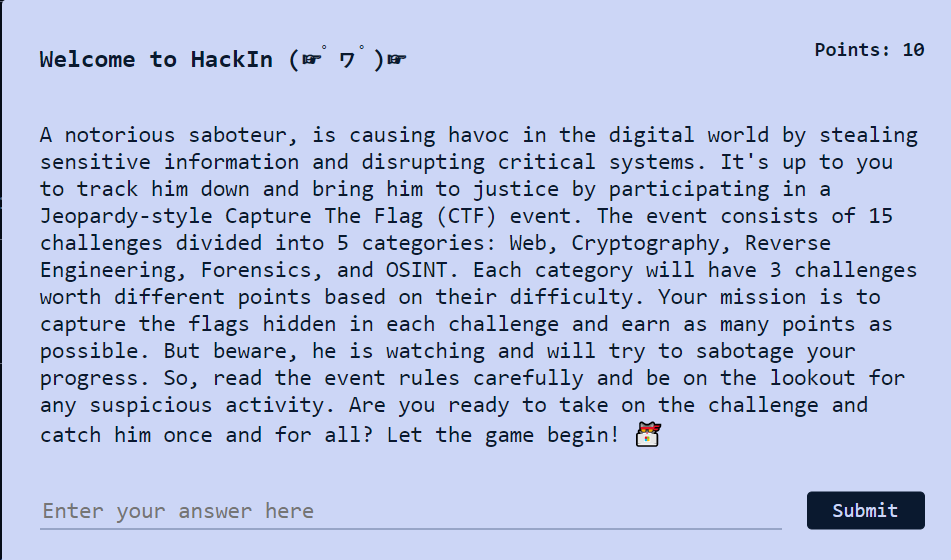
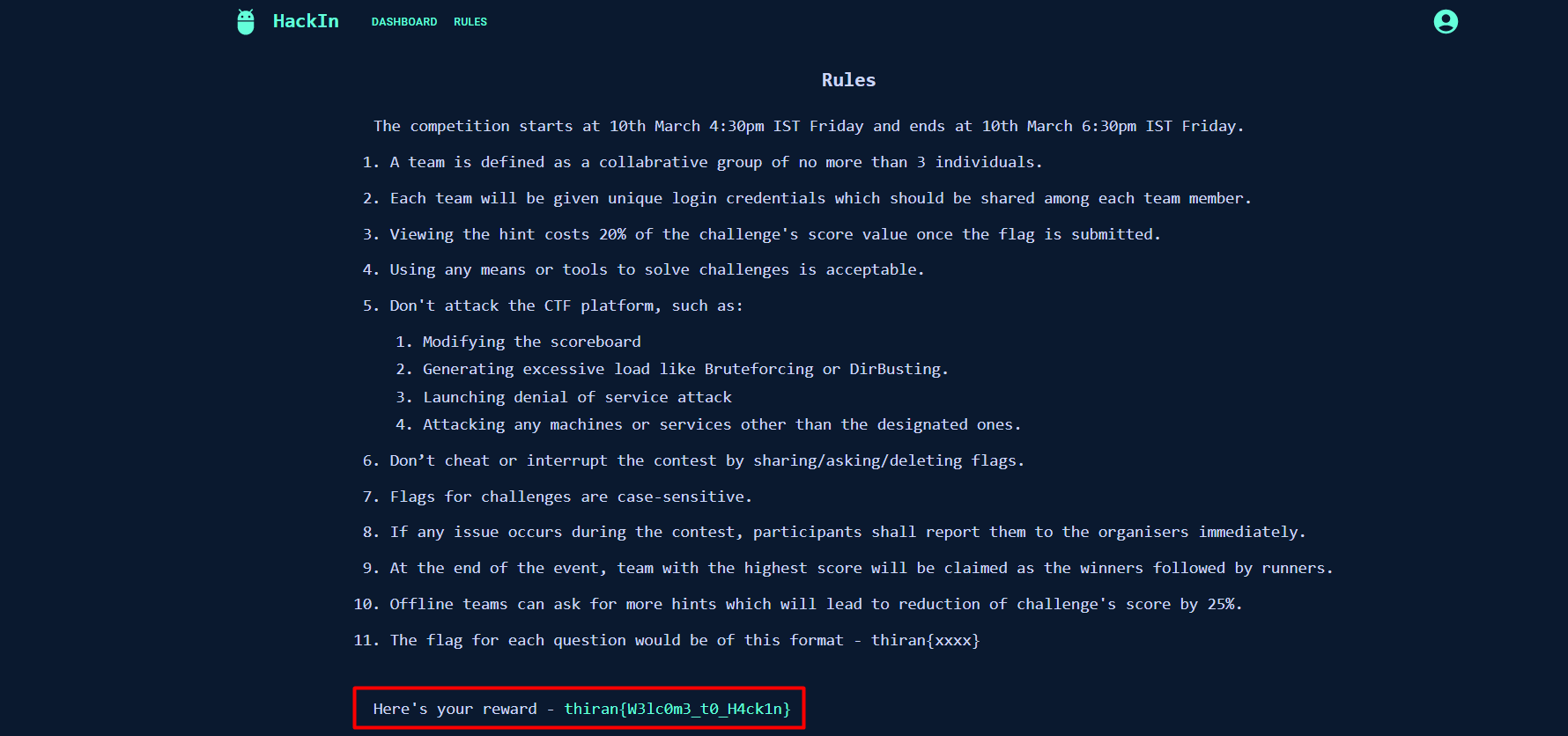
**HackIn- Thiran 2023 Writeups**



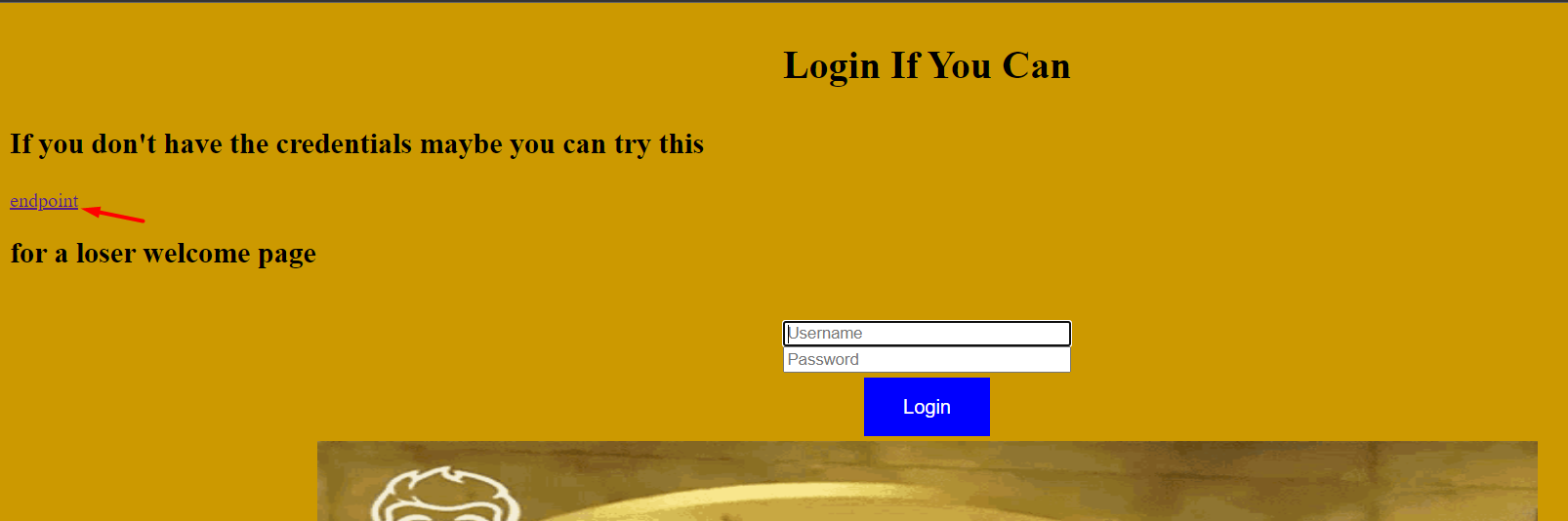
Navigating to the Rules page the flag could be seen at the bottom.



