MMU Workshop CTF

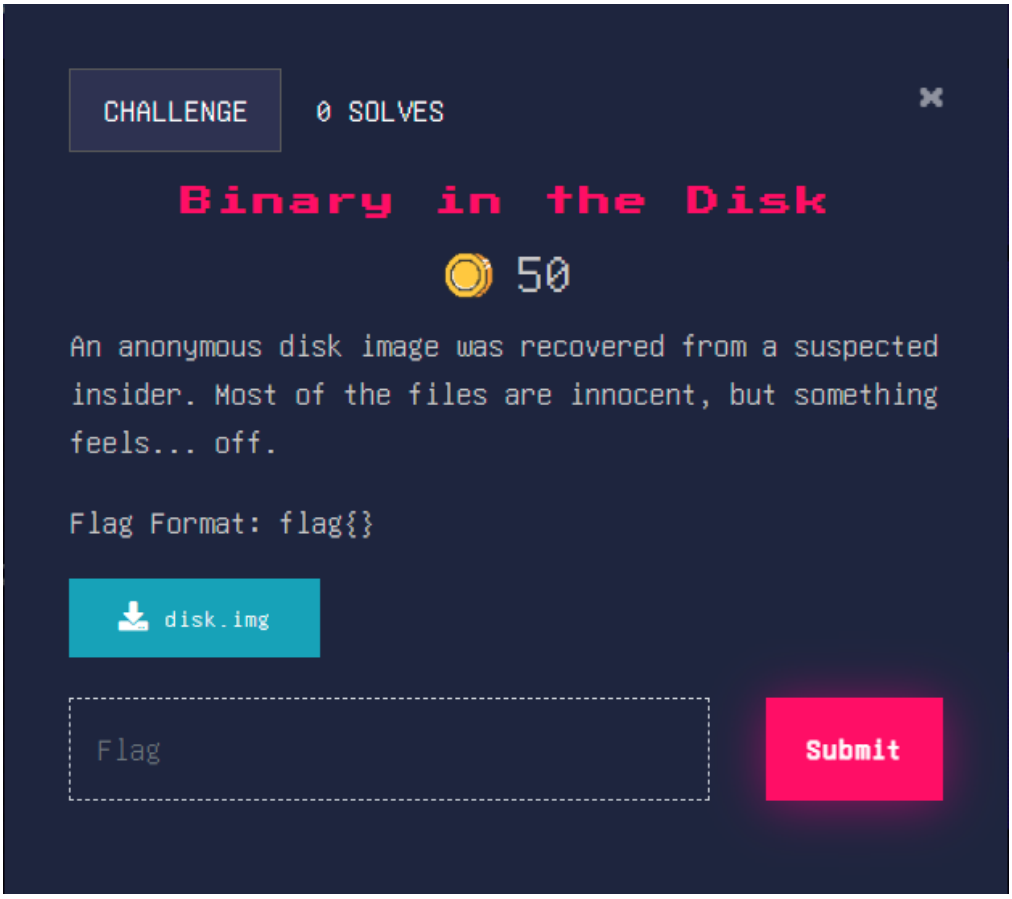
Prepared By:

conan a.k.a Nawfal

TABLE OF CONTENTS

[Forens] Binary in the Disk

Here is the question:



📚 Description

An anonymous disk image was recovered from a suspected insider. Most of the files are innocent, but something feels... off.

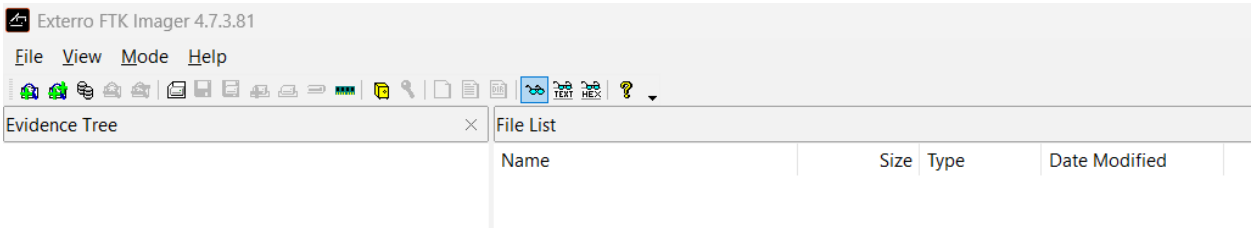
Flag Format: flag{}

✨ Walkthrough

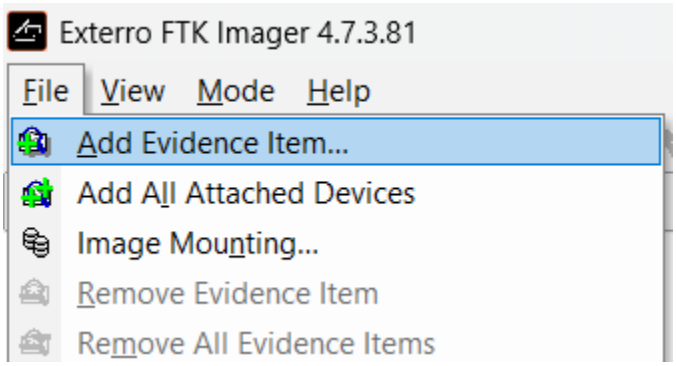
The challenge has given us a disk image file (.dd) to analyze.

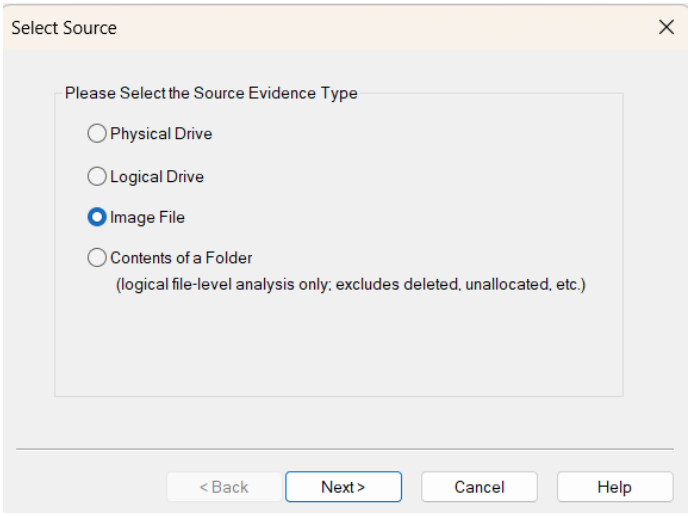
```
┌──(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/Downloads/chal/Disk Forensics]
└─$ file disk.img
disk.img: Linux rev 1.0 ext4 filesystem data, UUID=536d5374-45a0-4d8d-9b9b-517d6ea6c3bc (extents) (64bit) (large files)
(huge files)
```

To analyze the disk image, we will utilize a forensics tool called FTK Imager.



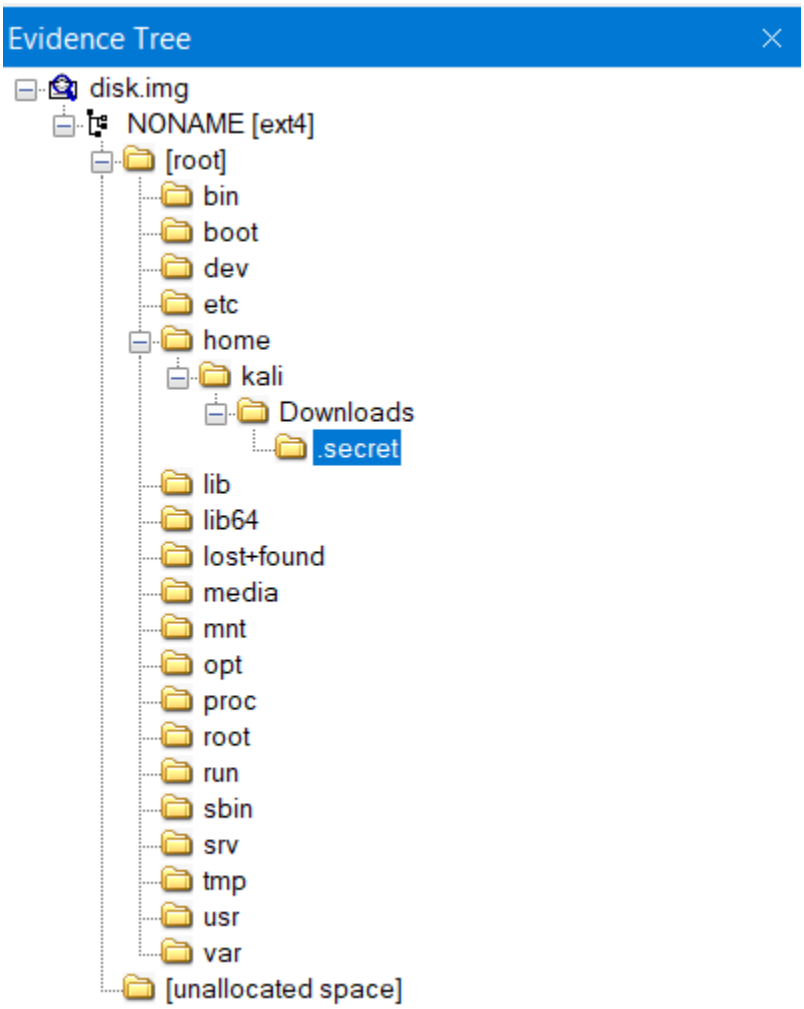
Let’s import the disk image as evidence inside the FTK Imager.





After importing the disk image, it will appear in the **Evidence Tree** section.

We can now view the structure inside the disk image to go through all the folders and directories. Usually, the creator of the challenge will hide something inside the home directory. LOOK! There is a secret folder holding a binary file inside 🧐.



File List

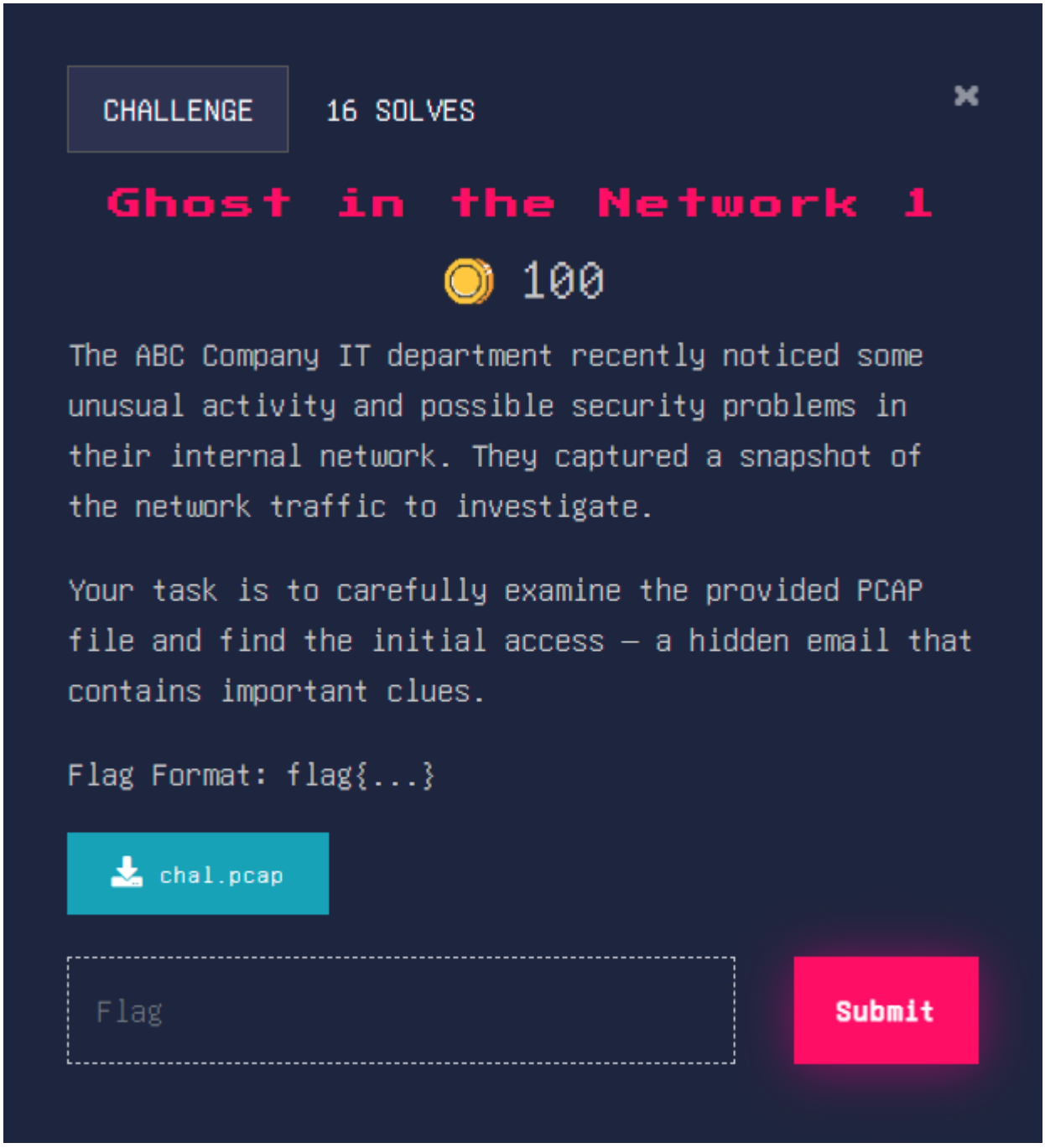
Name	Size	Type	Date Modified
update.bin	46 (1 KB)	Regular File	22/8/2025 3:31:37 ...

```
#!/bin/bash
echo "flag{w45_1t_H4rd_to_L00k?}"
```

Flag- flag{w45_1t_H4rd_to_L00k?}

[Forensics] Ghost in the Network 1

Here is the question:



🎭 Description

The ABC Company IT department recently noticed some unusual activity and possible security problems in their internal network. They captured a snapshot of the network traffic to investigate.

