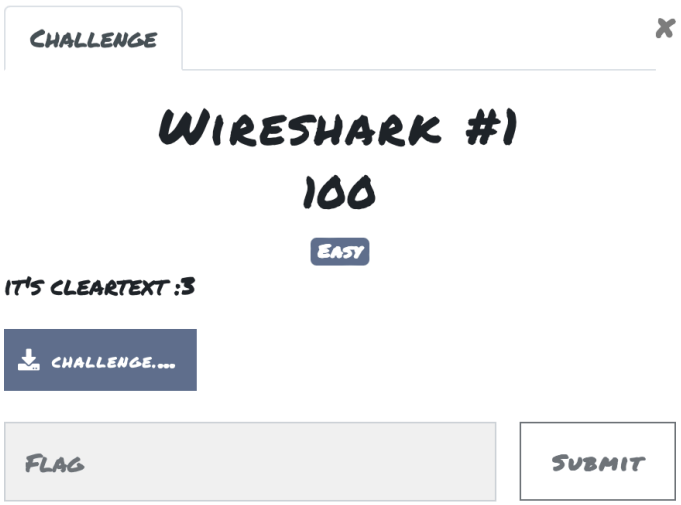
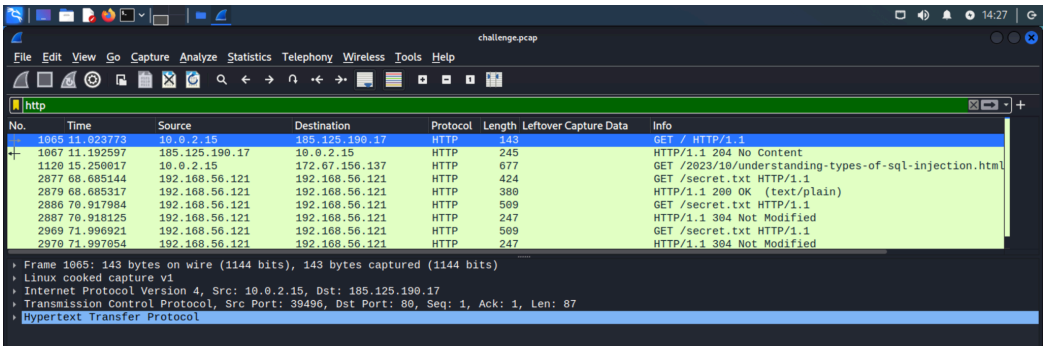
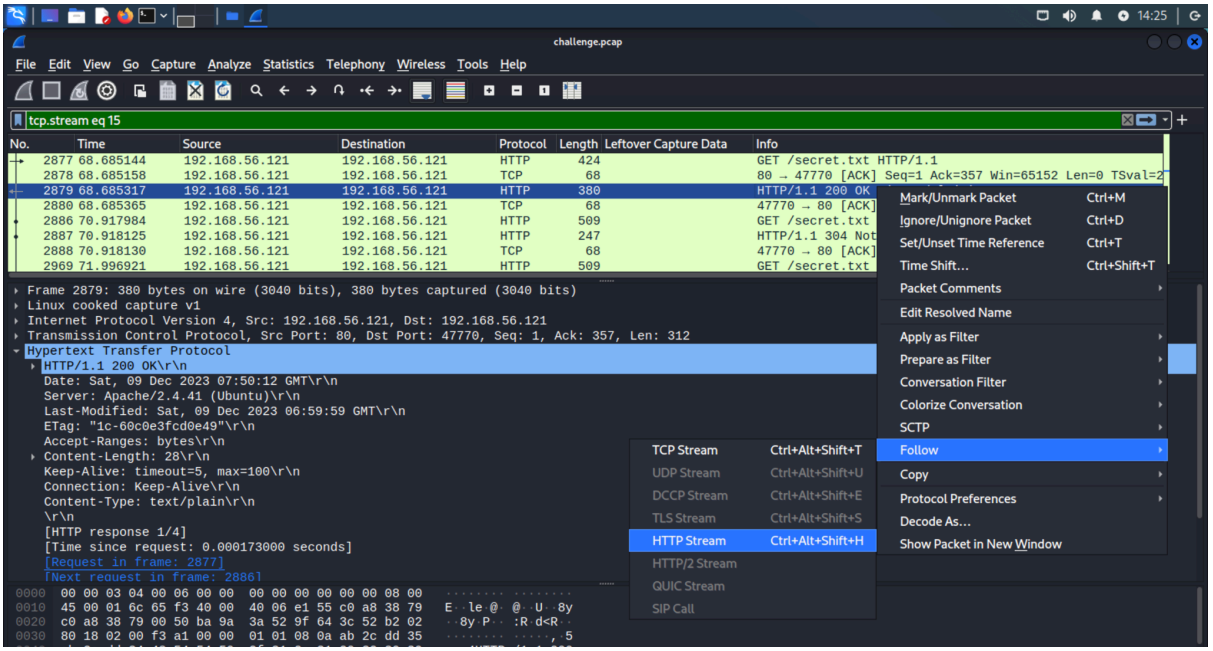
WIRESHARK #1