**--------------------------------------------------------------------------------------------------------------------------**

**WEB**

**Flask the Flag**



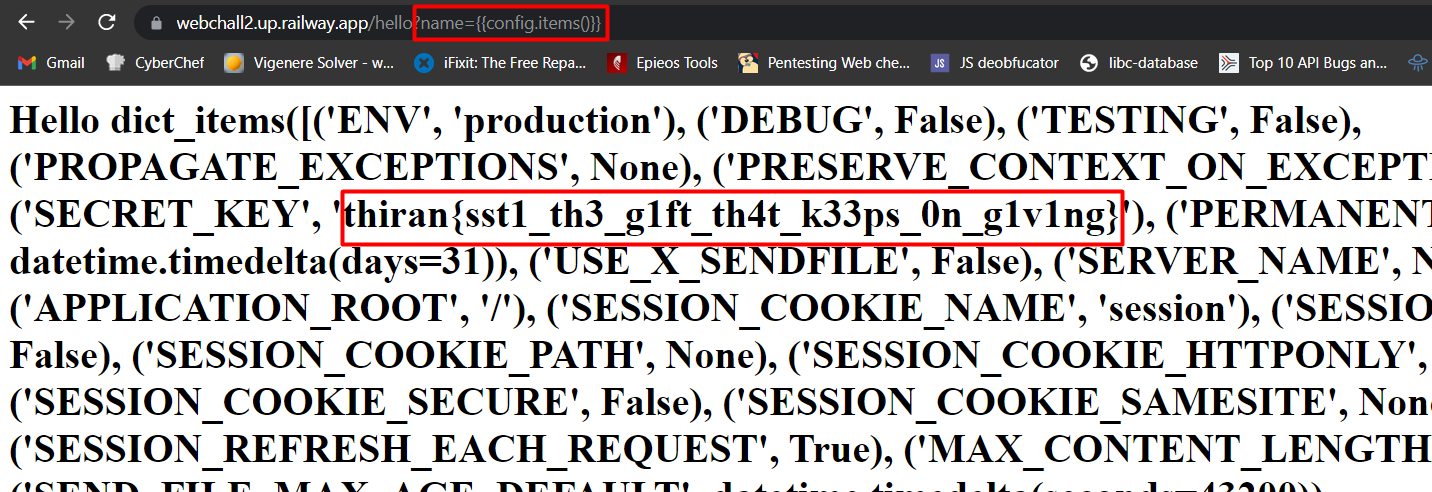
Navigating to the endpoint, the value in the **name** GET parameter is reflected in the webpage.



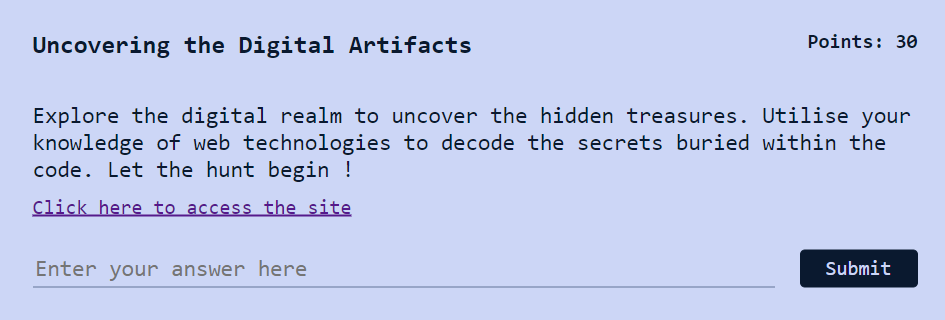
With this and the description mentioning the Flask backend language and server-side templates, it is obvious to try Server-Side Template Injection.

Payload: {{config.items()}}

Using this payload in the **name** parameter, the flag is obtained.

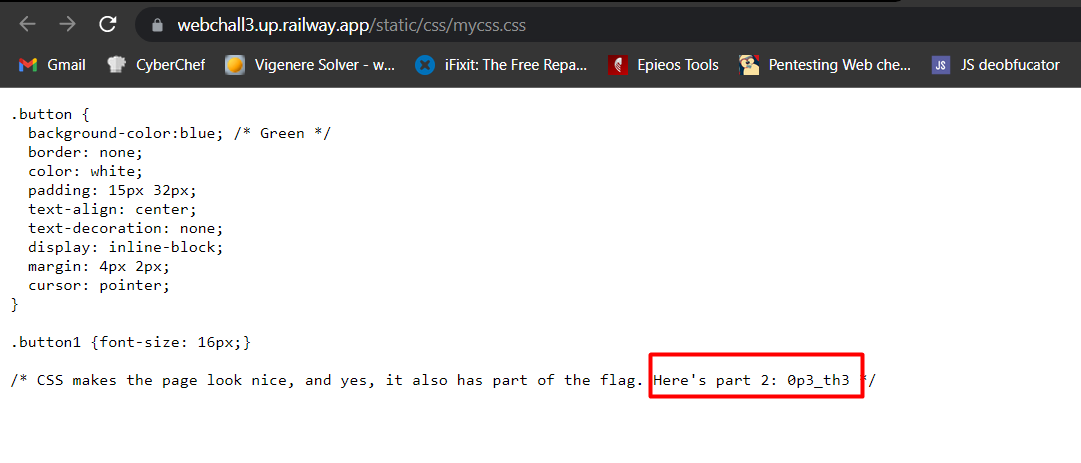


**Uncovering the Digital Artifacts**

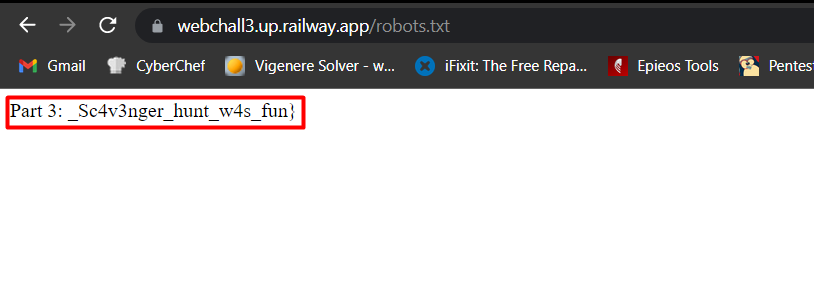


This is a simple challenge, as the flag is split into 3 parts and stored in HTML, and CSS files, and the last one is stored in the **robots.txt file** with a clue denoting the file in the JavaScript file.