Your task is to carefully examine the provided PCAP file and find the initial access — a hidden email that contains important clues.

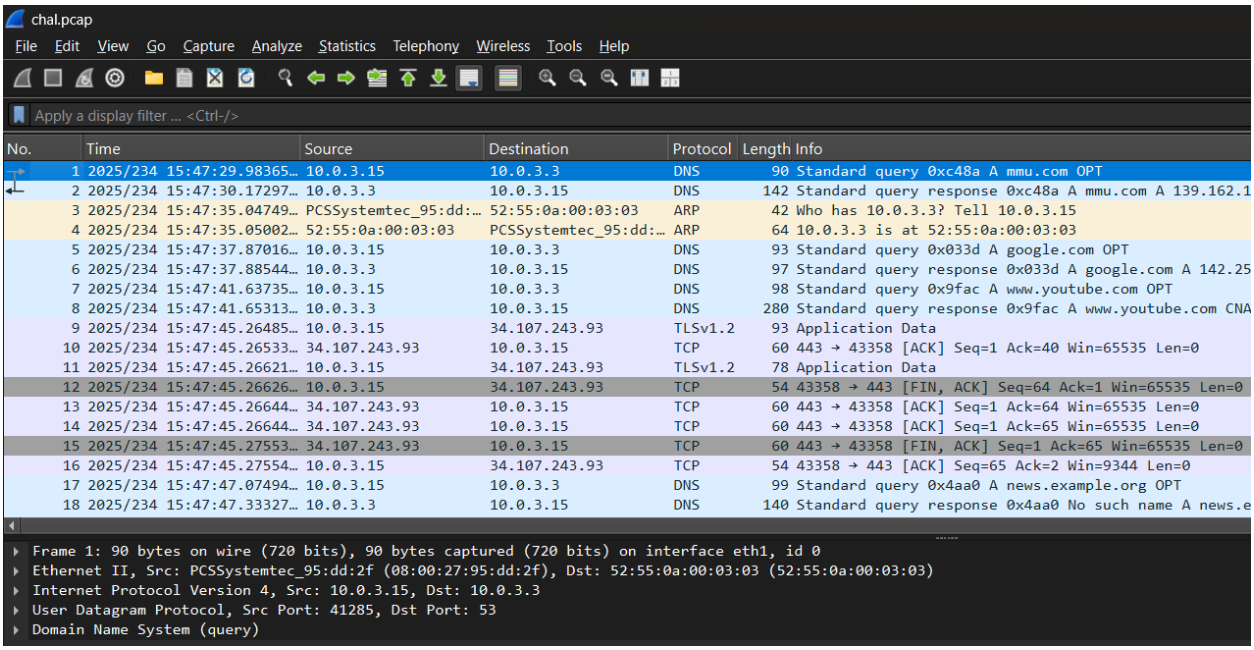
Flag Format: flag{...}

✨ Walkthrough

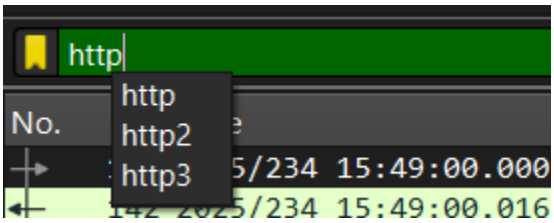
Going through the file given, we'll see a pcap file called "**chal.pcap**"

```
┌──(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/Downloads/chal/Network Forensics]
└─$ ls
chal.pcap

┌──(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/Downloads/chal/Network Forensics]
└─$ file chal.pcap
chal.pcap: pcapng capture file - version 1.0
```

By utilizing **Wireshark**, we can see the network packets captured for a period of time inside the pcap file

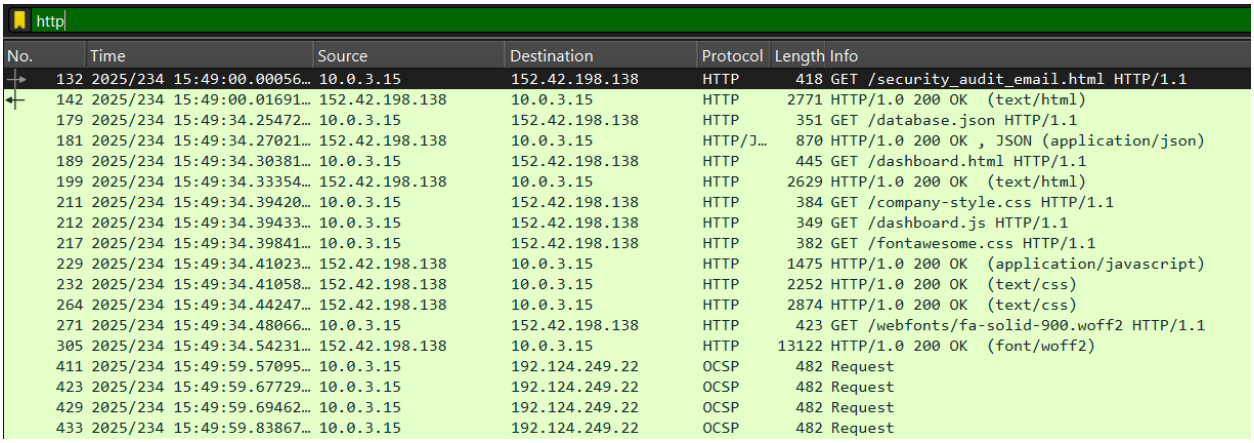


Not going go through all of it 😅, let's just focus on one protocol that I found interesting which is **http** Protocol.



Owhh, look at these. It seems like during the network capturing, they caught a website.

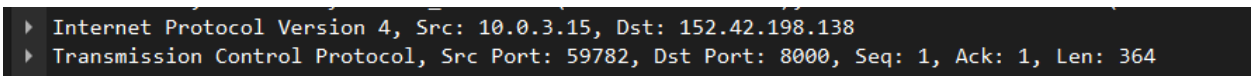
And...we found a ***suspicious email*** among the network packets



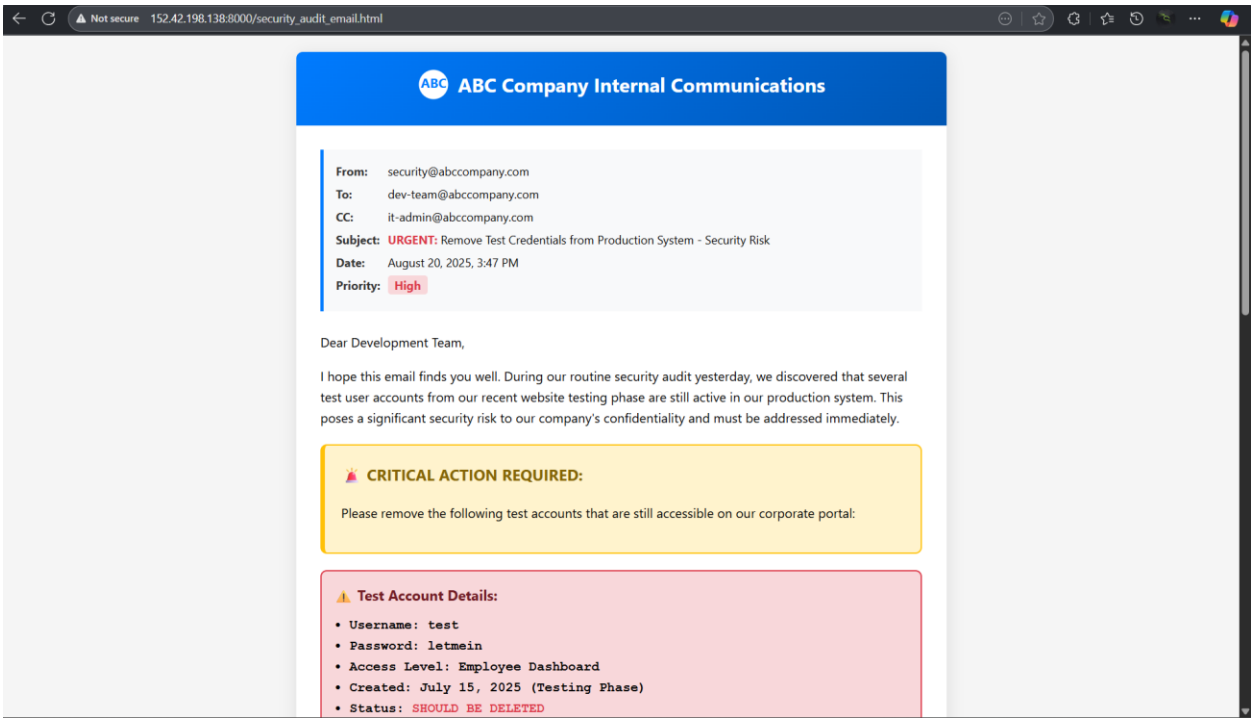
To access the website, we need to know the ip of the website and the port used to host it. From clicking the packet reveal **what we need!**

: 1. The destination IP (Website's IP)

2. The destination Port (Website's Port)



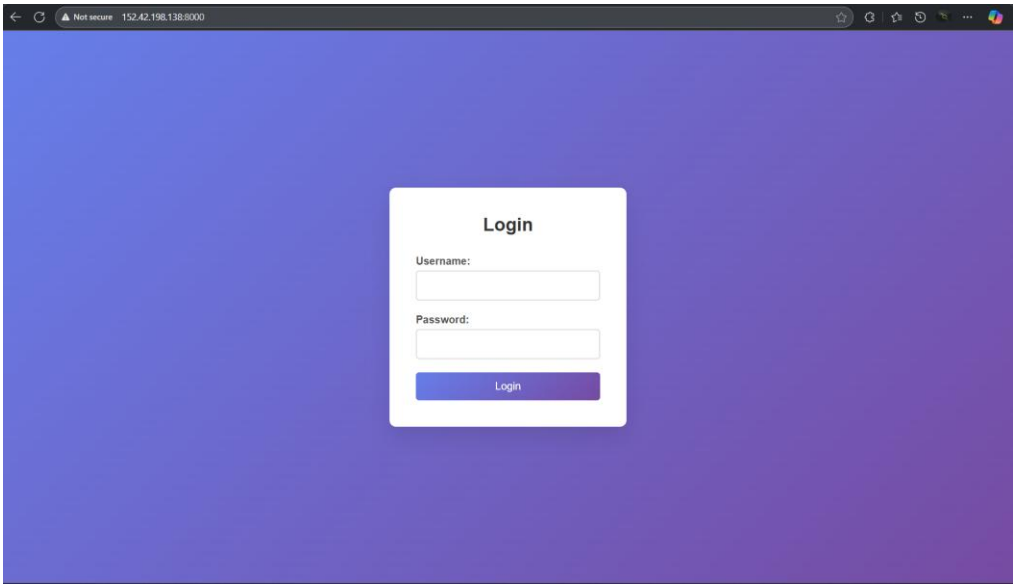
Opening the email file at the website reveals us with a **remainder email** from the **security team** to the **developing team** of the ABC Company to delete a testing user which could compromise the website's security.

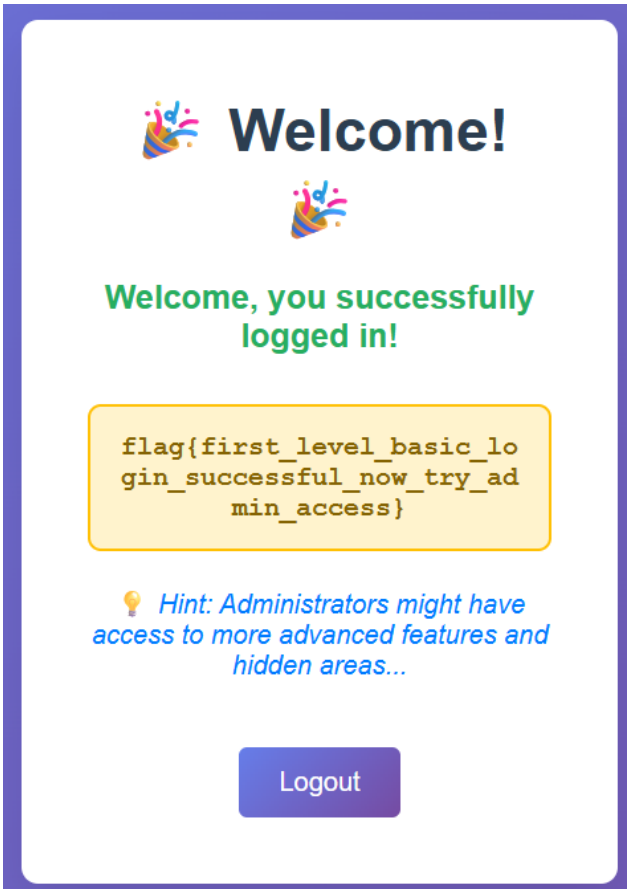


Now we got what we need we can try and **login** into the website using this credential.