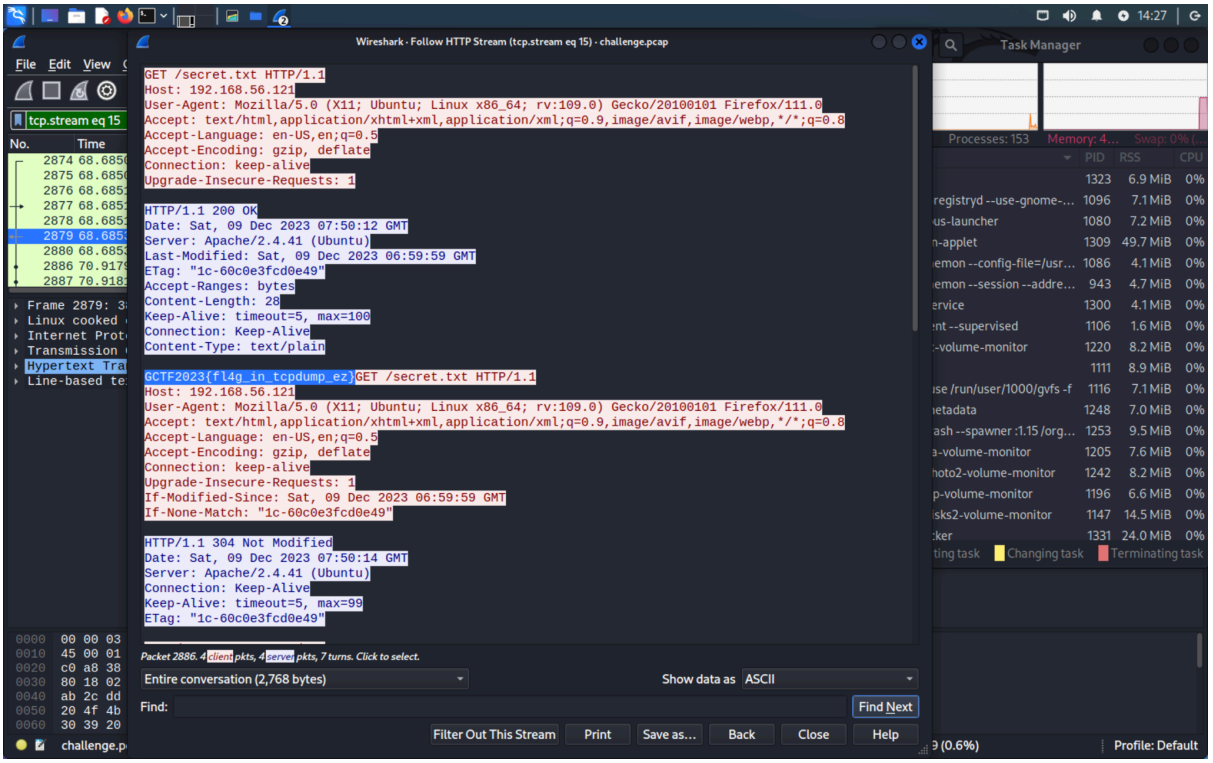




So firstly, I opened the file and applied a filter for http in the wireshark. Just like the hint given, it is clear text, and there is a suspicious plain txt file named “secret.txt” in the info.