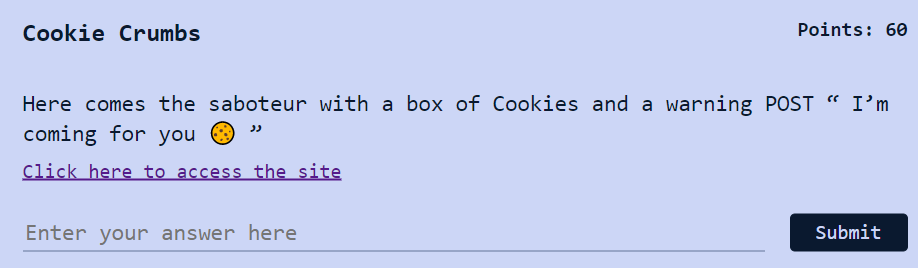


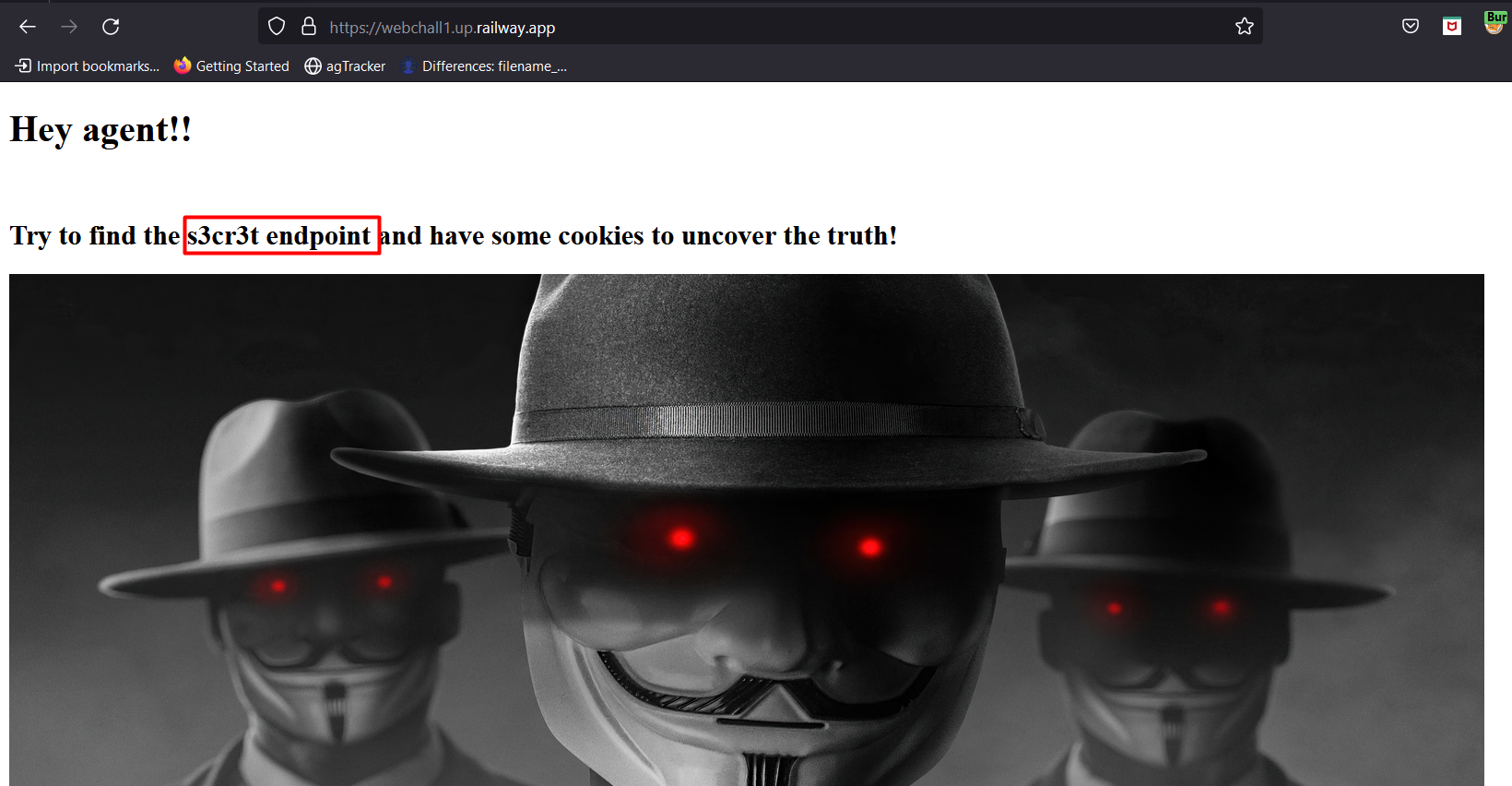


Flag: thiran{H0p3\_th3\_Sc4v3nger\_hunt\_w4s\_fun}

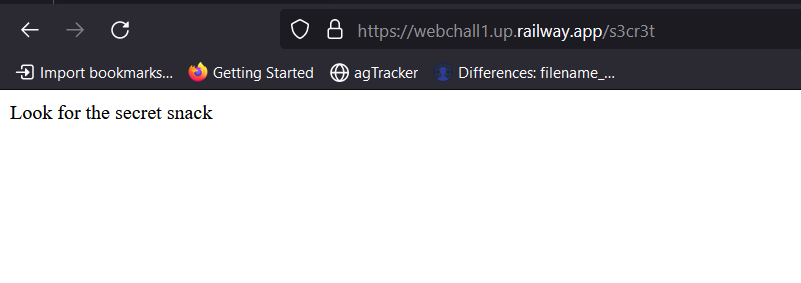
**--------------------------------------------------------------------------------------------------------------------------**

**Cookie Crumbs**

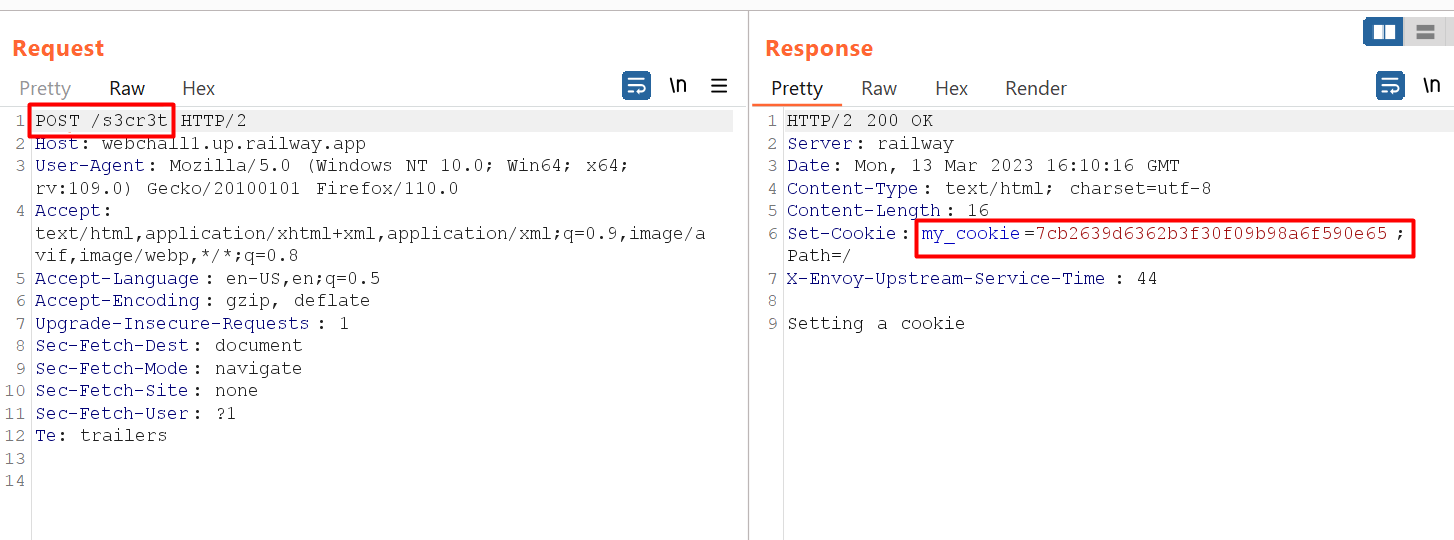




The word **s3cr3t** stands out on the page and accessing the endpoint gives the below response.



With the challenge description, it can be induced that the challenge is regarding cookies, and capitalizing the POST word indicates that a POST HTTP request must be used.



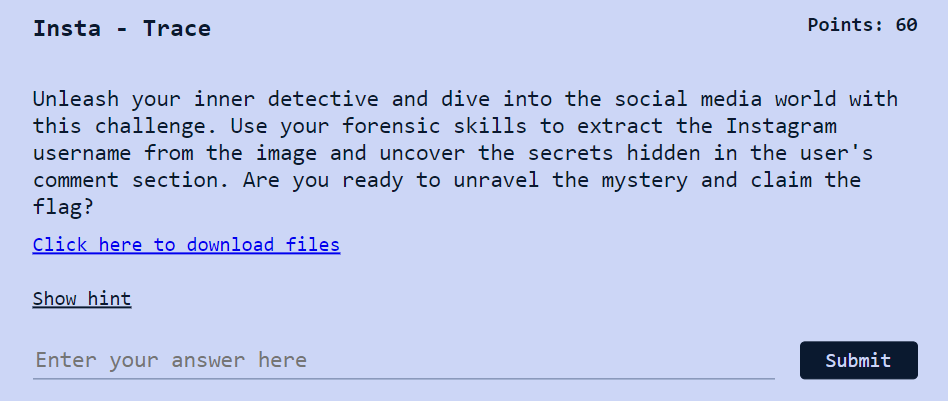
Sending a GET request with the cookie obtained, the flag is obtained.



