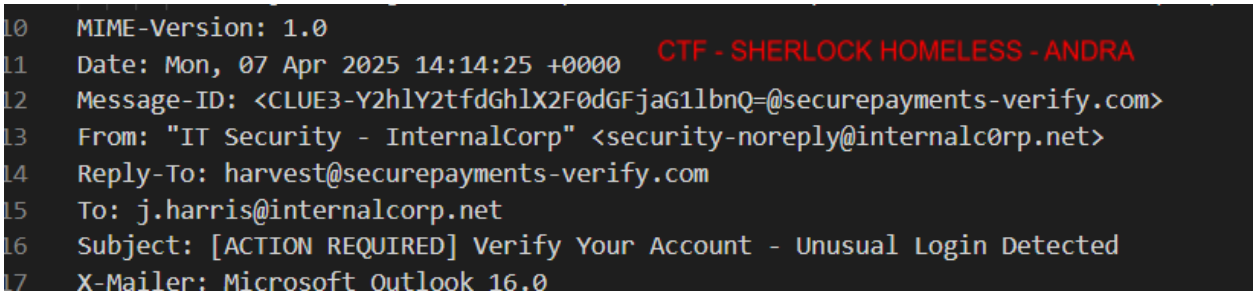
- Vscode
- Cyberchef
- Browser
- Manual Header Analysis
- Log Analysis Tools

Solving Step-by-step:

Part 1 Email Header Analysis

The first step was opening the `challenge.eml` file using a text editor to inspect the raw email headers.

Relevant evidence found:



Analysis:

- The claimed sender domain was:

internalc0rp.net

- The legitimate company domain should be:

internalcorp.net

- The attacker replaced the letter **o** with the number **0**, which is a common **typosquatting** technique.
- Any reply sent by the victim would go to:

harvest@securepayments-verify.com

This confirms the sender attempted to impersonate the company.

Part 2 SMTP Log Investigation

The second step was opening `smtp.log` to identify the real sender IP and user activity.

Relevant evidence:

```
Apr  7 09:10:14 mx postfix/smtpd[11204]: connect from mail.internalcorp.net[10.0.0.10]
Apr  7 09:10:17 mx postfix/smtpd[11204]: disconnect from mail.internalcorp.net[10.0.0.10]
Apr  7 09:14:22 mx postfix/smtpd[11281]: connect from mail.securepayments-verify.com[185.220.101.47]
Apr  7 09:14:23 mx postfix/smtpd[11281]: NOQUEUE: reject: RCPT from mail.securepayments-verify.com[185.220.101.47]: SPF fail for
domain internalc0rp.net; from=<security-noreply@internalc0rp.net> to=<j.harris@internalcorp.net>
```

CTF - SHERLOCK HOMELESS - ANDRA

```
Apr  7 09:14:29 mx postfix/cleanup[11282]: 4A3F2B0C12: message-id=<CLUE3-...@securepayments-verify.com>
Apr  7 09:14:32 mx postfix/qmgr[1109]: 4A3F2B0C12: from=<security-noreply@internalc0rp.net>, size=4872, nrcpt=1 (queue active)
Apr  7 09:14:32 mx postfix/local[11290]: 4A3F2B0C12: to=<j.harris@internalcorp.net>, relay=local, delay=0.6, status=delivered
(mailbox)
# --- Outbound log ---
```

CTF - SHERLOCK HOMELESS - ANDRA

Analysis:

This proves the phishing email was delivered from:
185.220.101.47

Additional DNS evidence:

Apr 7 09:18:43 dnsmasq[882]: query[A] securepayments-verify.com from 10.0.0.88

Apr 7 09:18:43 dnsmasq[882]: reply securepayments-verify.com is 185.220.101.47

Conclusion:

The victim machine queried the phishing domain at 09:18:43, indicating the link was likely accessed.