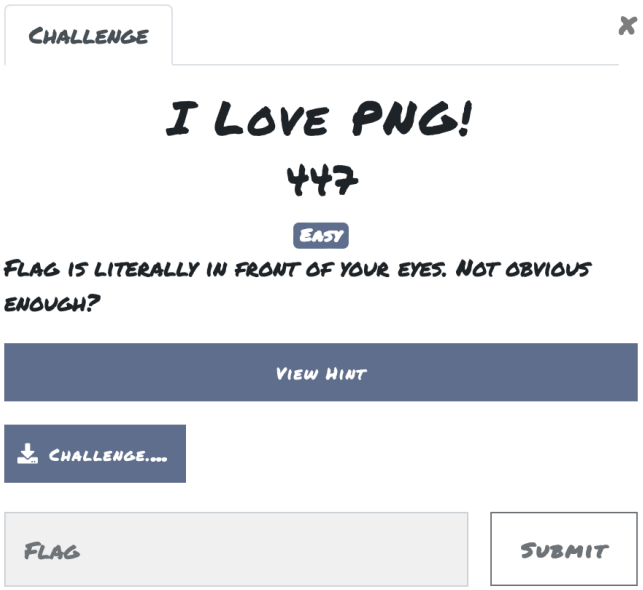
Then I just right click on the pcap, and follow the HTTP Stream.



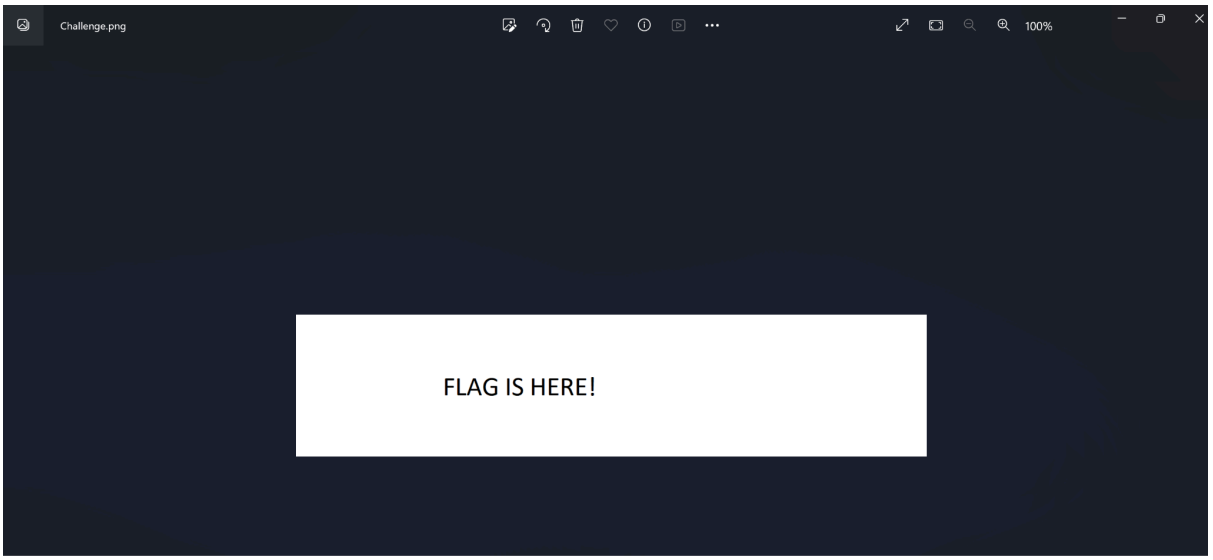
As shown above, I have highlighted the flag in the HTTP Stream.

Flag: GCTF2023{fl4g_in_tcpdump_ez}

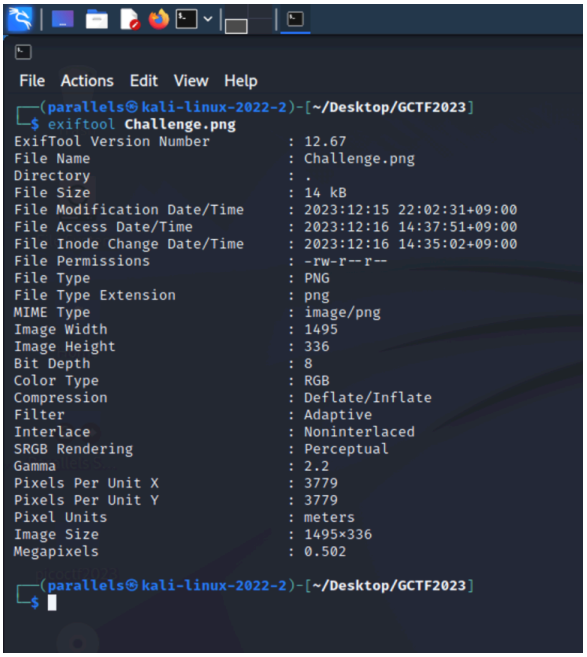
I LOVE PNG!