> 🚨 **CRITICAL ACTION REQUIRED:**
>
> Please remove the following test accounts that are still accessible on our corporate portal:

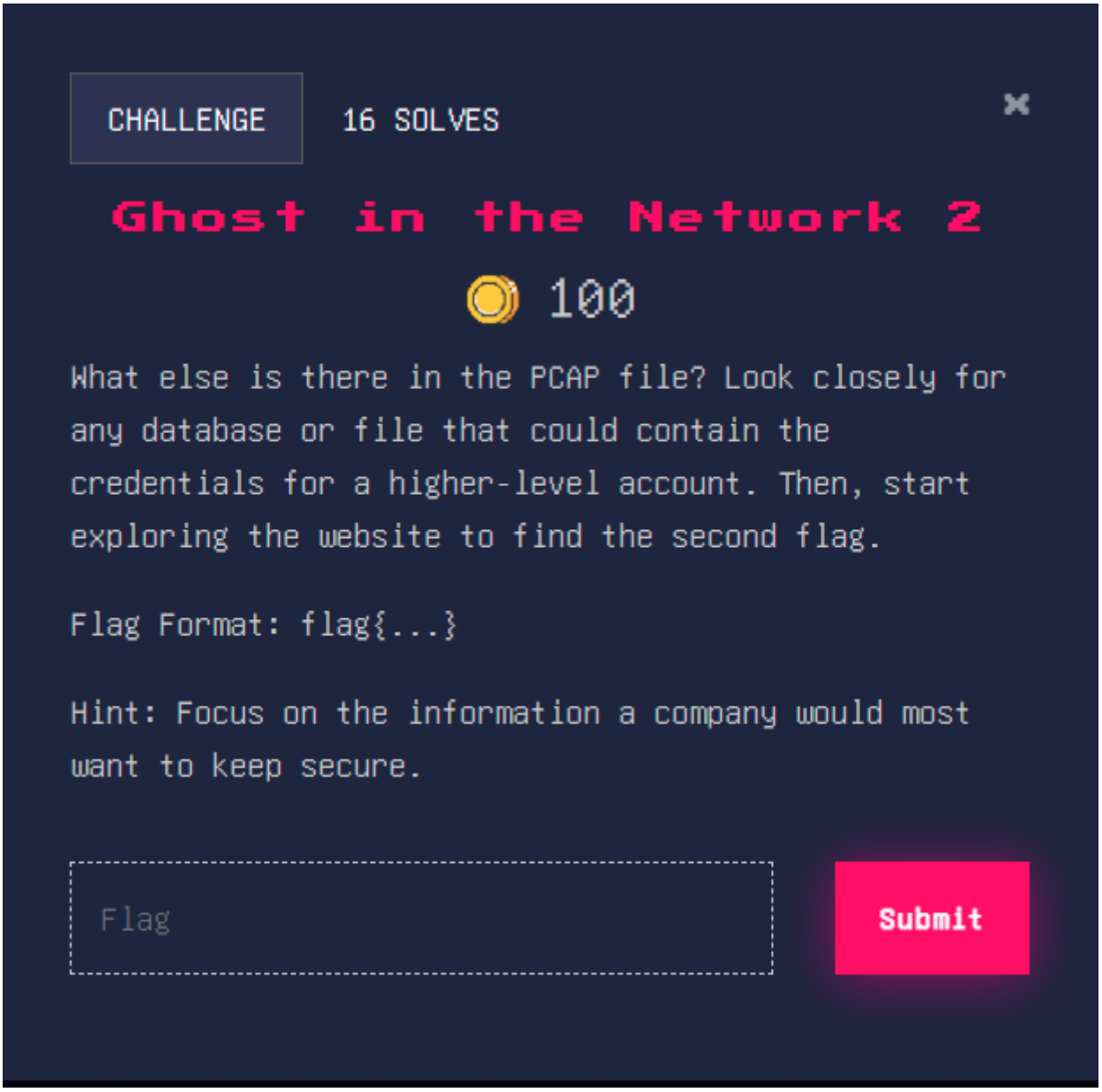
> ⚠️ **Test Account Details:**
>
> - `Username: test`
> - `Password: letmein`
> - `Access Level: Employee Dashboard`
> - `Created: July 15, 2025 (Testing Phase)`
> - `Status: SHOULD BE DELETED`





Flag - flag{first_level_basic_login_successful_now_try_admin_access}

[Forensics] Ghost in the Network 2



Description

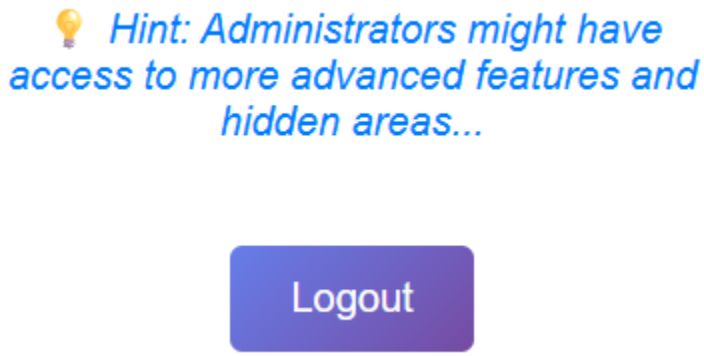
What else is there in the PCAP file? Look closely for any database or file that could contain the credentials for a higher-level account. Then, start exploring the website to find the second flag.

Flag Format: flag{...}

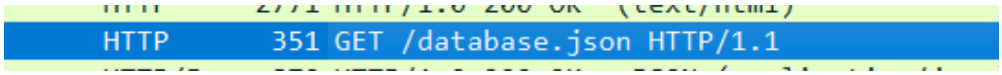
Hint: Focus on the information a company would most want to keep secure.

✨ Walkthrough

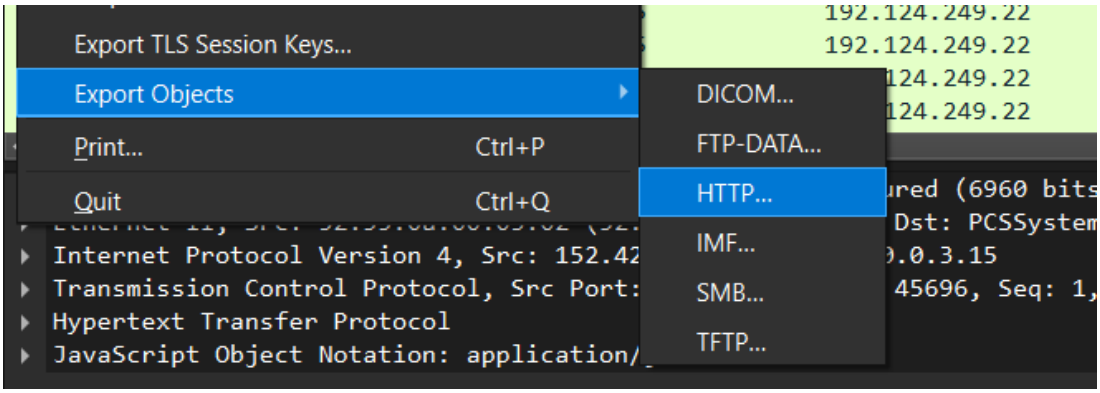
After retrieving the first flag, we get a hint from the challenge stated "***Administrators might have access to more advanced features and hidden areas***". This means that we must gain a higher level of access (admin) to get the second flag.

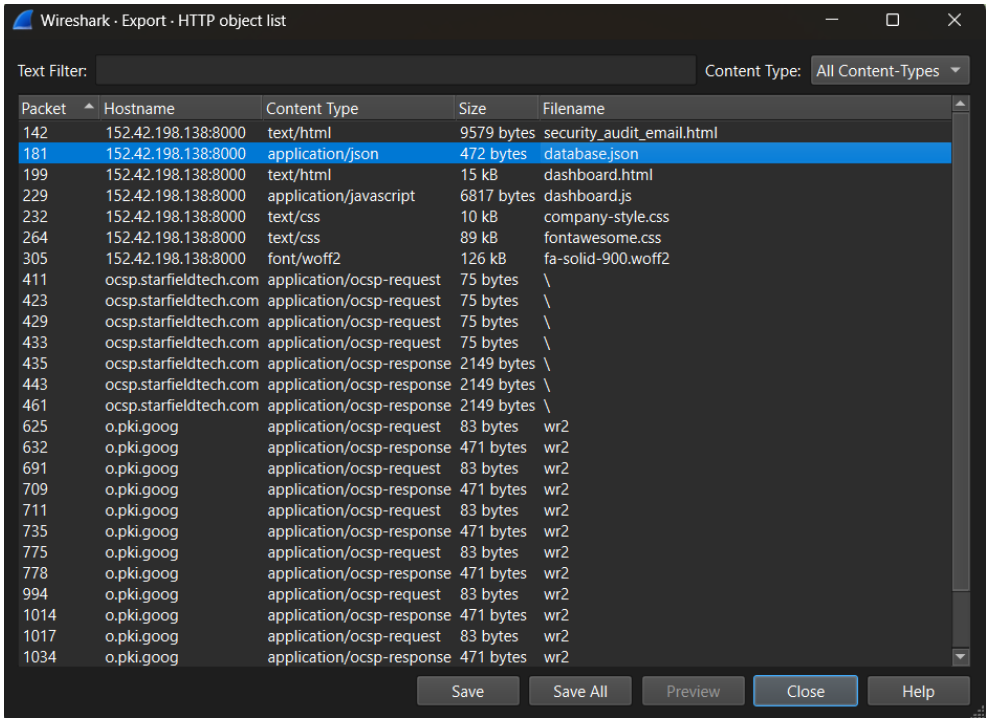


Looking back again at the **pcap file** reveals us with a **database.json file**.



We can now use a built-in tool inside **Wireshark** to download the file from the pcap.





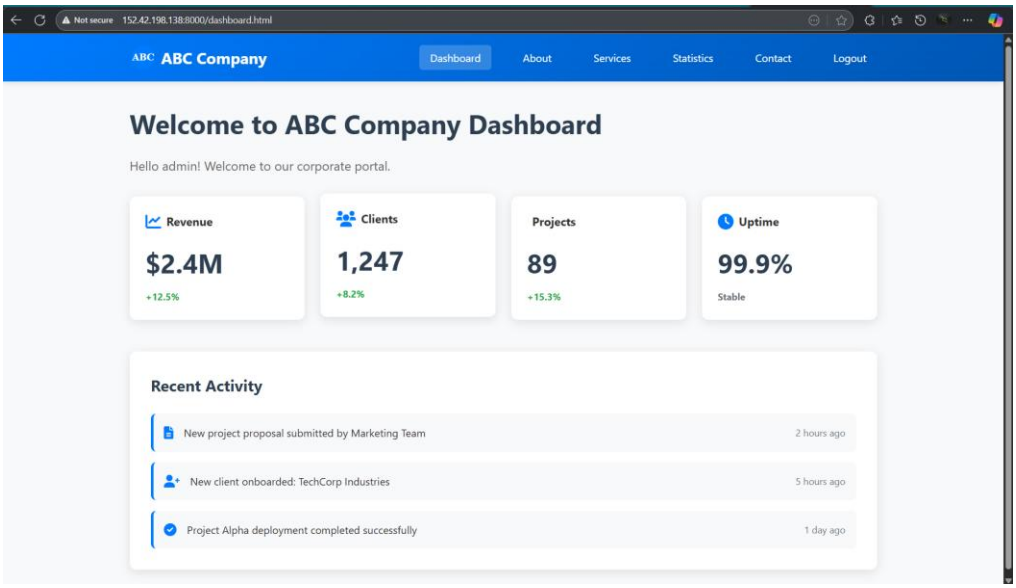
WOAHHH! It is a database file containing the credentials to access admin account.

database.json

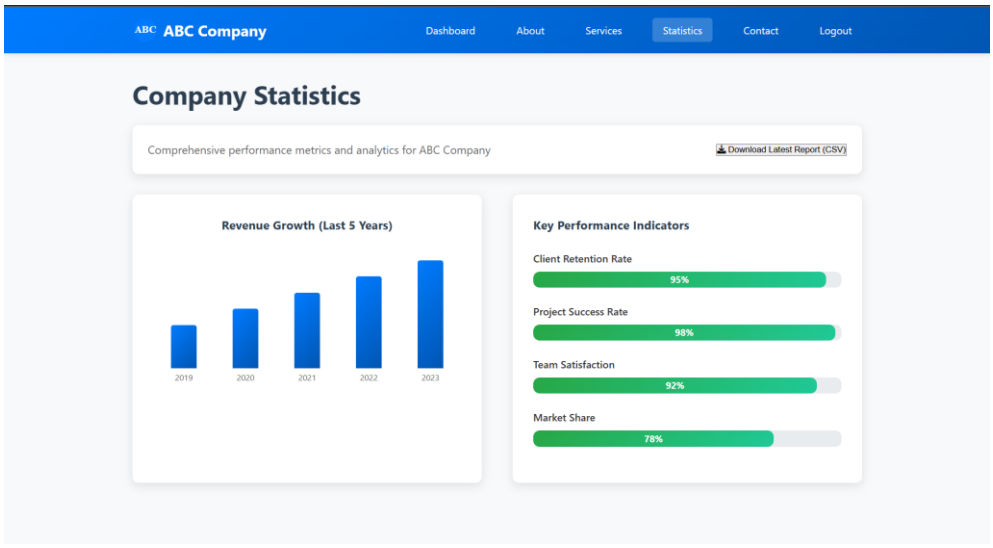
C: > Users > jacob > Downloads > chal > Network Forensics

```json
{
  "users": [
    {
      "id": 1,
      "username": "test",
      "password": "letmein",
      "role": "employee",
      "access_level": "basic"
    },
    {
      "id": 2,
      "username": "admin",
      "password": "password123",
      "role": "administrator",
      "access_level": "dashboard"
    },
    {
      "id": 3,
      "username": "user",
      "password": "secret",
      "role": "guest",
      "access_level": "basic"
    }
  ]
}
```