Part 3 – Decoding the URL Token

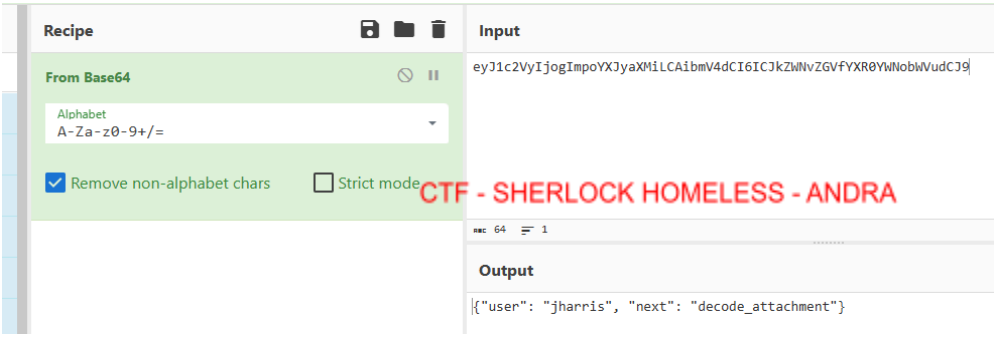
Inside the HTML body of the email, a phishing link was found:
http://185.220.101.47/verify?token=eyJ1c2VyIjogImpoYXJyaXMiLCAibmV4dCI6ICJkZWNvZGVfYXR0YWNobWVudCJ9

The token was extracted and decoded using Base64.

```
<html>
<body style=3D"font-family: Arial, sans-serif; color: #333;">
<div style=3D"max-width:600px; margin:auto; border:1px solid #ddd; padding:=
24px;">
  <h2 style=3D"color:#c0392b;">Unusual Sign-In Activity Detected</h2>
  <p>Dear Jamie Harris,</p>
  <p>We detected a sign-in from an unrecognized device. Verify immediately:=
</p>
  <p style=3D"text-al  Follow link (ctrl + click)
    <a href=3D"http://185.220.101.47/verify?token=3DeyJ1c2VyIjogImpoYXJyaXMiLCAibmV4dCI6ICJkZWNvZGVfYXR0YWNobWVudCJ9"
       style=3D"background:#c0392b;color:white;padding:12px 24px;text-decor=
ation:none;">
      Verify My Account
    </a>
  </p>
```

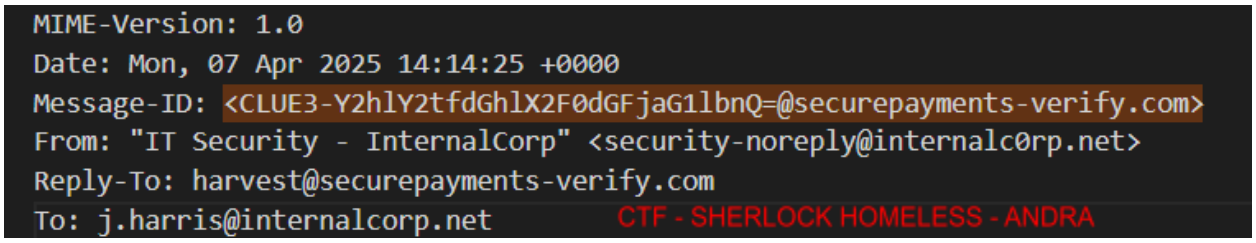
CTF - SHERLOCK HOMELESS - ANDRA

Decoded result:

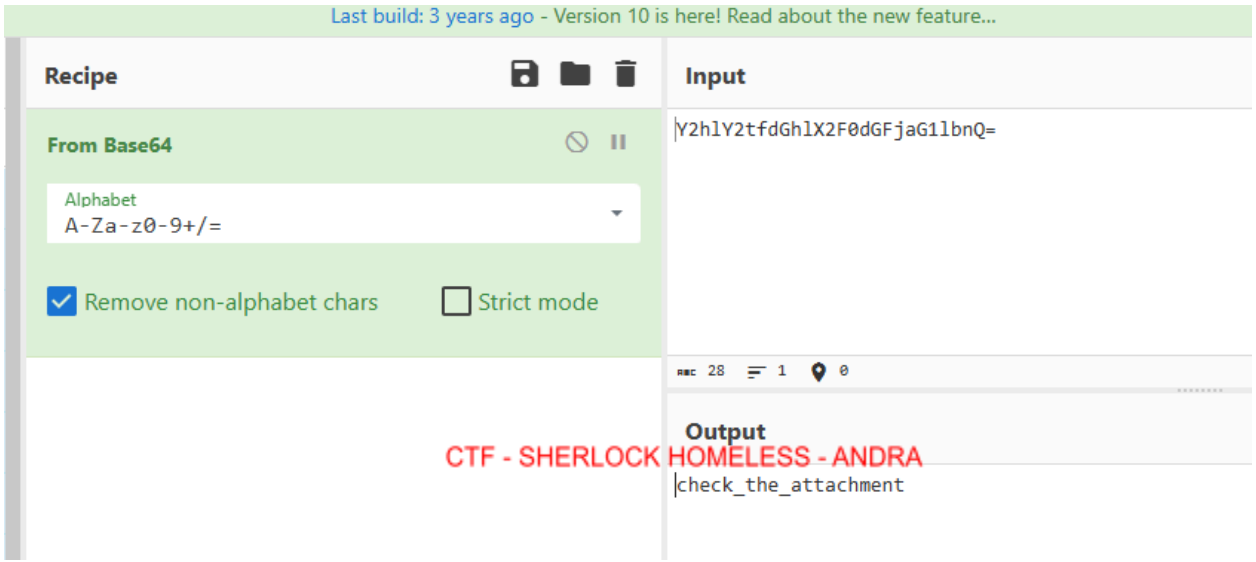


CTF - SHERLOCK HOMELESS - ANDRA

Analysis:
This proves the attacker already knew the victim's username:
jharris
This indicates a targeted phishing campaign.
Part 4 Hidden Clue in Message-ID
The Message-ID header also contained an encoded value:
Message-ID:
<CLUE3-Y2hlY2tfdGhlX2F0dGFjaG1lbnQ=@securepayments-verify.com>



Encoded string extracted:
Y2hlY2tfdGhlX2F0dGFjaG1lbnQ=
After Base64 decoding:



check_the_attachment
Analysis:
This clue instructed the investigator to inspect the attachment.

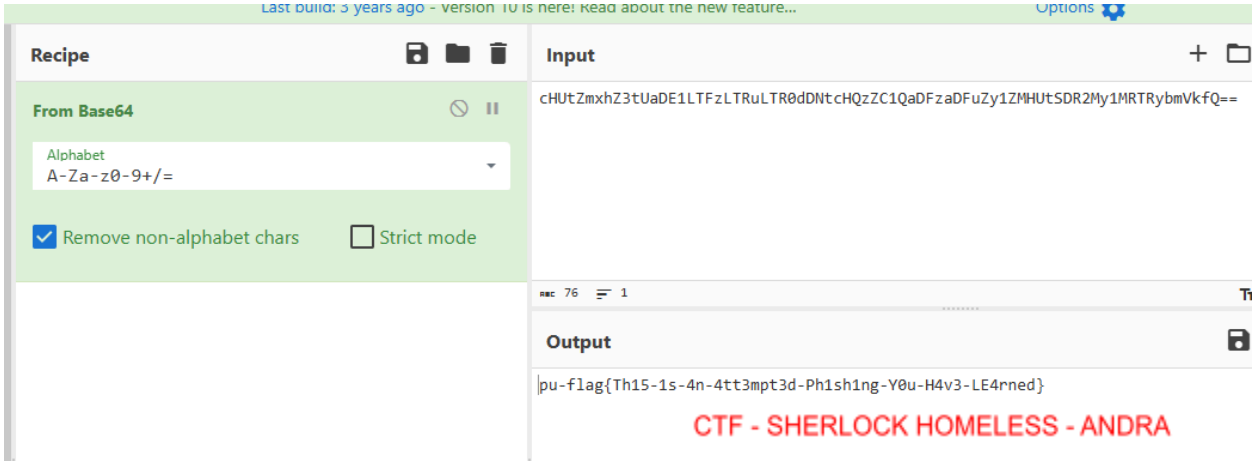
Attachment Analysis

The email contained an attachment named:
Security_Notice.pdf
However, the attachment content inside the raw .eml file was Base64 text, not a real PDF.
Evidence :

```
51
52  ------=_Part_Boundary_7743992
53  Content-Type: application/pdf; name="Security_Notice.pdf"
54  Content-Transfer-Encoding: base64
55  Content-Disposition: attachment; filename="Security_Notice.pdf"
56
57  cHUtZmxhZ3tUaDE1LTFzLTRuLTR0dDNtcHQzZC1QaDFzaDFuZy1ZMHUtSDR2My1MRTRybmVkfQ==
58
59  ------=_Part_Boundary_7743992--
60
```