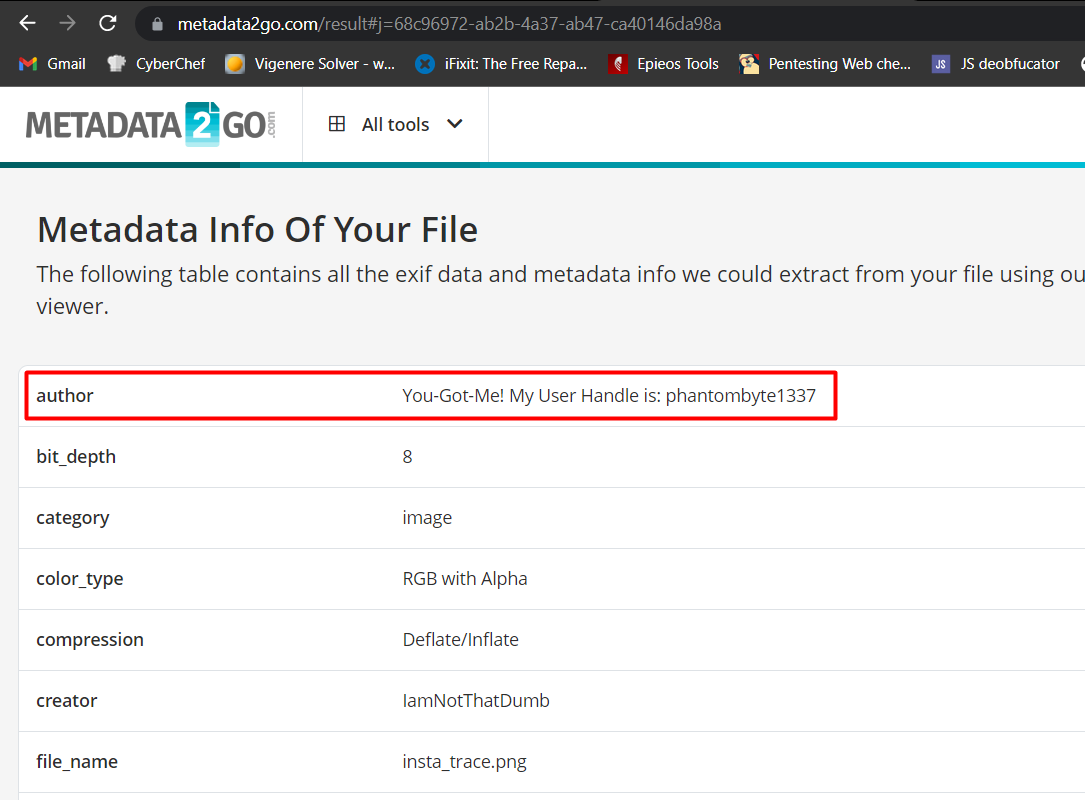
**--------------------------------------------------------------------------------------------------------------------------**

**OSINT**

**Insta Trace**



Viewing the metadata of the image, the user handle is revealed.

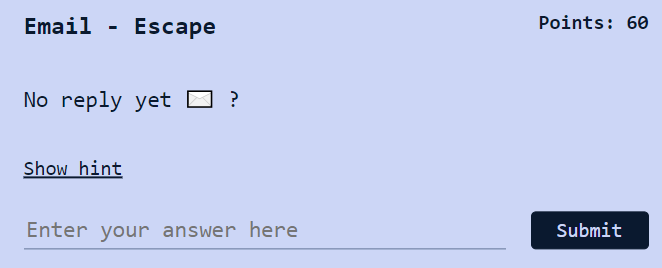


With the challenge name and description, the user handle belongs to Instagram and the flag is in the comments of a post by the user.

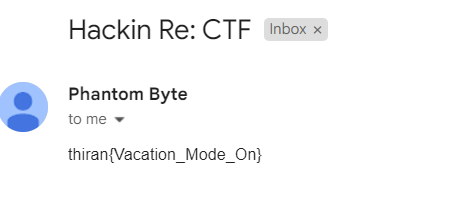
The other detail that can be obtained from the account is an email id.

Email: [phantobyte1337@gmail.com](mailto:phantobyte1337@gmail.com)

**--------------------------------------------------------------------------------------------------------------------------**

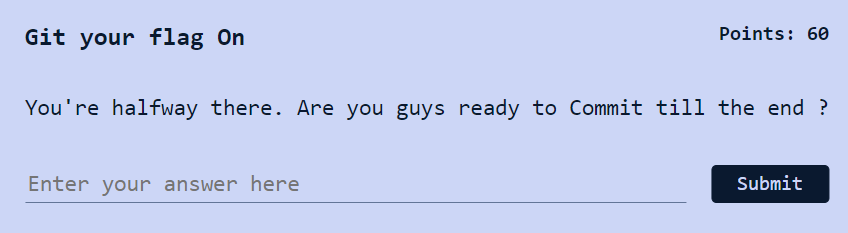
**Email Escape:**  


From the previous challenge, the email address is obtained, and sending an email to it gives us the flag as the **Auto Responder** or **Vacation Responder** is enabled for the account.

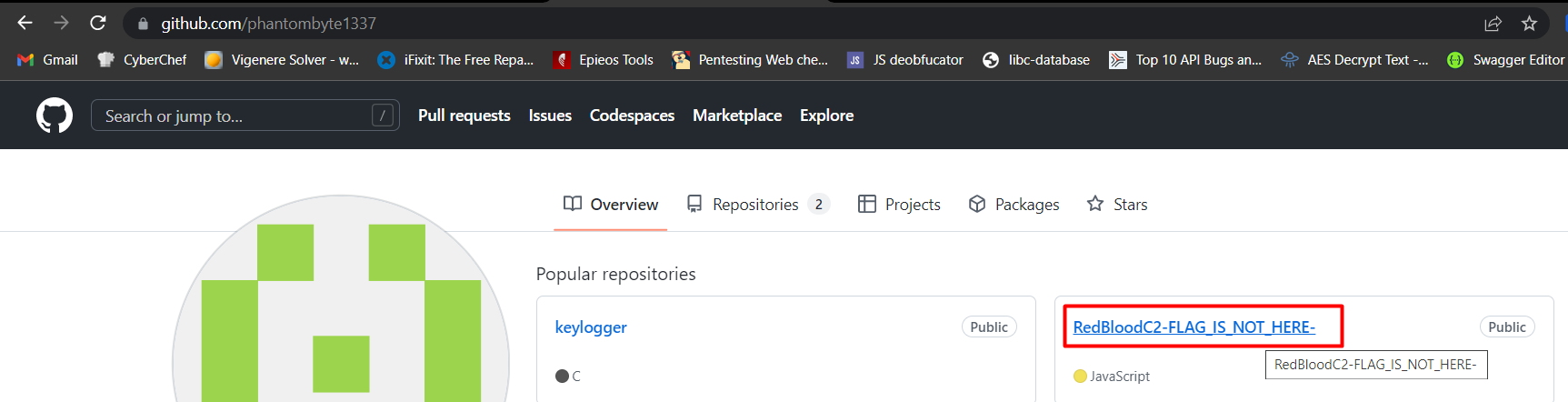


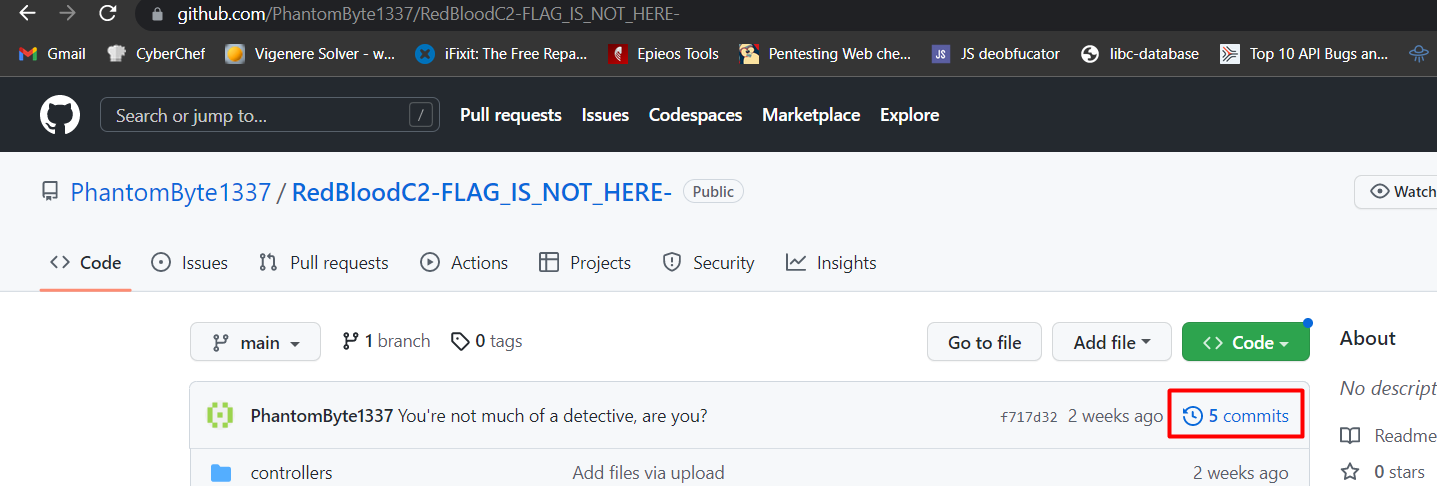
**--------------------------------------------------------------------------------------------------------------------------**