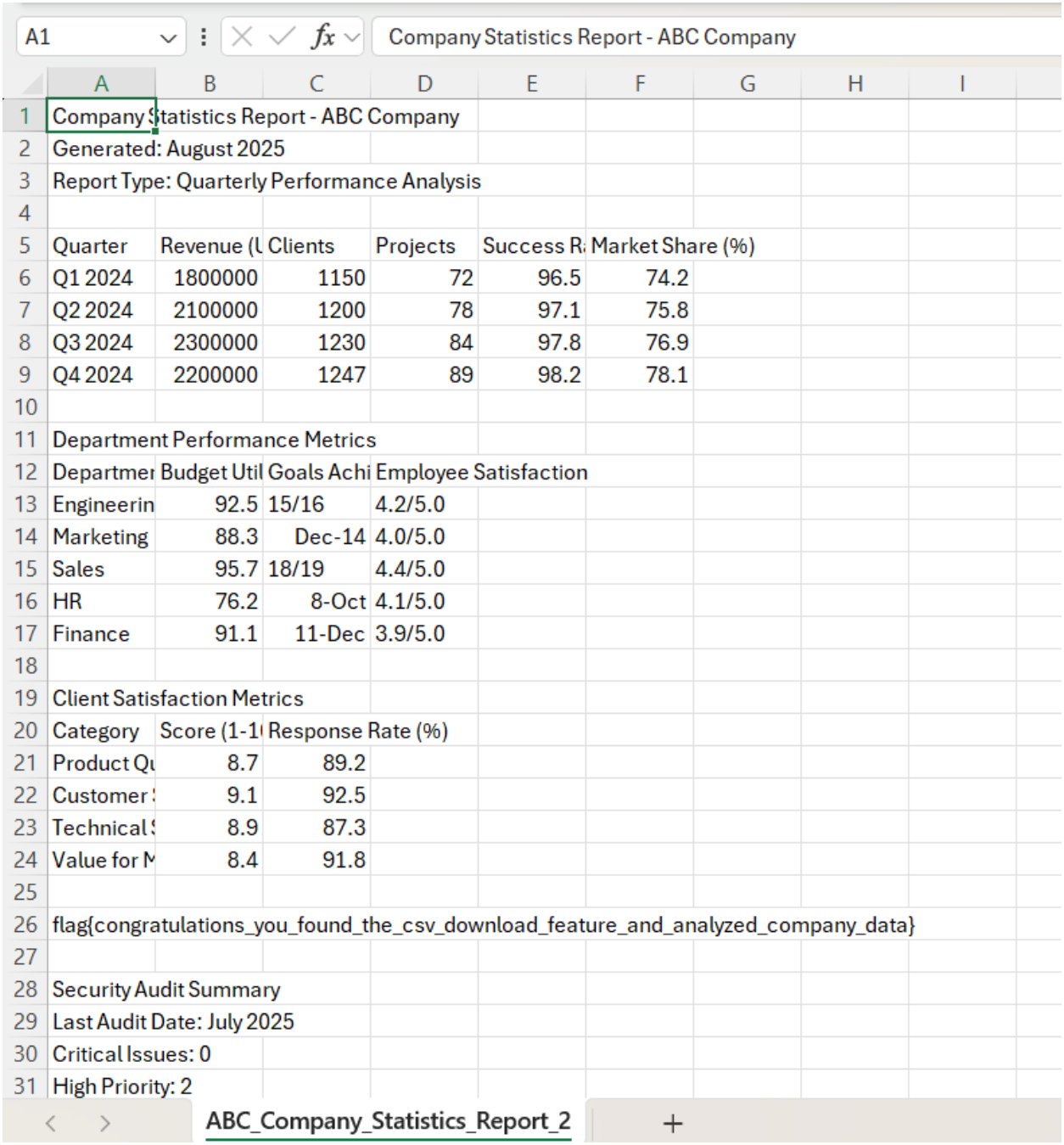
Log into the website using admin **credentials** we can now access the ABC Company Dashboard.



Let's go through all the pages inside the website... Based on the challenge again I suspect that the hint given about the thing company want to protect is the **company's data**. We can download the statistics file to view the content.



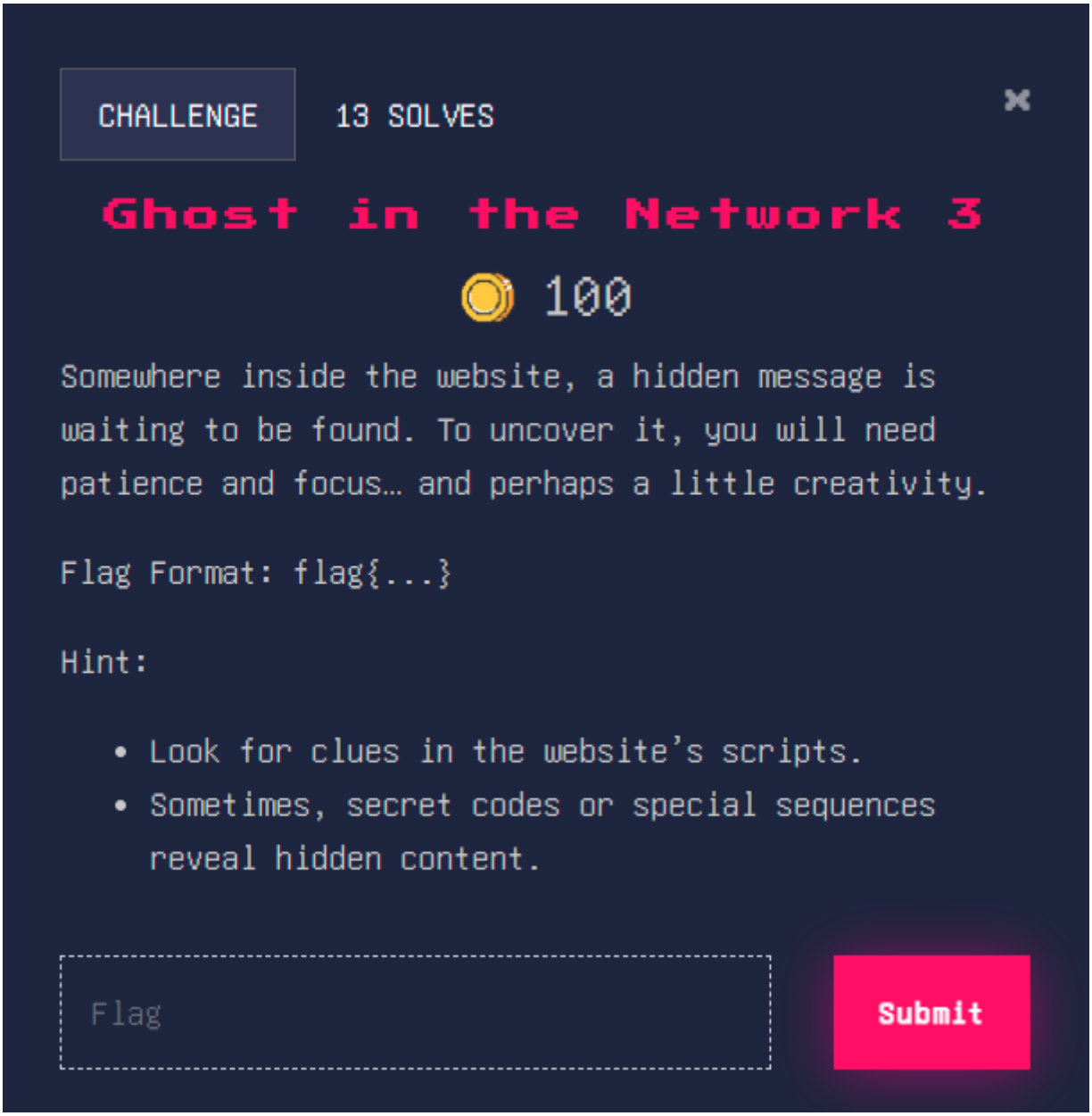
Look at the **flag!**



	A	B	C	D	E	F	G	H	I
1	Company Statistics Report - ABC Company								
2	Generated: August 2025								
3	Report Type: Quarterly Performance Analysis								
4									
5	Quarter	Revenue (U	Clients	Projects	Success R	Market Share (%)			
6	Q1 2024	1800000	1150	72	96.5	74.2			
7	Q2 2024	2100000	1200	78	97.1	75.8			
8	Q3 2024	2300000	1230	84	97.8	76.9			
9	Q4 2024	2200000	1247	89	98.2	78.1			
10									
11	Department Performance Metrics								
12	Departmer	Budget Util	Goals Achi	Employee Satisfaction					
13	Engineerin	92.5	15/16	4.2/5.0					
14	Marketing	88.3	Dec-14	4.0/5.0					
15	Sales	95.7	18/19	4.4/5.0					
16	HR	76.2	8-Oct	4.1/5.0					
17	Finance	91.1	11-Dec	3.9/5.0					
18									
19	Client Satisfaction Metrics								
20	Category	Score (1-1	Response Rate (%)						
21	Product Qu	8.7	89.2						
22	Customer S	9.1	92.5						
23	Technical S	8.9	87.3						
24	Value for M	8.4	91.8						
25									
26	flag{congratulations_you_found_the_csv_download_feature_and_analyzed_company_data}								
27									
28	Security Audit Summary								
29	Last Audit Date: July 2025								
30	Critical Issues: 0								
31	High Priority: 2								

Flag - flag{congratulations_you_found_the_csv_download_feature_and_analyzed_company_data}

[Forensics] Ghost in the Network 3



📚 Description

Somewhere inside the website, a hidden message is waiting to be found. To uncover it, you will need patience and focus... and perhaps a little creativity.

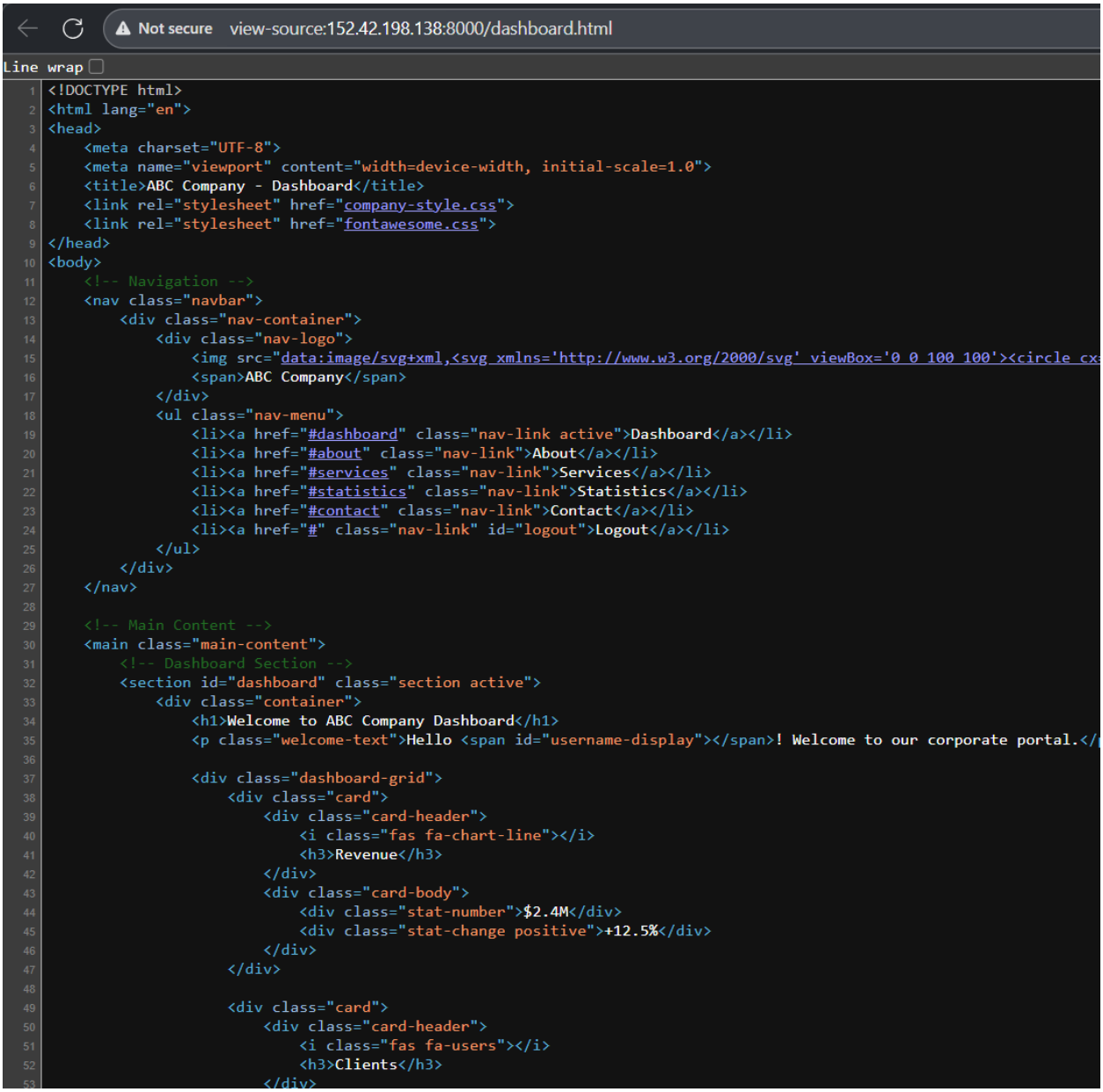
Flag Format: flag{...}

Hint:

- *Look for clues in the website's scripts.*
- *Sometimes, secret codes or special sequences reveal hidden content.*

✨ Walkthrough

To find the Last flag, let's go back to website and now we will view the source code of the website.