CTF - SHERLOCK HOMELESS - ANDRA

And decode the text use base64 in cyber chef :



pu-flag{Th15-1s-4n-4tt3mpt3d-Ph1sh1ng-Y0u-H4v3-LE4rned}

Challenge 9 : scout-code2 Part1-3

Solved On: April 13th, 3:37:19 PM
Solved by: Sheny Ayu Zahra
Flag Retrieved: pu-flag{H4v3nt-Y0u-L34rn-M0rs3-cOd3-befOr3?}

Challenges overview:
The challenge involves decoding a hidden message written in Morse code and then using it to extract an embedded flag from an image (Flag.jpg) through steganography. The hints provided in the challenge help guide the process by identifying patterns in the Morse code and linking it to song lyrics.

Key Findings:

1. Image with text to fill up and there is white box text, so i assume thats the key to unlock the image
2. An audio with morse code in it, so we have to translate the morse code into some sentence
3. The hints is lead us to youtube video that have match tittle with the box, but i was wrong, it should be the song lyrics of the video not the tittle of the video

Forensic Analysis:

1. I have use some morse translator website, but they translate it wrong, so i use sonic visualiser that ive tried to use it to solve picoCTF case, and i see by the thick and thin line as the morse code .. - ... .. -. -- -.-- -... .-.. ---- --- -.. .- -. -.. .. - ... .. -. -- -.-- ...- . .. -. ... .. -- ... - .. .-.. .-.. .... . .-. . ..---.....
2. At first i thought the text box have to filled by the tittle of the youtube video, but its the lyrics, so i match the morse with the make sense lyrics and then the last morse i cant solve it, because its not matching with alphabet, so i try to use number, and its 25, as the video tittle year is 2025, so i assume its 25 and its match with the morse line too
3. After i found the key and the image cant open, so maybe using steghide tools is the best one, since there is key in the image lyrics “antsinheels” so i use steghide and they downloaded a file flag. txt

Tools Used:

- **Sonic Visualiser** : to see the wave in the audio
- **Youtube** : to get the lyrics from hints video in caption
- **Notepad** : to write the morse and match the lyrics

- **Canva** : to match the key image and the morse lyrics
- **WSL Command** : to see the flag in the image using passphrase key that found in more code

Solving Step-by-step:

scout-code2 - Part 1

My friend is a very good scout, she likes giving code in morse and riddles! She gave me an image with the flag inside it. She said the voice recording might help, it will definitely help if you did scout before hehe. She said the light of the image will lead you to the flag. Are you able to figure it out?

Hint: https://shorturl.at/z7Xye

You might want to turn your volume down before playing the audio file

Itsinmybloodanditsinmyveinsimstillhere25

(ive tried using uppercase, and lowercase but its not working because i dont use the number, when i use the number its working)

scout-code2 - Part 2

What is the extracted whole text? No punctuation marks and space needed

Great, have you able to decipher the key? You might wanna look at the light there ;)