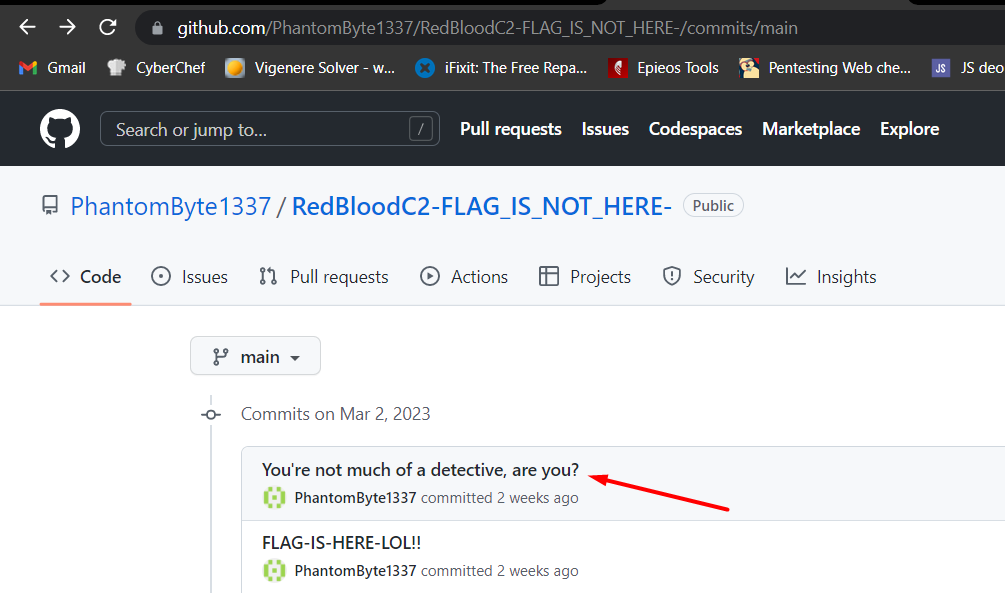
**Git your Flag on**

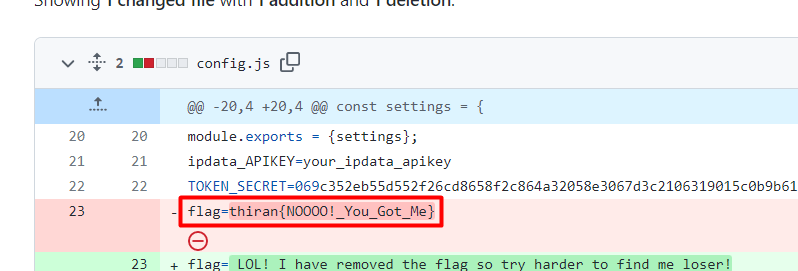


Looking for a GitHub account using the same username, an interesting account is identified and the flag is found using the commit history as denoted in the description.





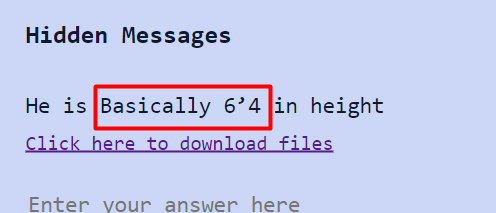




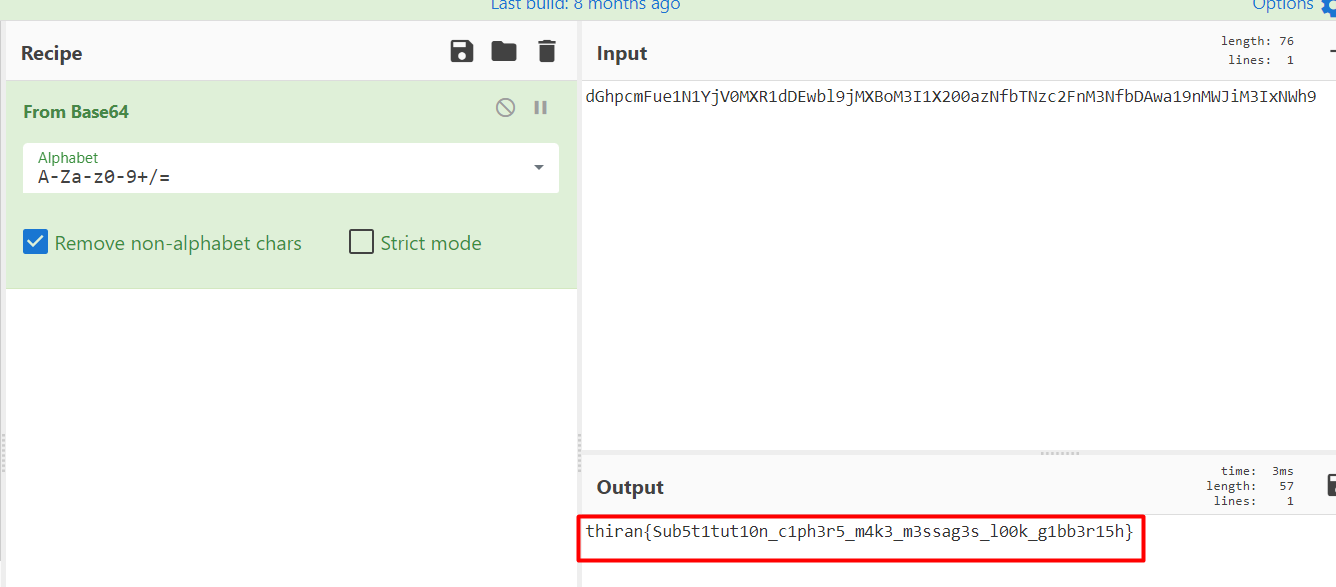
**--------------------------------------------------------------------------------------------------------------------------**

**Cryptography**

**Hidden Messages**

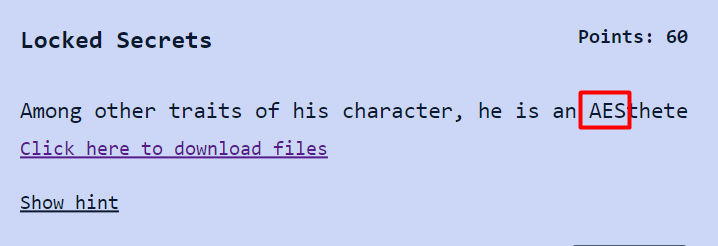


Base64 decoding the string, the flag is obtained.