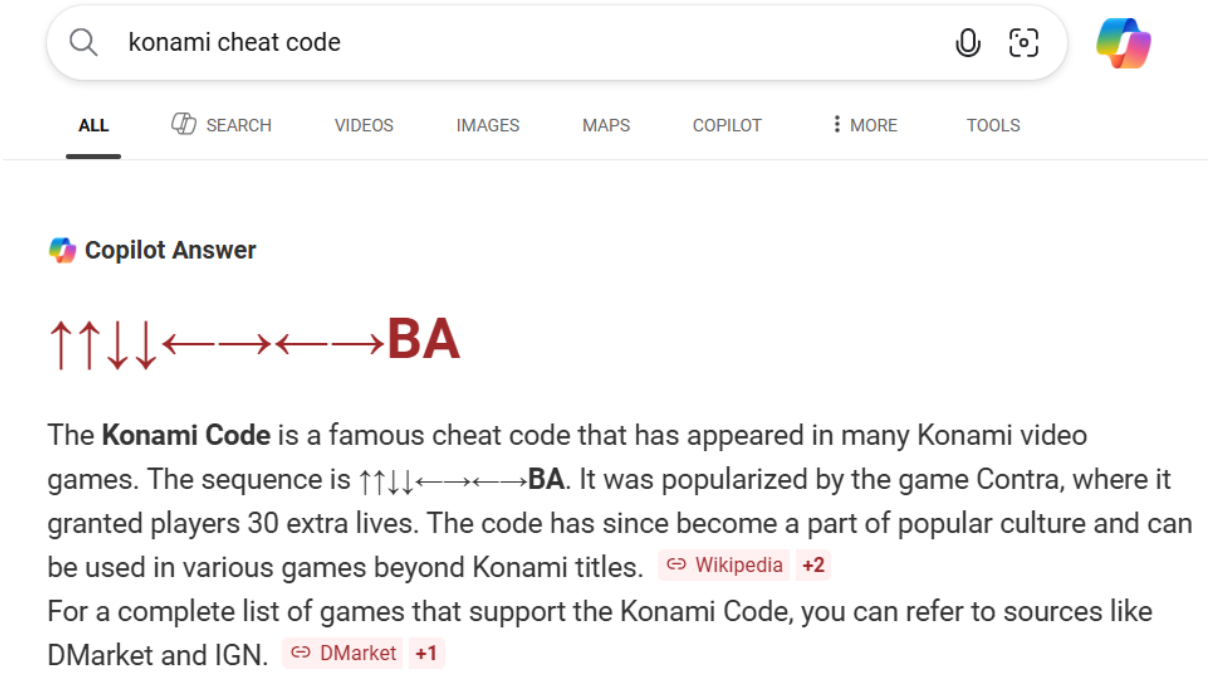
Tips from me: Always check the **javascript** files, there might be an interesting things there!

```
                        </form>
                    </div>
                </div>
            </div>
        </section>
    </main>

    <script src="dashboard.js"></script>
</body>
</html>
```

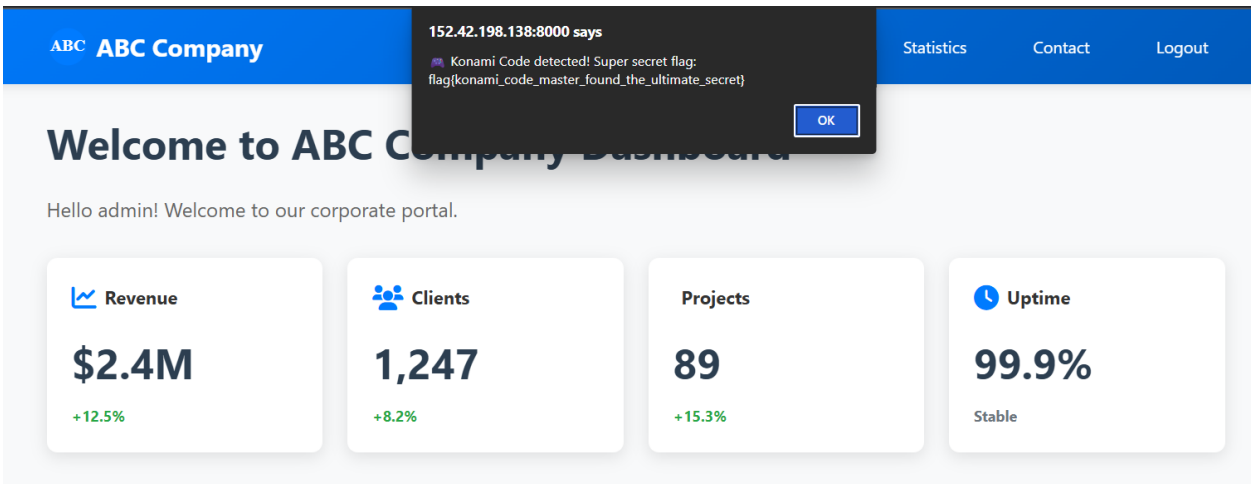
Volla! The flag is there! But I'm curious about something "The Konami cheat code?"

```
// Secret konami code easter egg (optional extra challenge)
let konamiCode = [];
const konami = [38, 38, 40, 40, 37, 39, 37, 39, 66, 65]; // â†‘â†‘â†“â†“â†â†’â†â†’BA

document.addEventListener('keydown', function(e) {
    konamiCode.push(e.keyCode);
    if (konamiCode.length > konami.length) {
        konamiCode.shift();
    }

    if (konamiCode.length === konami.length &&
        konamiCode.every((code, index) => code === konami[index])) {

        // Super secret flag for advanced users
        alert('ðŸŽ® Konami Code detected! Super secret flag: flag{konami_code_master_found_the_ultimate_secret}');
        konamiCode = [];
    }
});
```

I did some **googling** and found out that the combination of the Konami cheat code is:

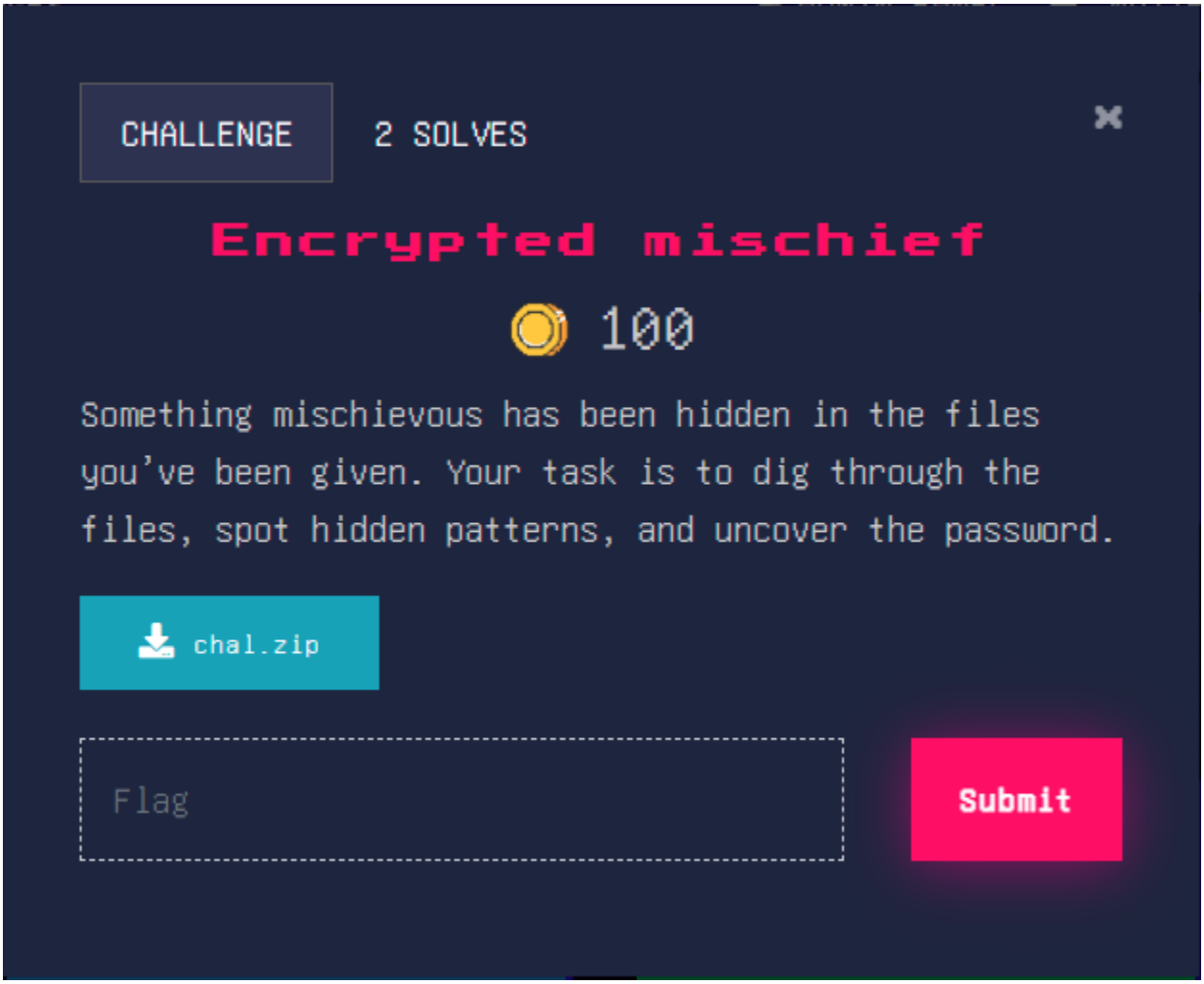


By clicking the buttons on the keyboard with the Konami cheat code, we triggered the last flag itself!



Flag - flag{konami_code_master_found_the_ultimate_secret}

[Forensics] Encrypted mischief



📚 Description

Something mischievous has been hidden in the files you've been given. Your task is to dig through the files, spot hidden patterns, and uncover the password.

✨ Walkthrough

Given a zip file. Let's unzip it first to view it's content.

```
┌──(kali㉿NawfalMatebook)-[/mnt/c/Users/
└─$ ls
chal.zip
```

```
┌──(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/
└─$ unzip chal.zip
```

We found 2 files inside, **“1_1.png”** and **“secrets.zip”** file. Let’s try to unzip the secrets.zip file first. **OH NO!!** it need a password to unzip it.

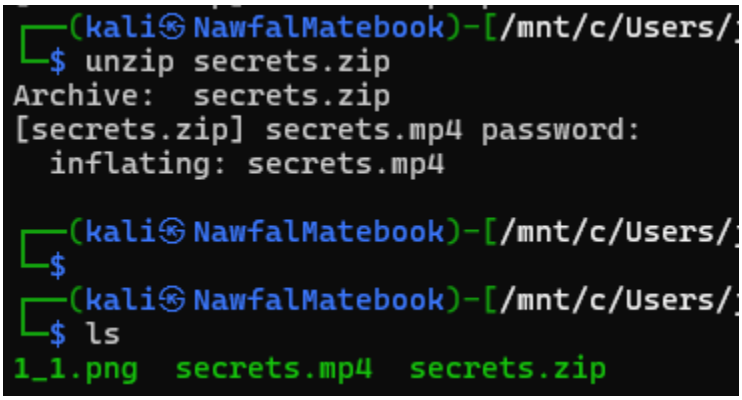
```
┌──(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/Downloads/chal/File Ana
└─$ file 1_1.png
1_1.png: PNG image data, 900 x 506, 8-bit/color RGB, non-interlaced
```

```
┌──(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/
└─$ ls
1_1.png  secrets.zip

┌──(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/
└─$ unzip secrets.zip
Archive:  secrets.zip
[secrets.zip] secrets.mp4 password:
```

I'm quite curious with why the challenge give us a png file, so I analyze the metadata using **exiftool**, then I found something interesting **"THE COMMMENT"**. MMU123 seems like a password to me 🤭.

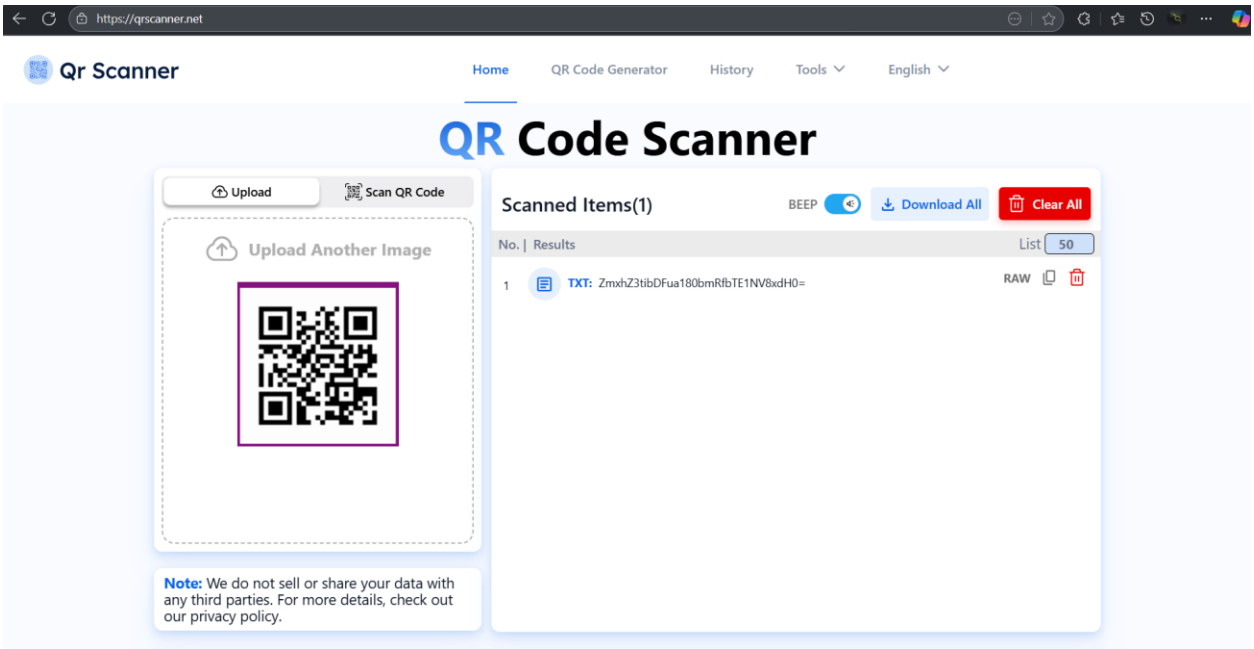
```
┌──(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/Downloads/chal/File Analysis/Challenge1
└─$ exiftool 1_1.png
ExifTool Version Number         : 13.25
File Name                       : 1_1.png
Directory                       : .
File Size                       : 467 kB
File Modification Date/Time     : 2025:04:16 17:52:08+08:00
File Access Date/Time           : 2025:08:24 17:54:09+08:00
File Inode Change Date/Time     : 2025:08:24 17:54:09+08:00
File Permissions                : -rwxrwxrwx
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 900
Image Height                    : 506
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Orientation                     : Horizontal (normal)
X Resolution                    : 71.983
Y Resolution                    : 71.983
Resolution Unit                 : inches
Y Cb Cr Positioning             : Centered
Exif Version                    : 0210
Components Configuration        : Y, Cb, Cr, -
Flashpix Version                : 0100
Color Space                     : Uncalibrated
Exif Image Width                : 900
Exif Image Height               : 506
Pixels Per Unit X               : 2834
Pixels Per Unit Y               : 2834
Pixel Units                     : meters
Exif Byte Order                 : Little-endian (Intel, II)
Warning                         : IFD0 pointer references previous IFD0 directory
Comment                         : MMU123
Image Size                      : 900x506
Megapixels                      : 0.455
```

Letssss goo!! We can unzip the zip file. The content seems to be a mp4 file, Lets try opening it.