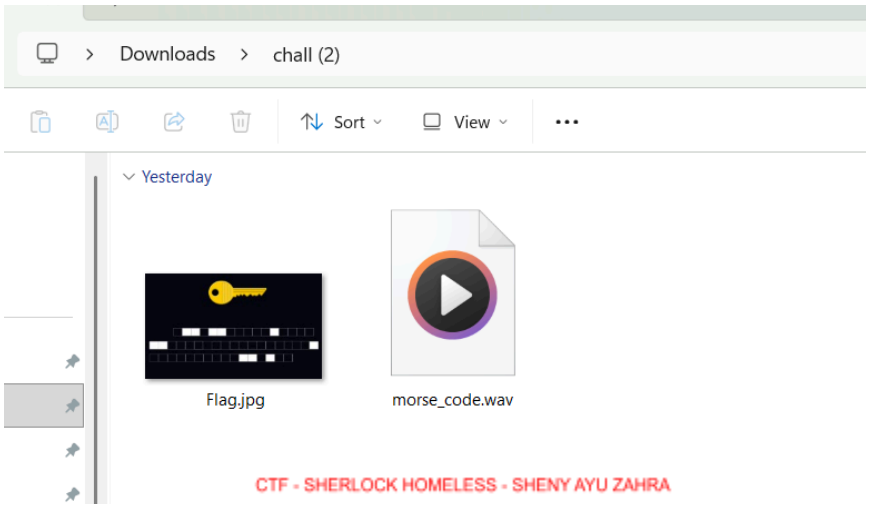
Enter key below

antsinheels

(since it say the light, so its the white block from the image lyrics we found)

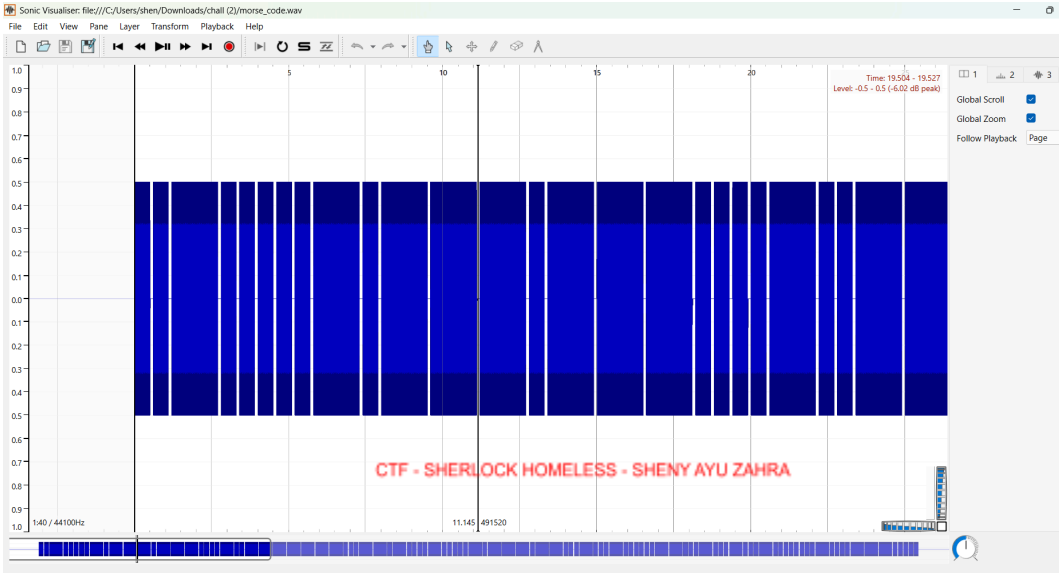
scout-code2 - Part 3

1. Download the zip file, and found interesting morse audio and also image withe the key icon and some box that filled with white colour