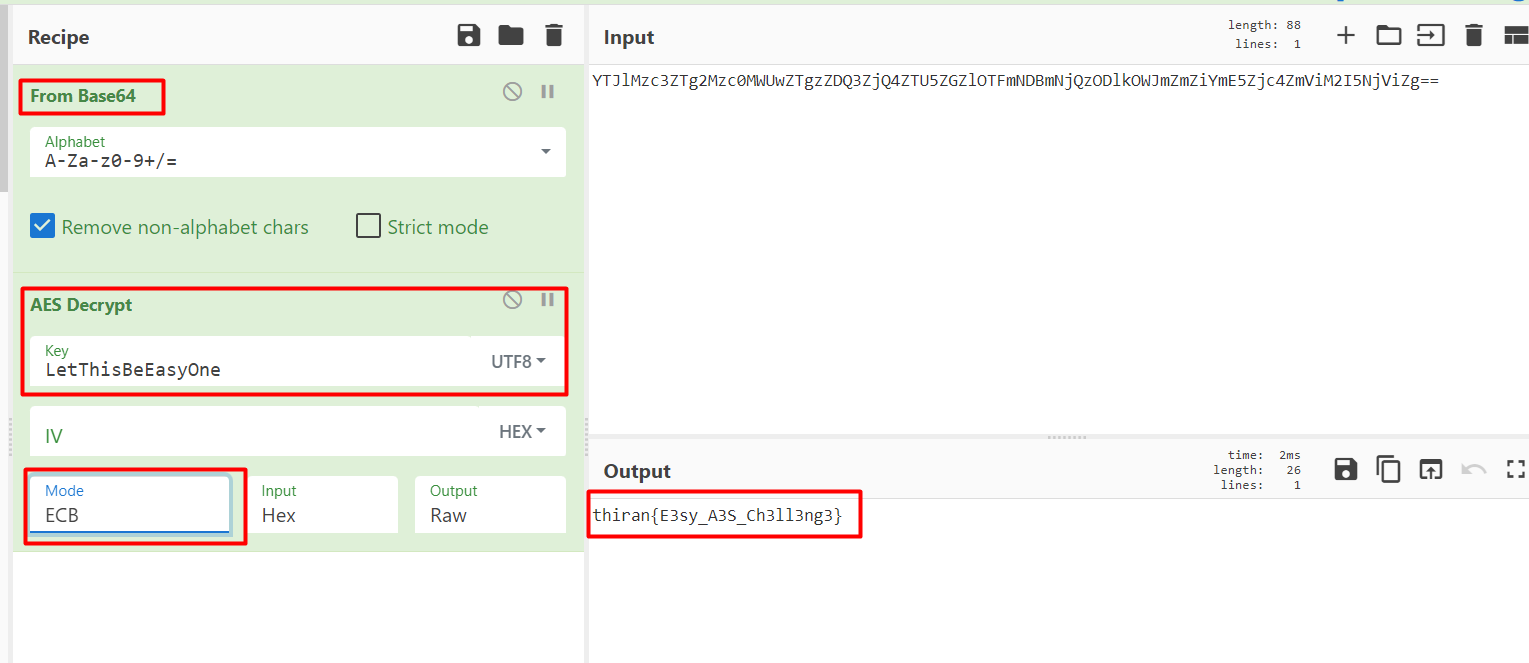


**--------------------------------------------------------------------------------------------------------------------------**

**Locked Secrets**

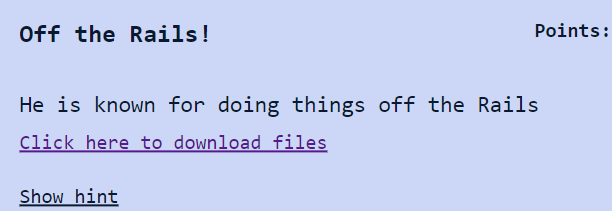


An AES encrypted string along with the key is provided. Just finding the correct mode and format of key and plugging the values to any decryptor gives the flag.



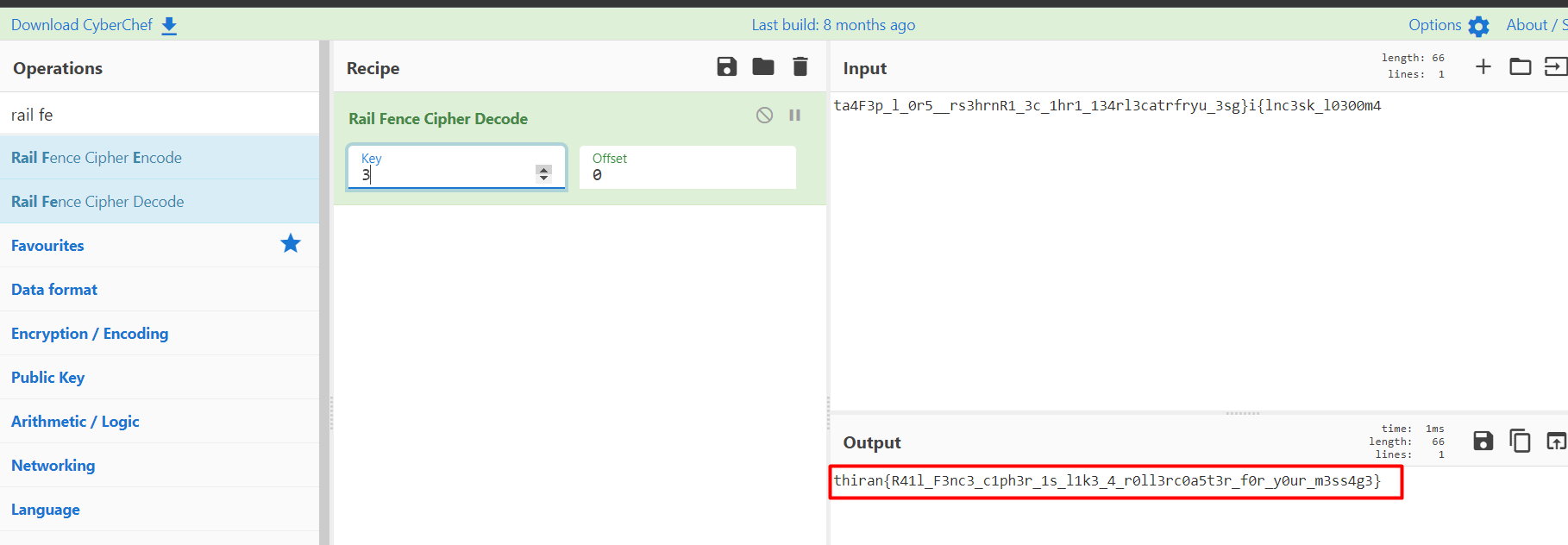
**--------------------------------------------------------------------------------------------------------------------------**

**Off the Rails**

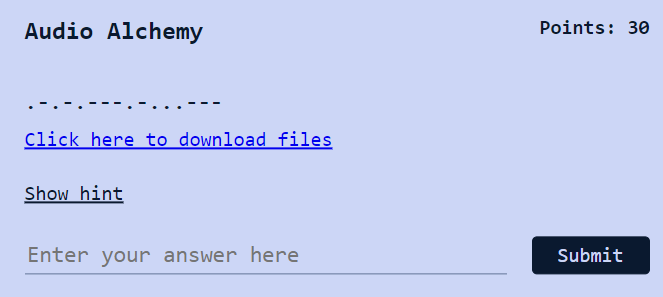


A rail fence encrypted string is provided and to obtain the flag the offset needs to be brute-forced.

Offset=3 gives the flag



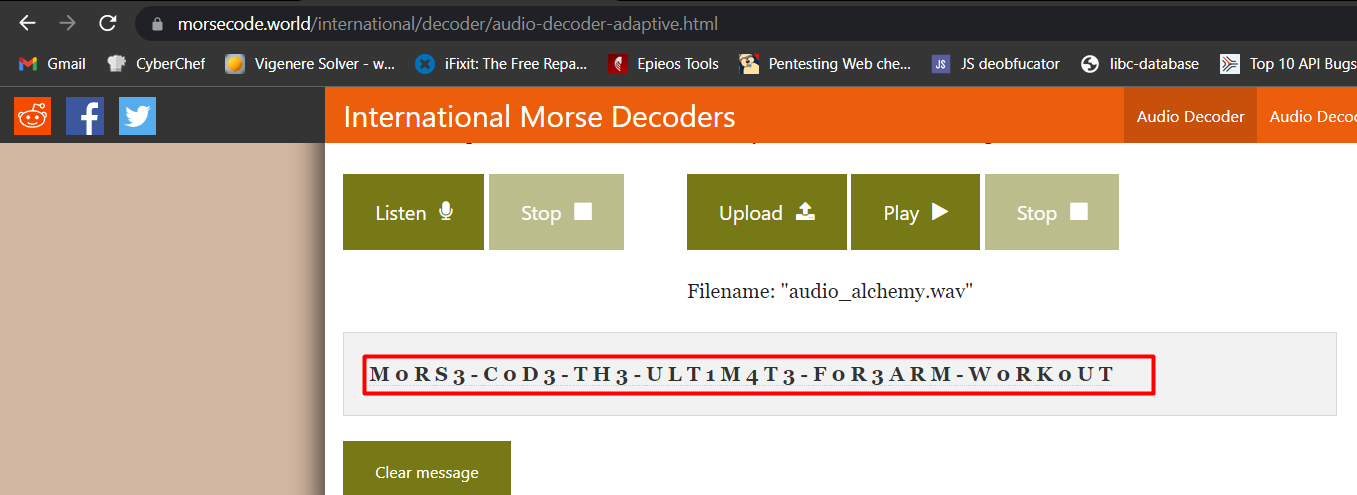
**Forensics**



The challenge is decoding an audio file containing data encoded using Morse Code.