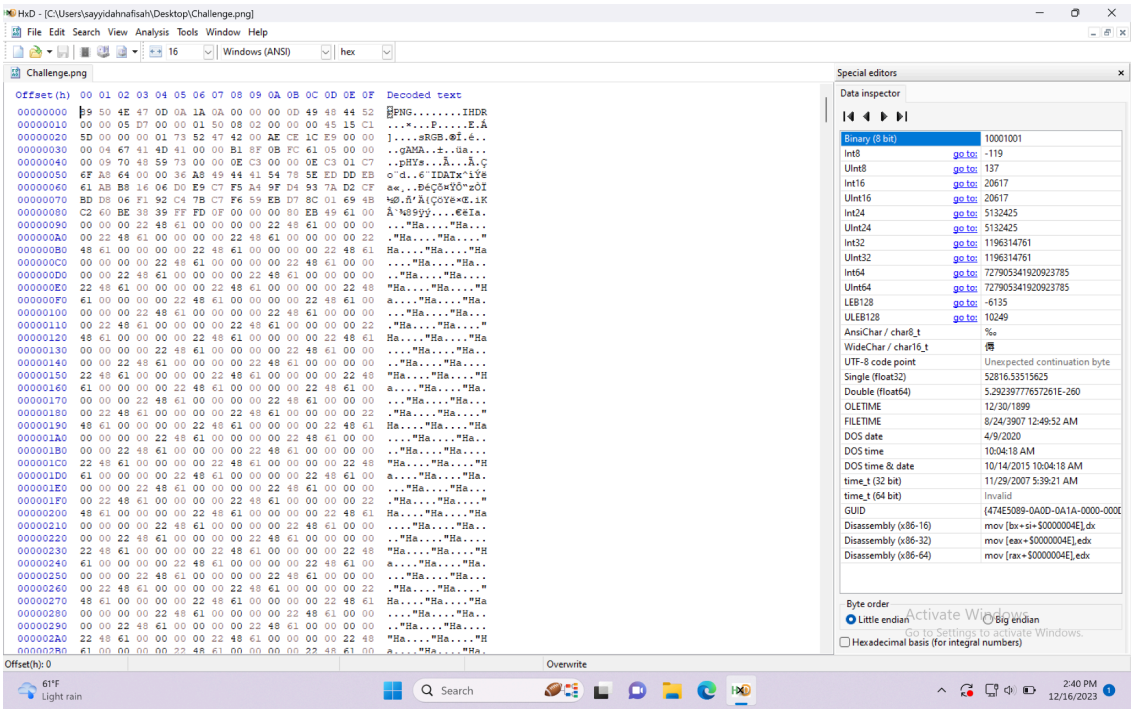
Initially I couldn't open the file in mac os or kali linux, but then I tried to open it in windows vm and it works fine there.



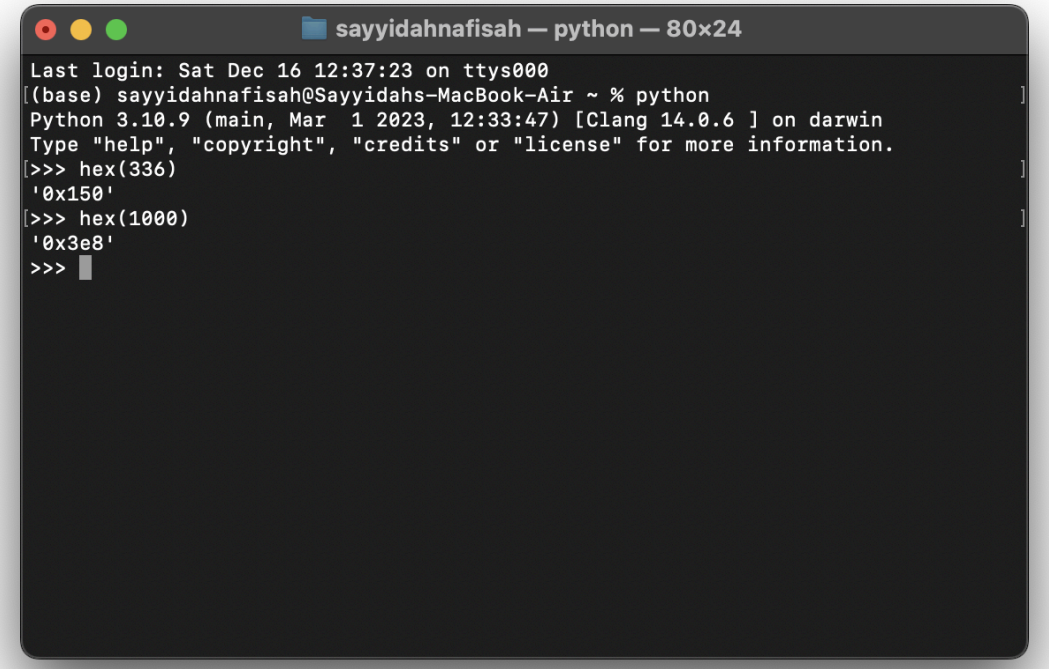
So this is how the image originally looks like. However, this was the only thing on the image and there was no flag. The image also looks like it has been cropped.