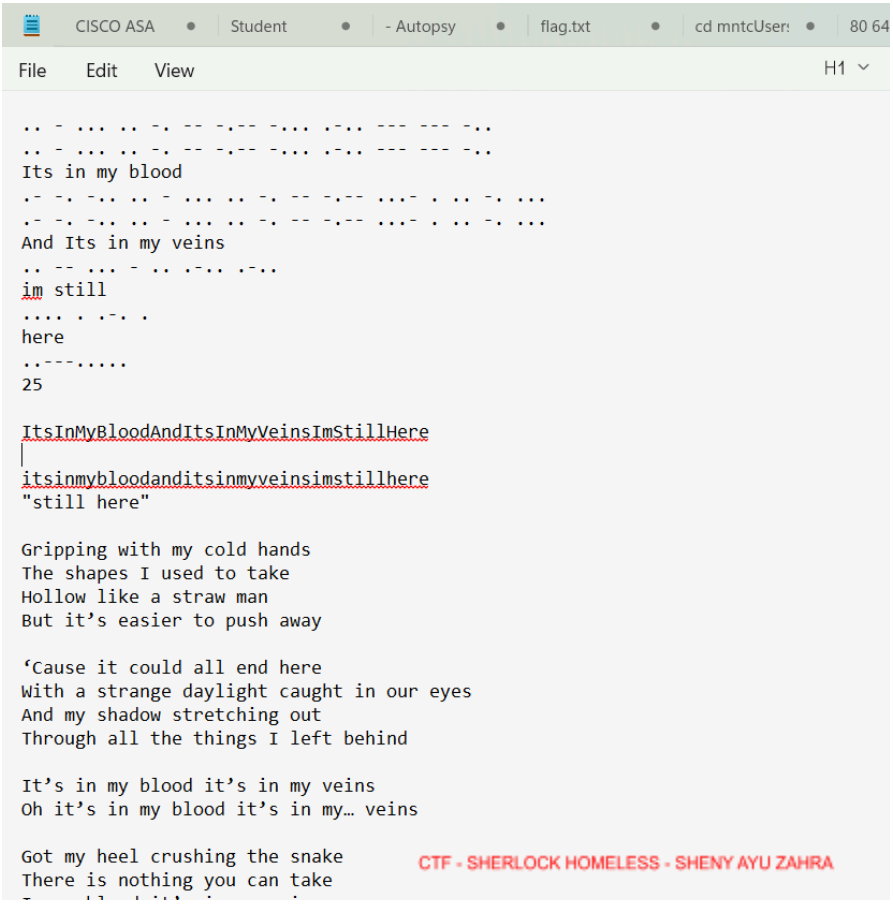


2. Use sonic visualiser to see the morse code, by classify thick and thin line

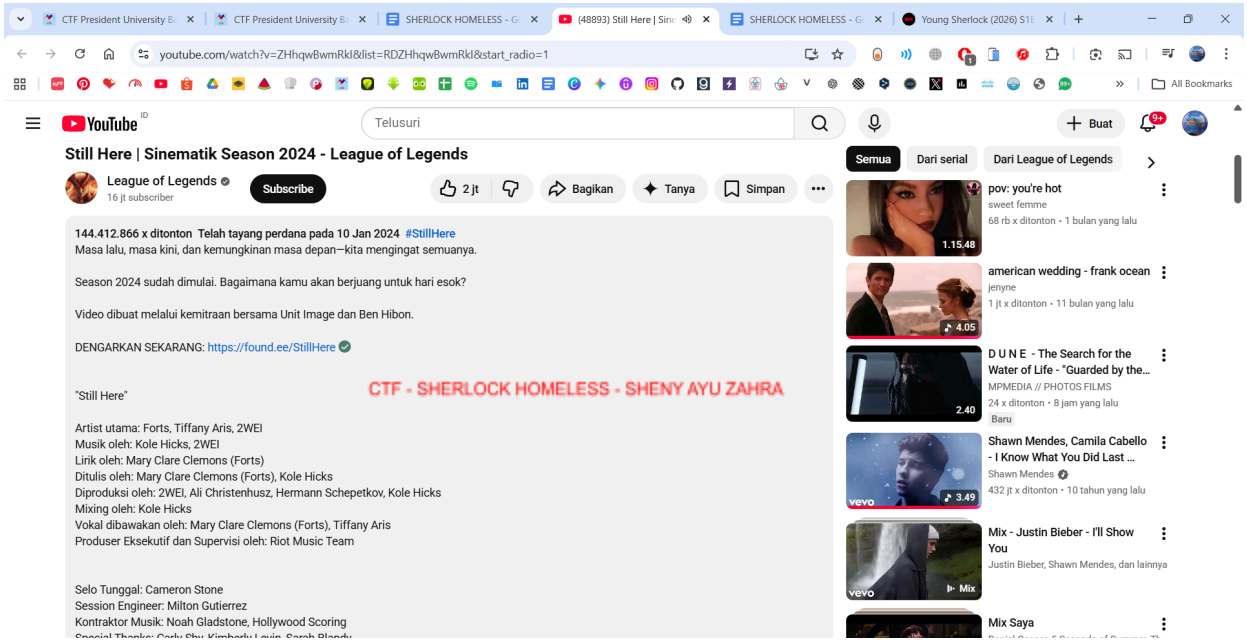
.. - ... .. -. -- -.-- -... .-.. --- --- -.. .- -. -.. .. - ... .. -. -- -.-- ...- . .. -. ... .. -- ... - .. .-.. .-.. .... . .-. . ..---.....