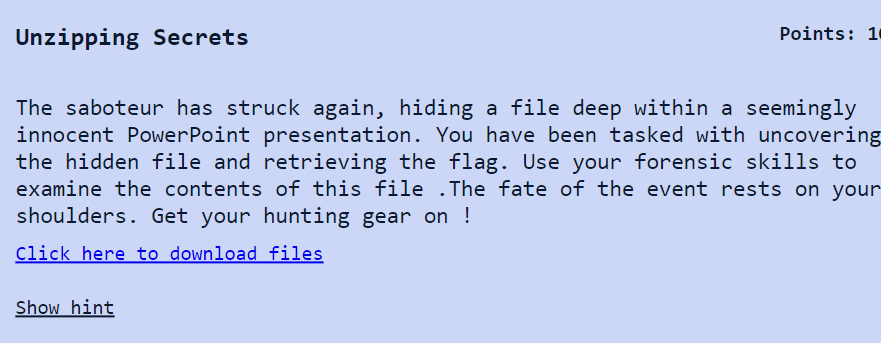
Finding the right decoder and uploading the audio file gives the flag.

Note: The text obtained must be used in between **thiran{}**



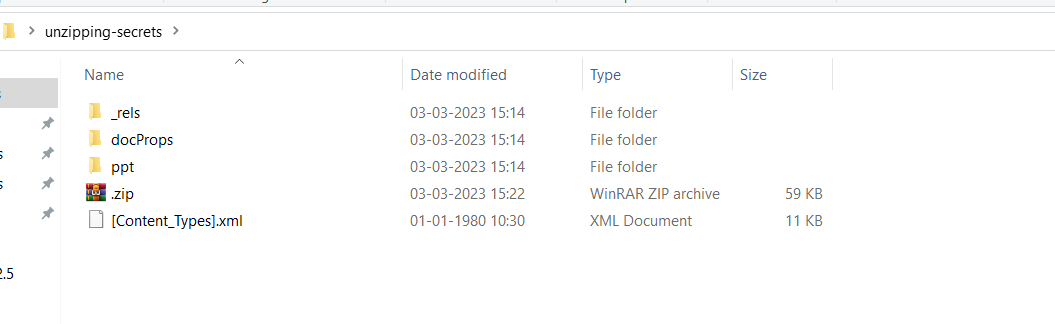
Flag: **thiran{M0RS3-C0D3-TH3-ULT1M4T3-F0R3ARM-W0RK0UT}**

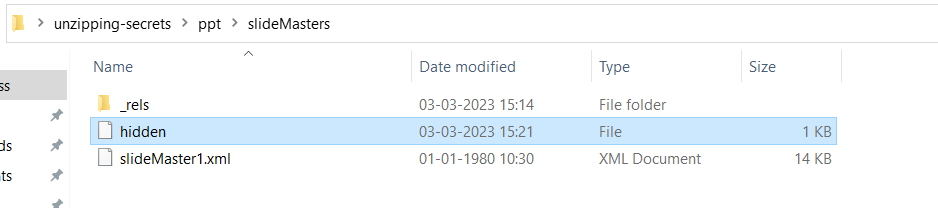
**--------------------------------------------------------------------------------------------------------------------------**

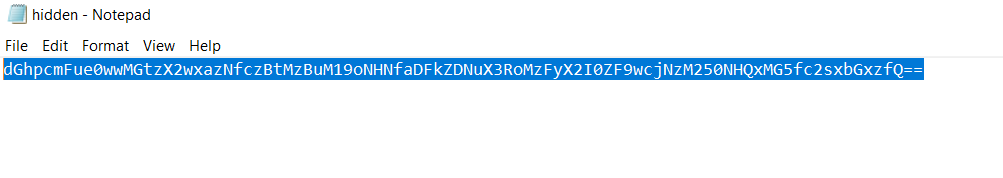


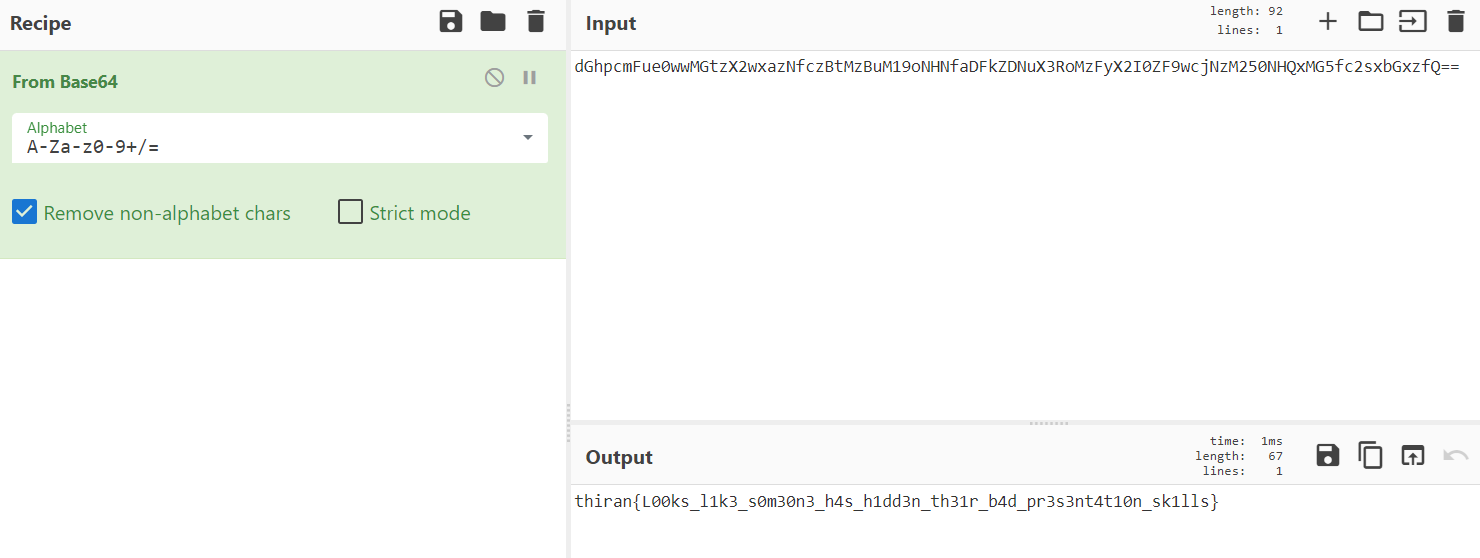
PPT or PPTM files in general are zip files with multiple folders and files. So, to solve this challenge,

* the extension of the file must be changed to**.zip**
* Extract all the files
* Flag is encoded as base64 string in a file named **hidden**



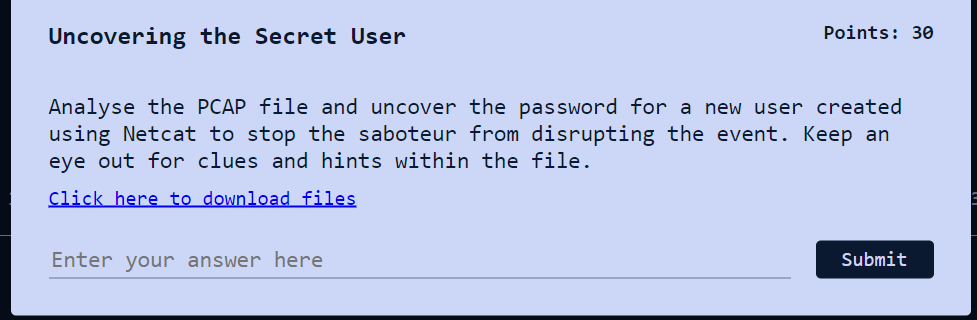






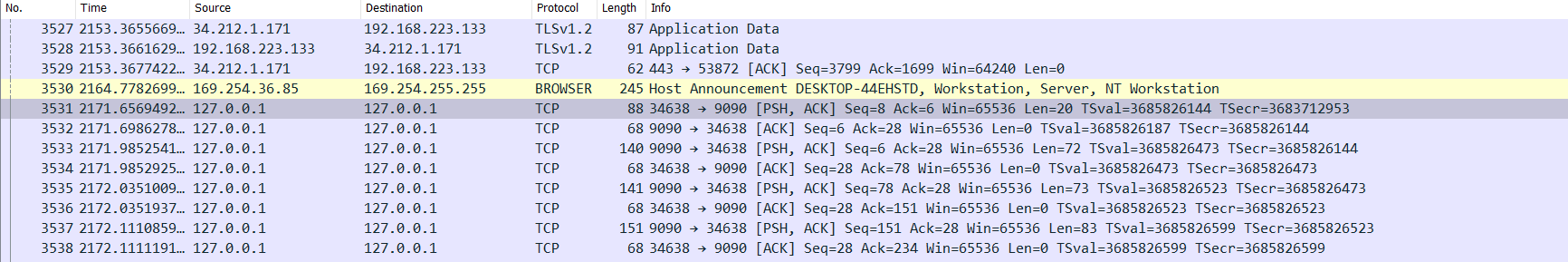
**--------------------------------------------------------------------------------------------------------------------------**