A QR code?!

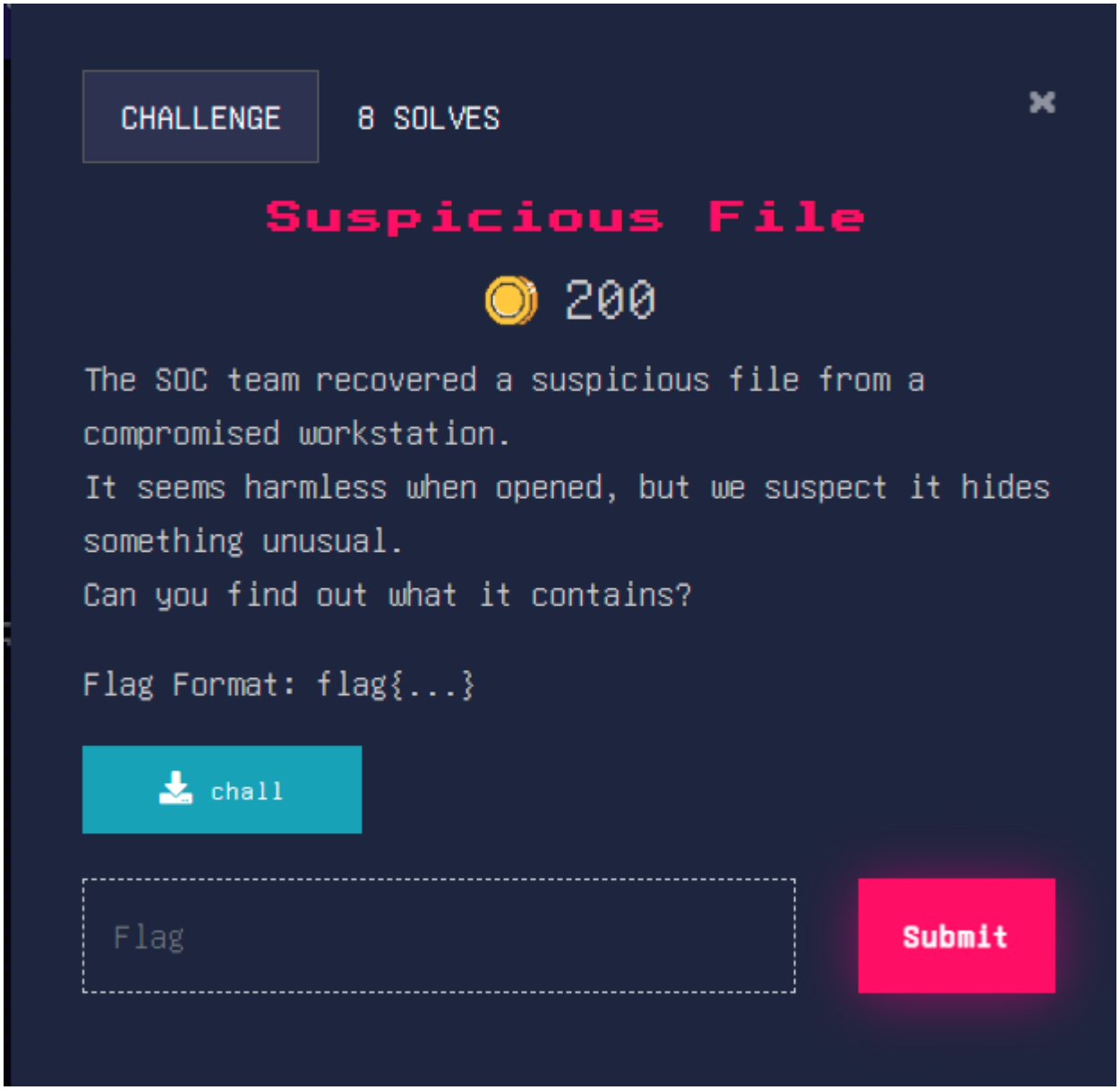
Using online QR Code Scanner, we decode it into a string, **“Base64”?**



Let’s decode it in kali using the “**base64 -d”** function, and I was correct here is the flag!

⚐ **Flag - flag{bl1nk_4nd_m155_1t}**

[Forensics] Suspicious File



📚 Description

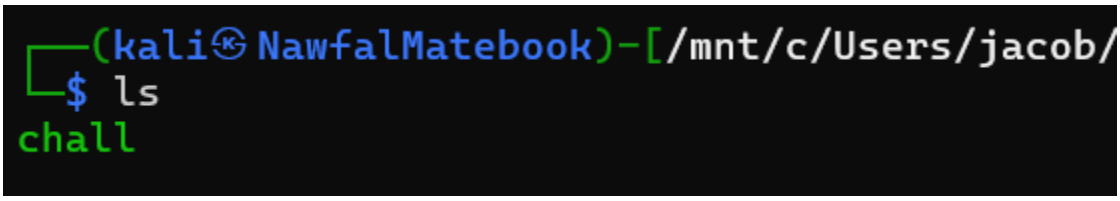
The SOC team recovered a suspicious file from a compromised workstation.
It seems harmless when opened, but we suspect it hides something unusual.
Can you find out what it contains?

Flag Format: flag{...}

✨ Walkthrough

Solution

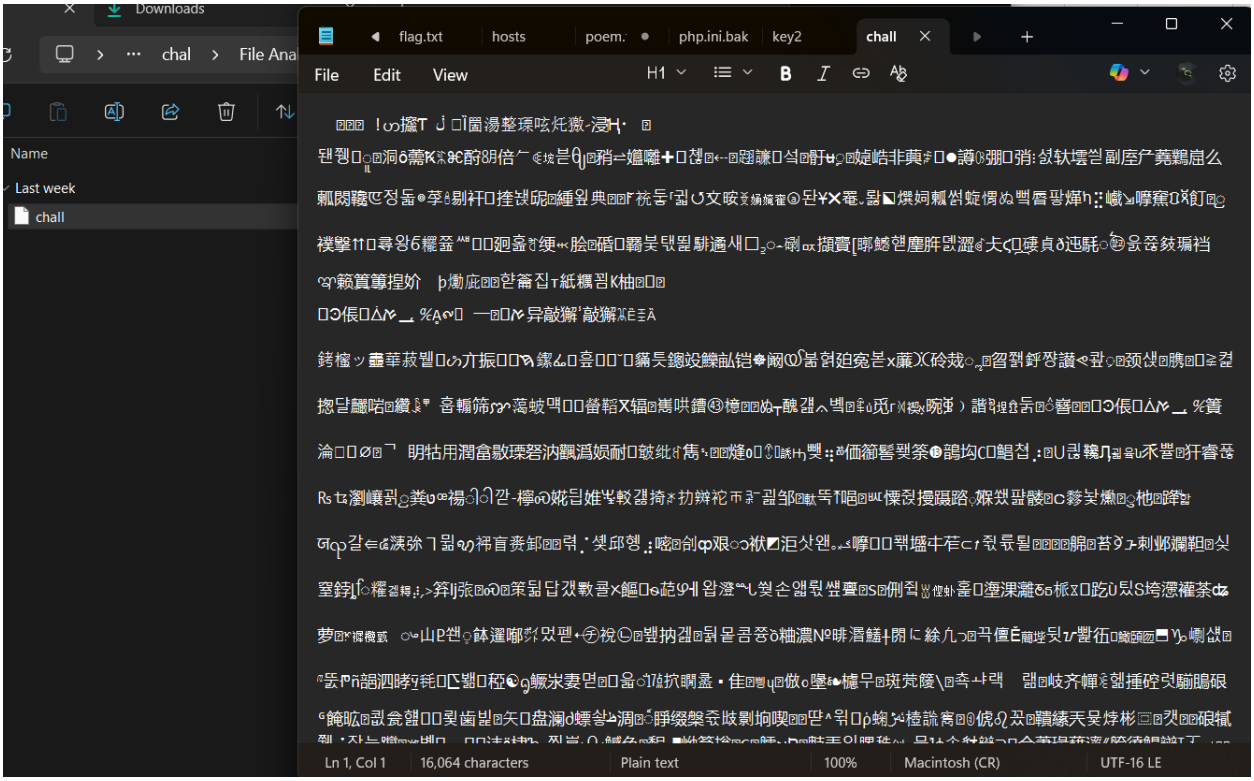
Given a file with no extension, **What?!! No file format?** 😱



Let’s utilize the file command to view the type of file it is, we now know it is a zip file (probably)

```
┌──(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/Downloads/chal/File Analysis/Challenge2]
└─$ file chall
chall: Zip archive data, made by v4.5, extract using at least v2.0, last modified Jan 01 1980 00:00:00, uncompressed siz
e 1718, method=deflate
```

Ok now I’m trying to read it content using notepad.... LOST HOPE 😓



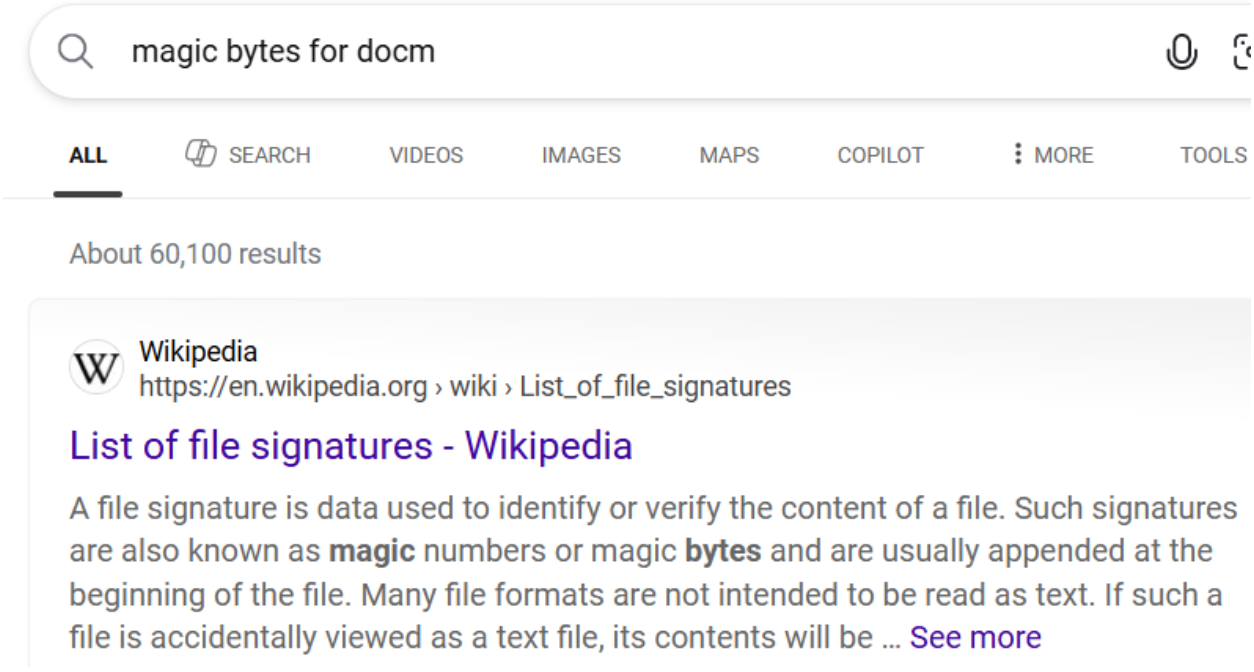
Ok now we know it is a zip file, there is a command to view the embedded file inside it, **binwalk command**, odd… There is a lot of word and xml files inside, now I suspect it is actually a **word file**.