3. I use notepad and morse code image in google to see what the morse wanna say



4. I try to open the link on my friend laptop and it leads to the sam video, because at first i thought we have to see the playlist description or something since the playlist is mix that made for us from youtube, so its not, its about the legends of lord video, I have try using the video tittle but i ask maam farah for the clue, seems it doesnt feel right if i use the tittle, and she said using the lyrics, so i use the lyrics to match the morse code



5. I use the tittle video to solve the key, but its weird somehow



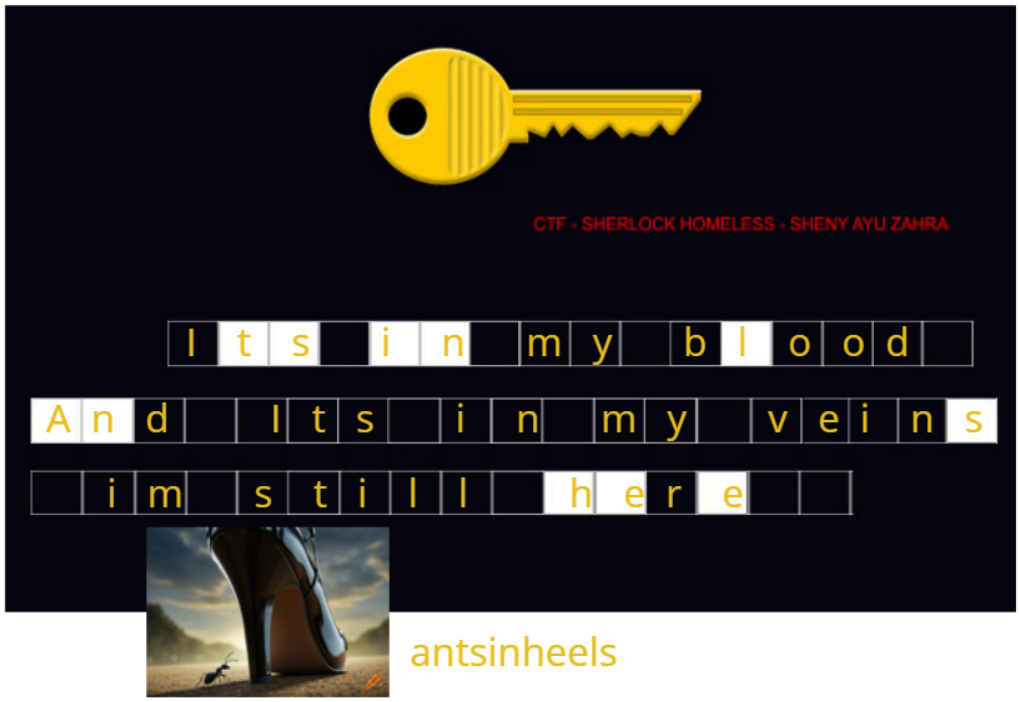
attilgedse

6. Then i use the lyrics but its still weird, because the last white block is im, and no space needed, so its not the right key