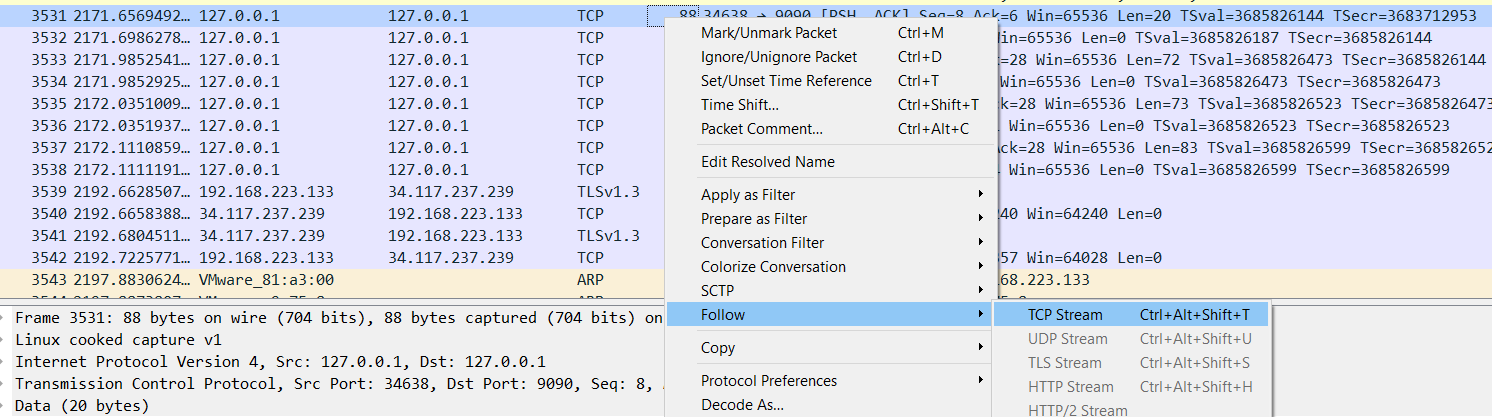
**Uncovering the Secret User**

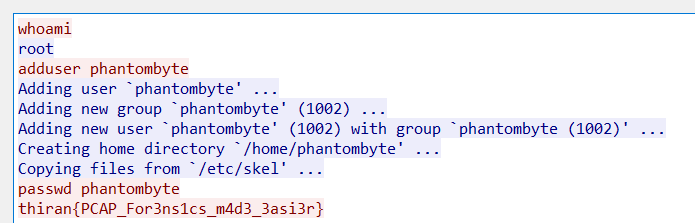


The challenge is to find the password of a new user being created by analyzing the PCAP file.

At the end of the file, we observe weird traffic within the same system’s local IP (127.0.0.1), and following the TCP stream various Linux commands to create a new user is observed along with the flag.

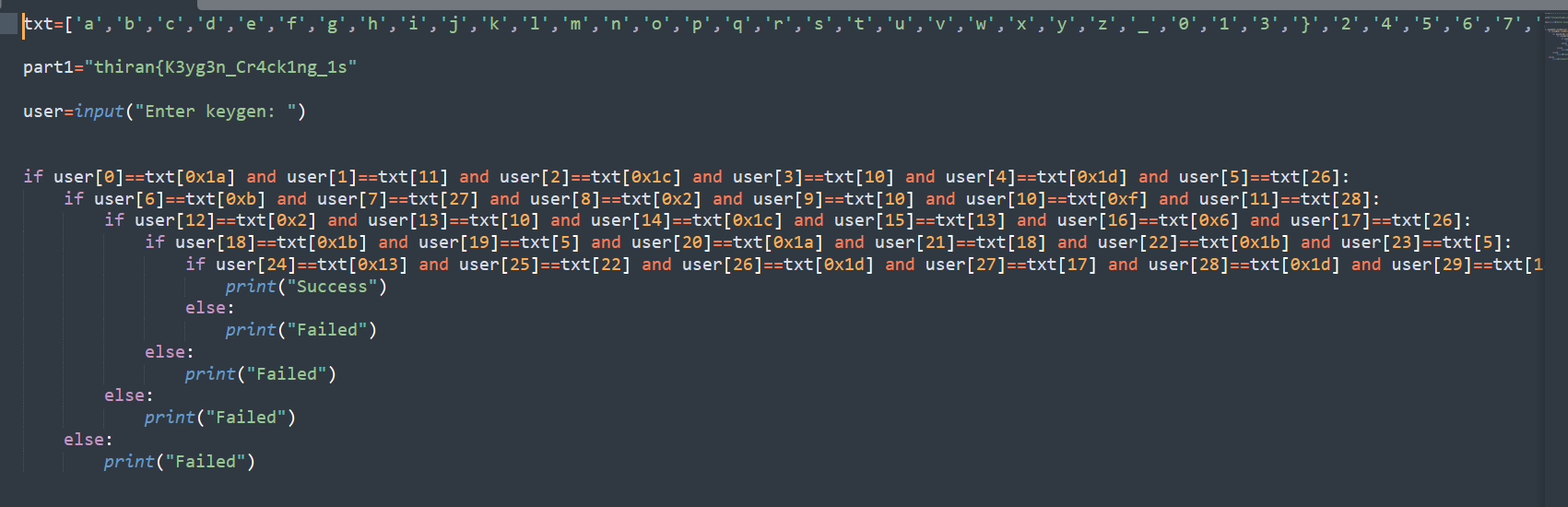






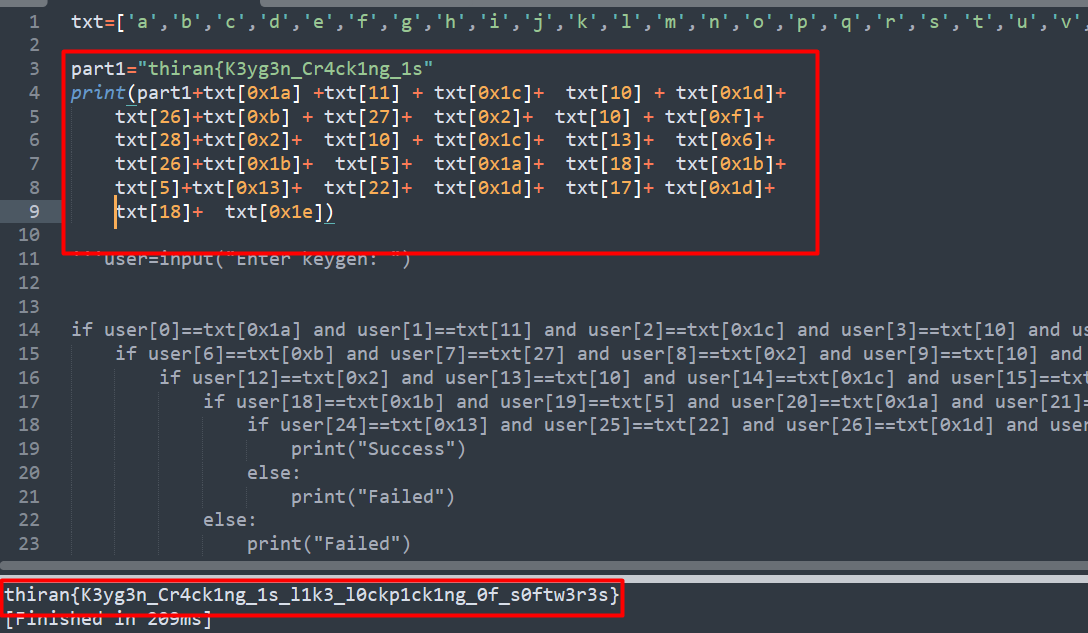
**Reversing**

Encoding Enigma



This is a simple string comparison but implemented using a list and indices with hexadecimal and integer formats.

To obtain the flag just print all the **txt[index]** variables.



For the remaining reversing challenges just uploading the binaries to Ghidra and analyzing the decompiled code gives the flag.