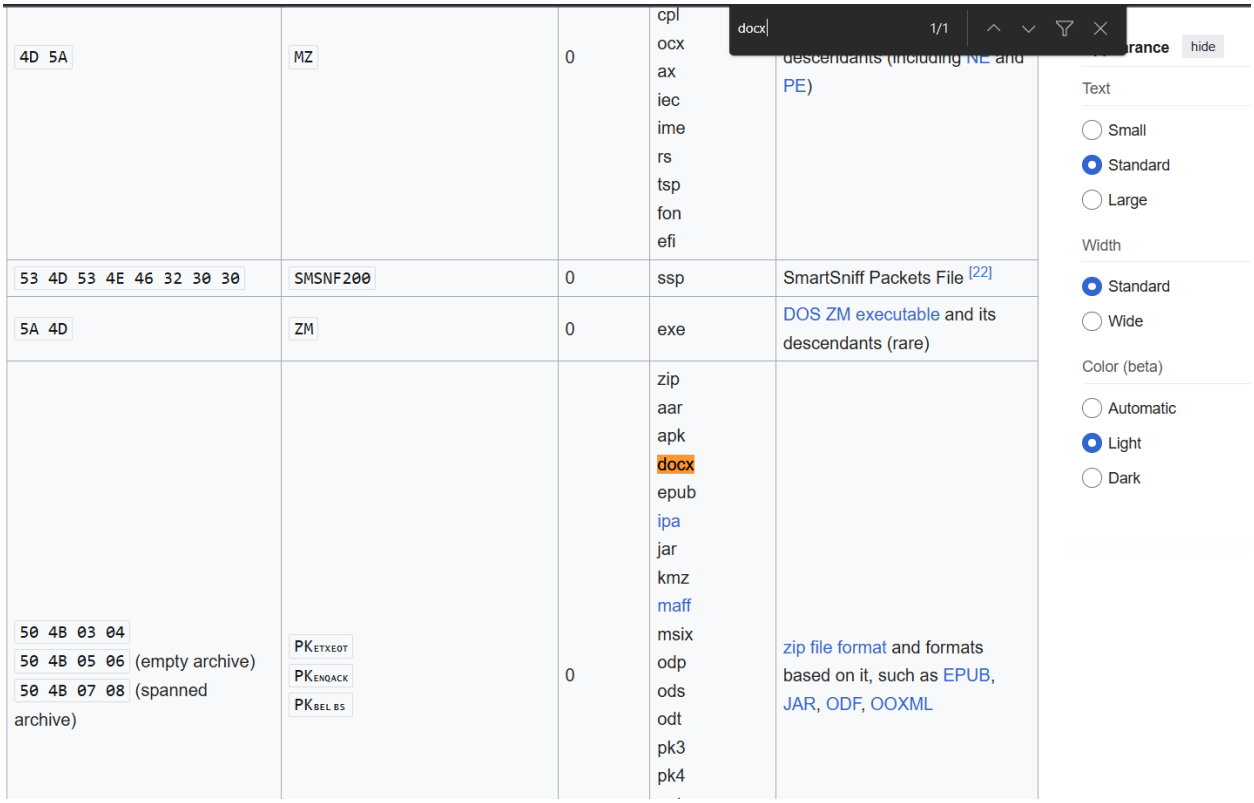
```
┌──(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/Downloads/chal/File Analysis/Challenge2]
└─$ binwalk chall

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
997           0x3E5           Zip archive data, at least v2.0 to extract, compressed si
ize: 590, name: _rels/.rels
1797          0x705           Zip archive data, at least v2.0 to extract, compressed si
size: 5667, name: word/document.xml
3121          0xC31           Zip archive data, at least v2.0 to extract, compressed si
ize: 1214, name: word/_rels/document.xml.rels
3757          0xEAD           Zip archive data, at least v2.0 to extract, compressed si
size: 12288, name: word/vbaProject.bin
7763          0x1E53          Zip archive data, at least v2.0 to extract, compressed si
size: 7643, name: word/theme/theme1.xml
9448          0x24E8          Zip archive data, at least v2.0 to extract, compressed si
ize: 277, name: word/_rels/vbaProject.bin.rels
9699          0x25E3          Zip archive data, at least v2.0 to extract, compressed si
ize: 2784, name: word/vbaData.xml
10414         0x28AE          Zip archive data, at least v2.0 to extract, compressed si
size: 3741, name: word/settings.xml
11707         0x2DBB          Zip archive data, at least v2.0 to extract, compressed si
ize: 260, name: customXml/item1.xml
11966         0x2EBE          Zip archive data, at least v2.0 to extract, compressed si
ize: 341, name: customXml/itemProps1.xml
12285         0x2FFD          Zip archive data, at least v2.0 to extract, compressed si
```

Using the **xxd command** I need to view the header of the file to view the **magic bytes** (the important thing that declare the format of the file) , the first 8 bytes of the files reveal that the magic bytes is **0000 0000**, now that's why it changes it format before!

```
┌──(env)(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/Downloads/chal/File
└─$ xxd chall | head -10
00000000: 0000 0000 1400 0600 0800 0000 2100 1f10  ............!...
00000010: 2865 ac01 0000 b606 0000 1300 0802 5b43  (e............[C
00000020: 6f6e 7465 6e74 5f54 7970 6573 5d2e 786d  ontent_Types].xm
00000030: 6c20 a204 0228 a000 0200 0000 0000 0000  l ...(..........
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000060: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000080: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000090: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

By doing some research, we found out that the magic bytes of a word file is **50 4B 03 04.**





4D 5A	MZ	0	cpl ocx ax iec ime rs tsp fon efi	descendants (including NE and PE)
53 4D 53 4E 46 32 30 30	SMSNF200	0	ssp	SmartSniff Packets File [22]
5A 4D	ZM	0	exe	DOS ZM executable and its descendants (rare)
50 4B 03 04 50 4B 05 06 (empty archive) 50 4B 07 08 (spanned archive)	PK ETX EOT PK ENQ ACK PK BEL BS	0	zip aar apk docx epub ipa jar kmz maff msix odp ods odt pk3 pk4	zip file format and formats based on it, such as EPUB, JAR, ODF, OOXML

Now we utilize a tool called **hexedit** to edit the magic bytes of the file

```
┌──(env)(kali㉿NawfalMatebook)-[/mnt/c/Users
└─$ hexedit chall
```

Before changes:

```
00000000   00 00 00 00  14 00 06 00  08 00 00 00  21 00 1F 10  28 65 AC 01  ............!...(e..
00000014   00 00 B6 06  00 00 13 00  08 02 5B 43  6F 6E 74 65  6E 74 5F 54  ..........[Content_T
00000028   79 70 65 73  5D 2E 78 6D  6C 20 A2 04  02 28 A0 00  02 00 00 00  ypes].xml ...(......
0000003C   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000050   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000064   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000078   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
0000008C   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000000A0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000000B4   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000000C8   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000000DC   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000000F0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000104   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000118   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
0000012C   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000140   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000154   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000168   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
0000017C   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000190   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000001A4   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000001B8   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000001CC   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000001E0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000001F4   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
---  chall       --0x0/0x7D81--0%-------------------------------------------------------------------
```

After changes:

```
00000000   50 4B 03 04  14 00 06 00  08 00 00 00  21 00 1F 10  28 65 AC 01  PK..........!...(e..
00000014   00 00 B6 06  00 00 13 00  08 02 5B 43  6F 6E 74 65  6E 74 5F 54  ..........[Content_T
00000028   79 70 65 73  5D 2E 78 6D  6C 20 A2 04  02 28 A0 00  02 00 00 00  ypes].xml ...(......
0000003C   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000050   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000064   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000078   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
0000008C   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000000A0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000000B4   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000000C8   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000000DC   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000000F0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000104   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000118   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
0000012C   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000140   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000154   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000168   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
0000017C   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
00000190   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000001A4   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000001B8   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000001CC   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000001E0   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
000001F4   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ....................
-**  chall       --0x4/0x7D81--0%-------------------------------------------------------------------
```

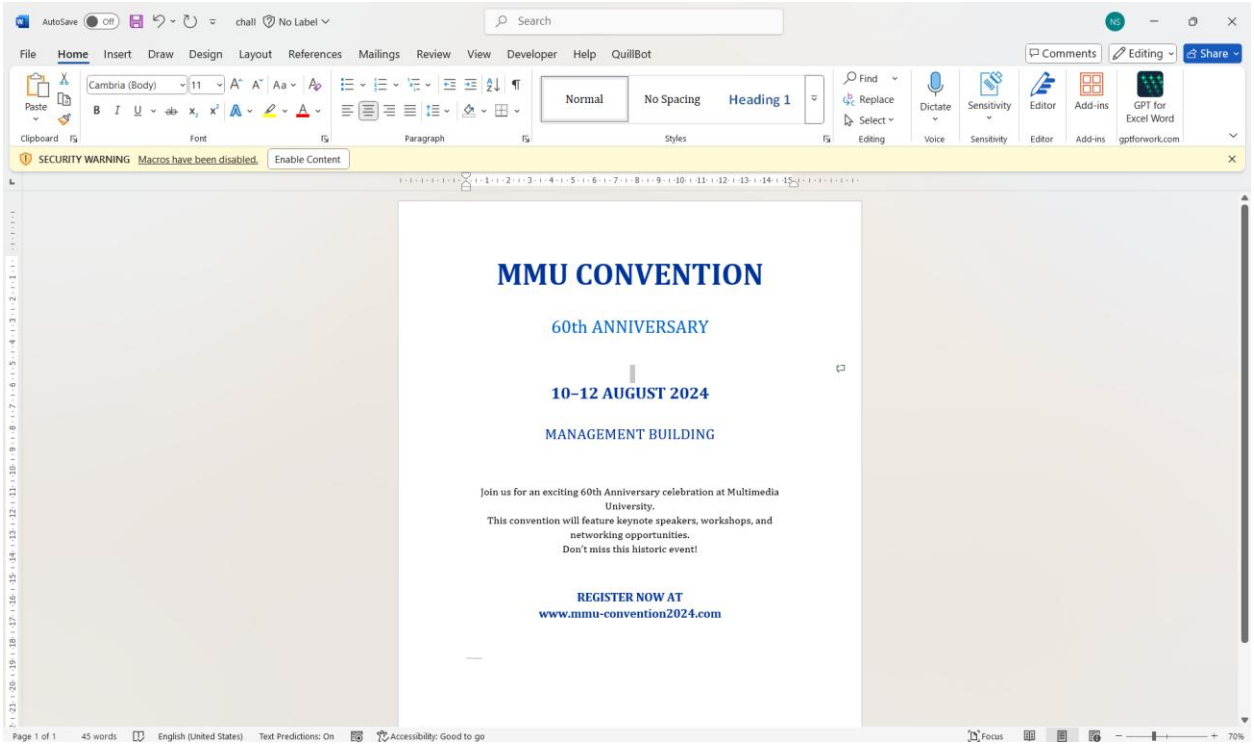
Now it is revealed that the files is actually a **Microsoft Word Files !**

```
┌──(env)(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/
└─$ file chall
chall: Microsoft Word 2007+
```

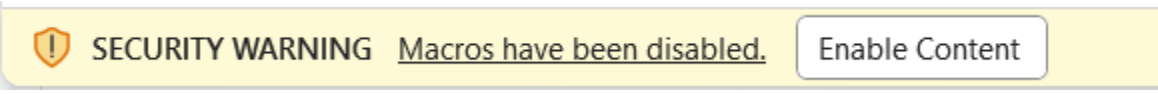
Lets **rename** the file to open the it in Microsoft word.

```
┌──(env)(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob
└─$ mv chall chall.docm
```

Opening the file gets us an invitation to a **MMU Convention**, which I don't find anything odd except.....



A macros warning!, someone must have tempered with the macros of the word file



How to view the file macros? There is actually a tool for that called olevba in oletools

Guide to installation:

1) **Kali cannot directly install the tool using pip3 because of security issues so we must create a virtual environment using the command below**

```
┌──(kali㉿NawfalMatebook)-[/mnt/
└─$ python3 -m venv env

┌──(kali㉿NawfalMatebook)-[/mnt/
└─$ source env/bin/activate
```

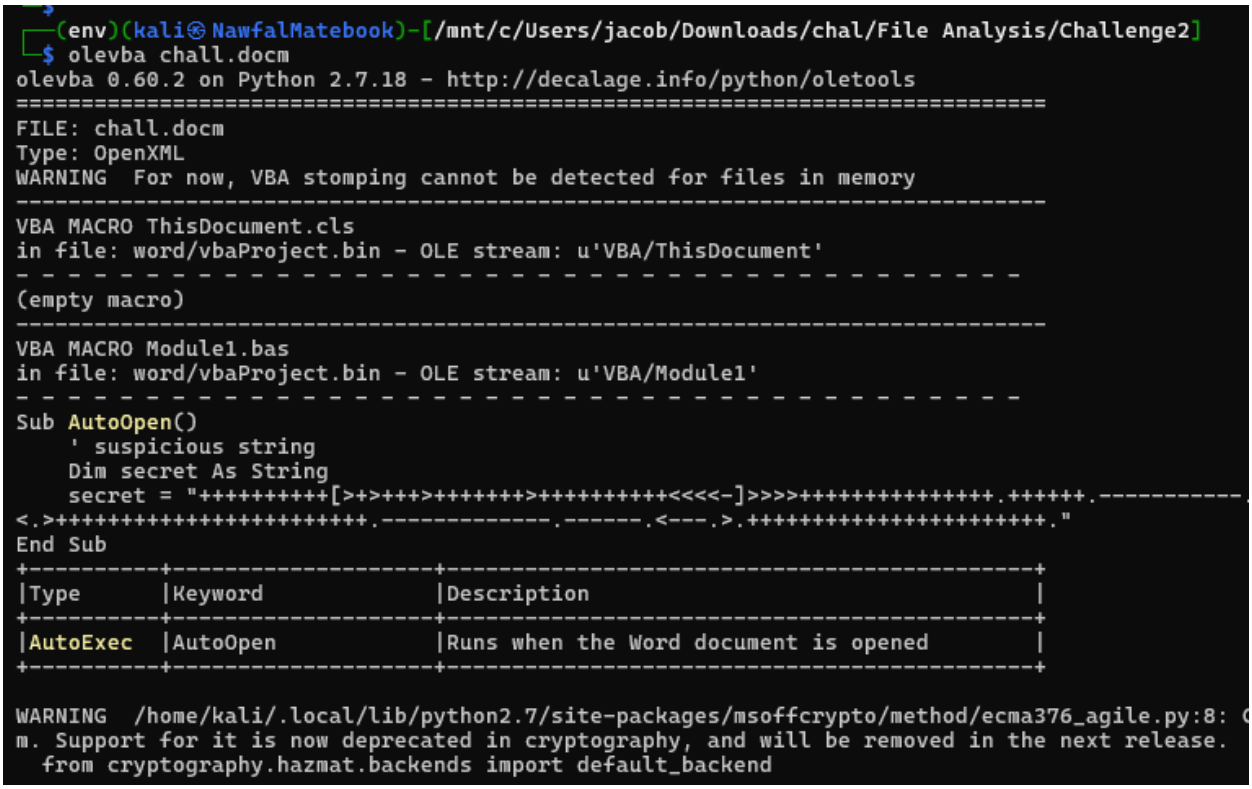
2) **Now we can safely install the oletools inside our kali**