antsinerils

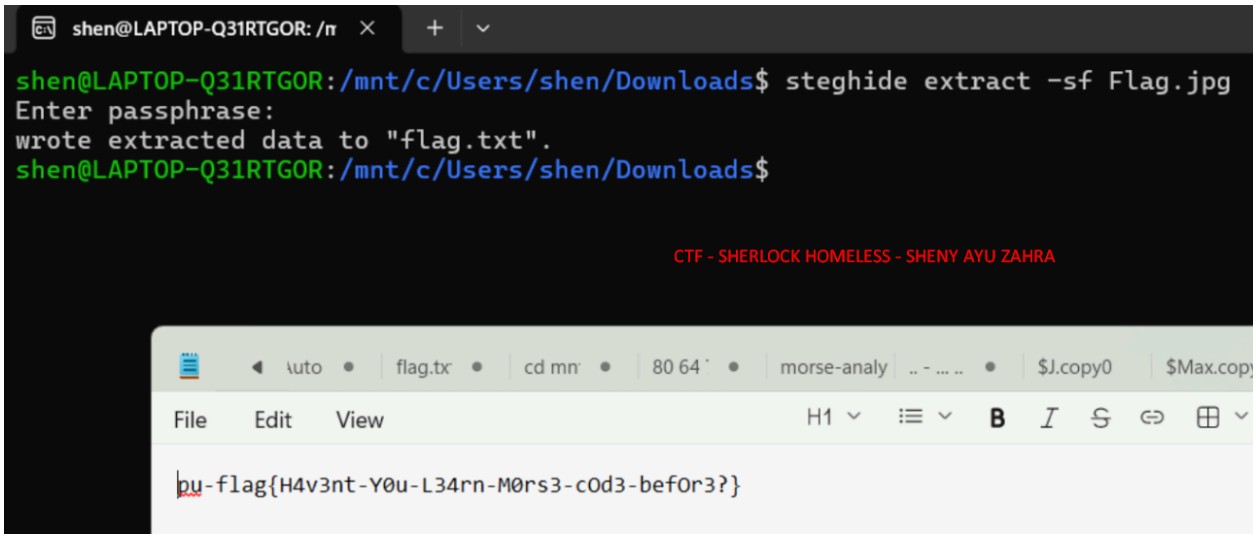
7. And i added space after veins word, and its perfect, its about ants in the heels. OMG



antsinheels

ItsinmybloodAnsItsinmyveinsimstillhere25

8. After i find the key, i use steghide to extract something on the image, since the image have key icon, after that they extracted a file flag.txt, and i open the file and there is it, “the flaggg”



Challenge 10 : EzTraverse Part 1-2

Solved On: April 13th, 1:00:07 PM
Solved by: Muhammad Gale Adi Saputra
Flag Retrieved: pu-flag{p3rs1st3nc3_v14_cr0nd}

Challenges overview:
Linux disk-image forensics challenge. A web server was compromised; the sysadmin grabbed a full disk image before taking the box offline. Part 1 requires the full folder path to the file the attacker used to maintain access. Part 2 requires retrieving the flag

Key Findings:

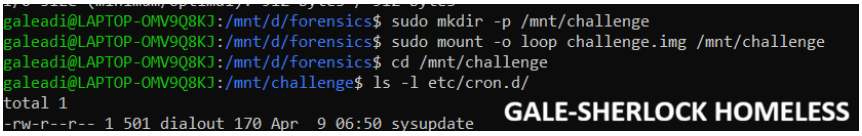
- The attacker created a fake “system update” cron job that runs every minute as root.
- Persistence file: /etc/cron.d/sysupdate
- The flag was hidden as a comment inside the same cron file.

Forensic Analysis:

- Identified that challenge.img is a raw ext4 filesystem (not a partitioned disk).
- Mounted the image directly using loop device.
- Located the malicious cron job in /etc/cron.d/sysupdate.

Tools Used:
WSL CLI, file, fdisk, mount, grep, cat

Solving Step-by-step:
Part 1
Our web server was acting strange last night. The sysadmin managed to grab a disk image before taking the box offline. Find out how the attacker maintained access.
Give the folder path to the file e.g: /usr/local/bin/fileaccess
First I must make an folder for mount, and because the file image is raw system, so no need offset, then go into file system, check is there any cron folder, because the attacker hide the backdoor in cron folder



Part 2

Easy enough is it to get the flag?

Don't forget, the format is pu-flag{flag}

Enter flag below

After that open the cron file “cat etc/cron.d/sysupdate” In this file we will get a fake cron job, and the hidden flag wichis “pu-flag{p3rs1st3nc3_v14_cr0nd}”