```
┌──(env)(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/Downloads/chal/File An
└─$ pip3 install oletools
Collecting oletools
  Using cached oletools-0.60.2-py2.py3-none-any.whl.metadata (16 kB)
Collecting pyparsing<4,>=2.1.0 (from oletools)
  Using cached pyparsing-3.2.3-py3-none-any.whl.metadata (5.0 kB)
Collecting olefile>=0.46 (from oletools)
  Using cached olefile-0.47-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting easygui (from oletools)
  Using cached easygui-0.98.3-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting colorclass (from oletools)
  Using cached colorclass-2.2.2-py2.py3-none-any.whl.metadata (5.2 kB)
Collecting pcodedmp>=1.2.5 (from oletools)
  Using cached pcodedmp-1.2.6-py2.py3-none-any.whl.metadata (11 kB)
Collecting msoffcrypto-tool (from oletools)
  Using cached msoffcrypto_tool-5.4.2-py3-none-any.whl.metadata (10 kB)
Collecting cryptography>=39.0 (from msoffcrypto-tool->oletools)
  Using cached cryptography-45.0.6-cp311-abi3-manylinux_2_34_x86_64.whl.m
Collecting cffi>=1.14 (from cryptography>=39.0->msoffcrypto-tool->oletool
  Using cached cffi-1.17.1-cp313-cp313-manylinux_2_17_x86_64.manylinux201
Collecting pycparser (from cffi>=1.14->cryptography>=39.0->msoffcrypto-to
  Using cached pycparser-2.22-py3-none-any.whl.metadata (943 bytes)
Using cached oletools-0.60.2-py2.py3-none-any.whl (989 kB)
Using cached pyparsing-3.2.3-py3-none-any.whl (111 kB)
Using cached olefile-0.47-py2.py3-none-any.whl (114 kB)
Using cached pcodedmp-1.2.6-py2.py3-none-any.whl (30 kB)
```

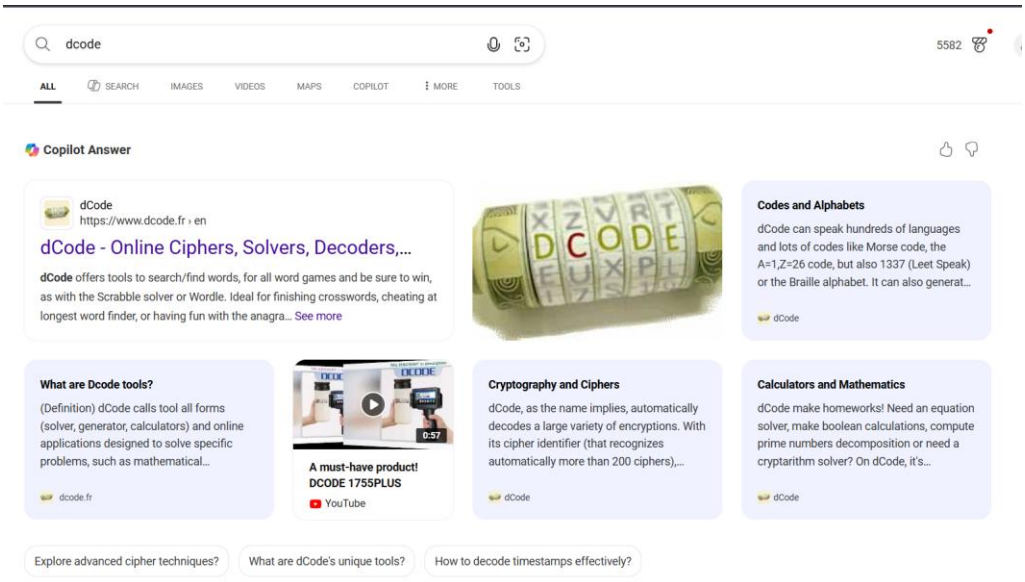
3) **Check if the olevba is now useable inside our kali.**

```
┌──(env)(kali㉿NawfalMatebook)-[/mnt/c/Users/jacob/Downloads/chal/File Analys
└─$ olevba
olevba 0.60.2 on Python 2.7.18 - http://decalage.info/python/oletools

olevba.py

olevba is a script to parse OLE and OpenXML files such as MS Office documents
(e.g. Word, Excel), to extract VBA Macro code in clear text, deobfuscate
and analyze malicious macros.
XLM/Excel 4 Macros are also supported in Excel and SLK files.

Supported formats:
    - Word 97-2003 (.doc, .dot), Word 2007+ (.docm, .dotm)
    - Excel 97-2003 (.xls), Excel 2007+ (.xlsm, .xlsb)
    - PowerPoint 97-2003 (.ppt), PowerPoint 2007+ (.pptm, .ppsm)
    - Word/PowerPoint 2007+ XML (aka Flat OPC)
    - Word 2003 XML (.xml)
    - Word/Excel Single File Web Page / MHTML (.mht)
    - Publisher (.pub)
    - SYLK/SLK files (.slk)
    - Text file containing VBA or VBScript source code
    - Password-protected Zip archive containing any of the above
    - raises an error if run with files encrypted using MS Crypto API RC4

Author: Philippe Lagadec - http://www.decalage.info
License: BSD, see source code or documentation
```

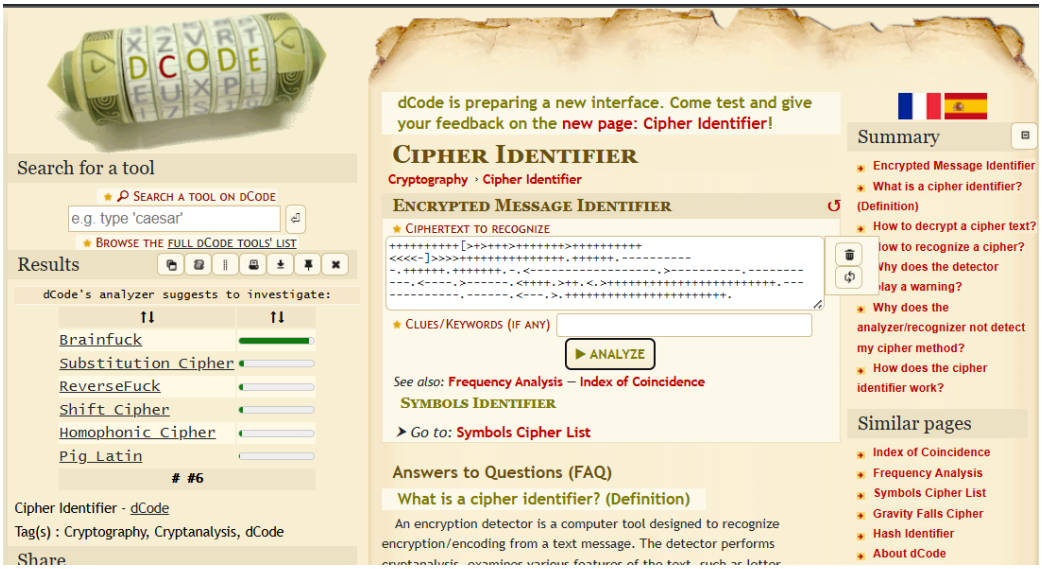
Ok now we can use the **olevba** to view the macros of the word file. **What is that? A suspicious string?!** Look like encryption to me

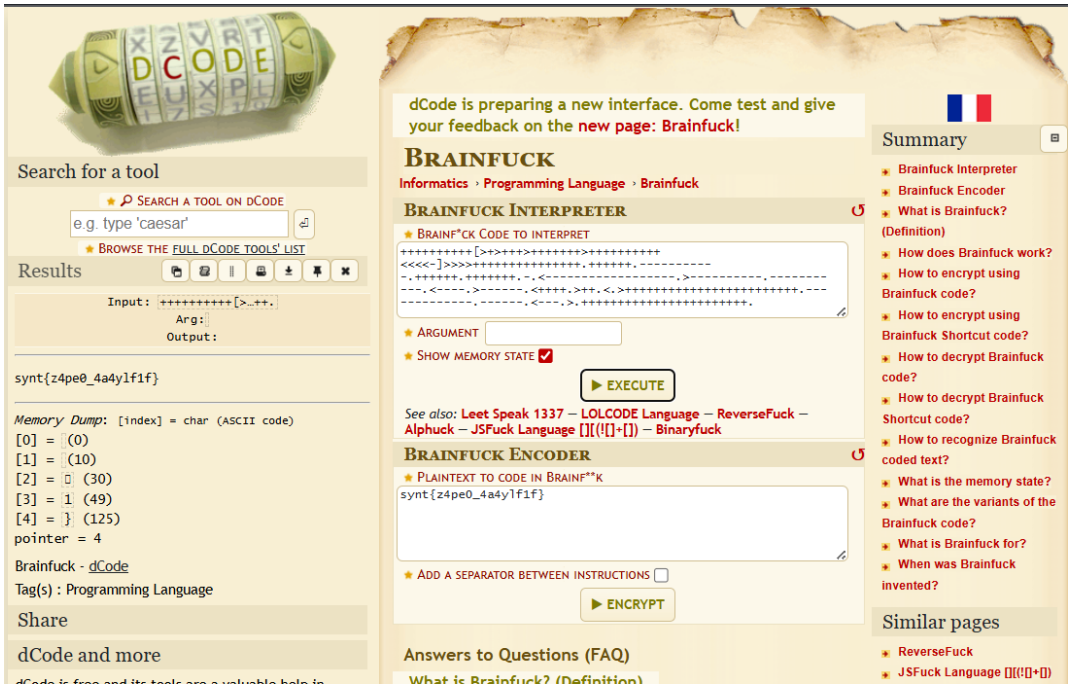


Using my favorite tool to decode, [Decrypt a Message - Cipher Identifier - Online Code Recognizer](), we can now analyze the encryption.



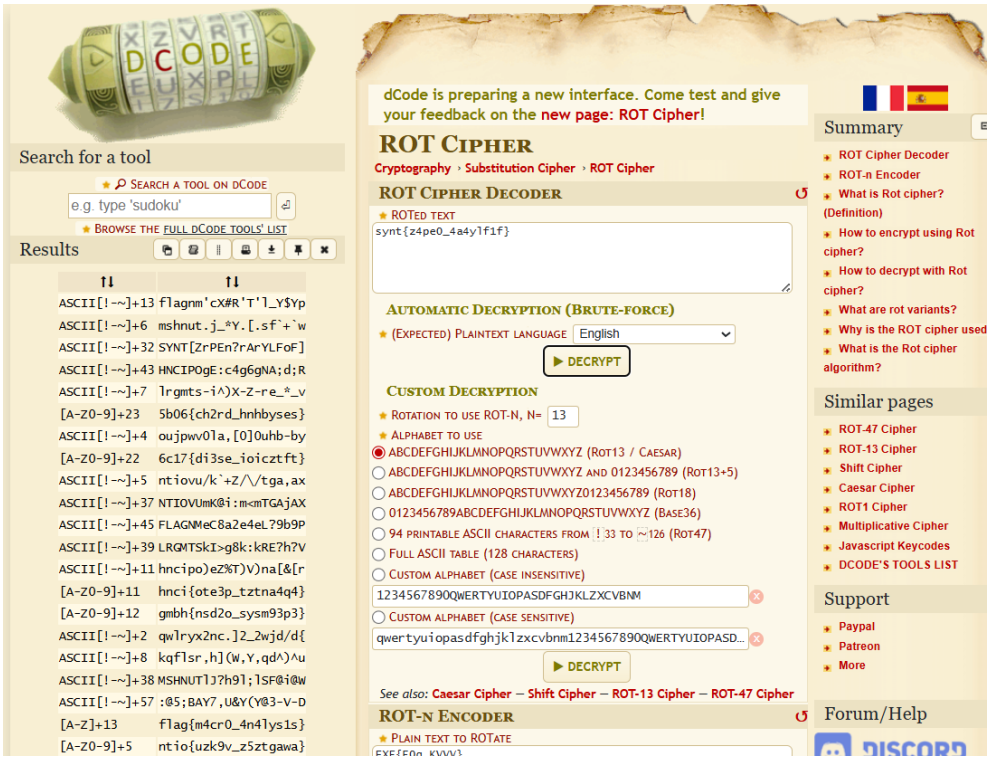
Oh it is a **brainf**k encryption**, interpreted the encryption reveal us with another encryption.





Using the same method as before we can analyze the encryption again and now we got a **ROT Cipher instead!**

Among the decode is the **FLAG!**



Flag - flag{m4cr0_4n4lys1s}