Then I use exiftool to see the image info. As we can see here, the height is 336. This info will be used later to edit the image's height.

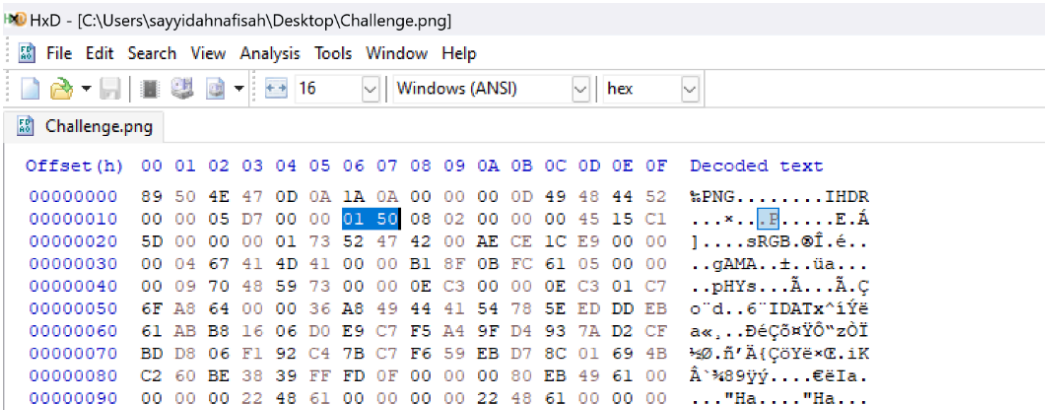


Then, I opened the file using HxD, a hex editor program to edit the hex value of the image. So the numbers here are in hex values.

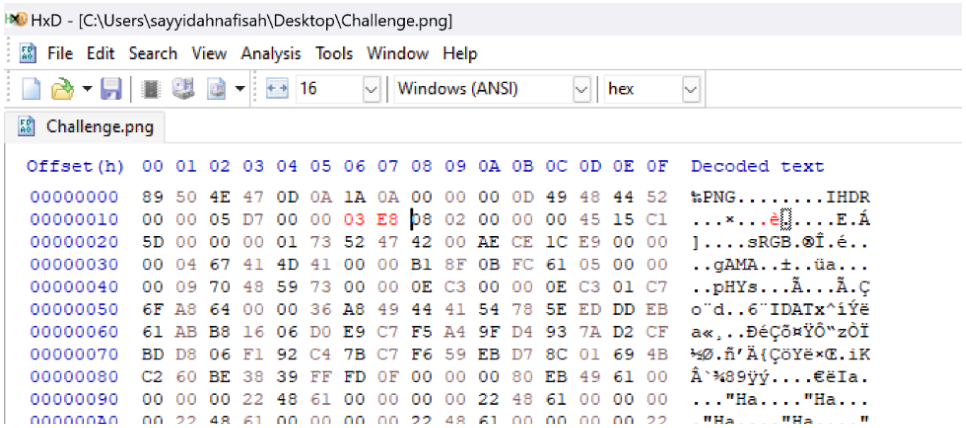


Then, I used python to convert the height that I retrieved from the information earlier into a hex value and the hex value for 336 is 0x150.

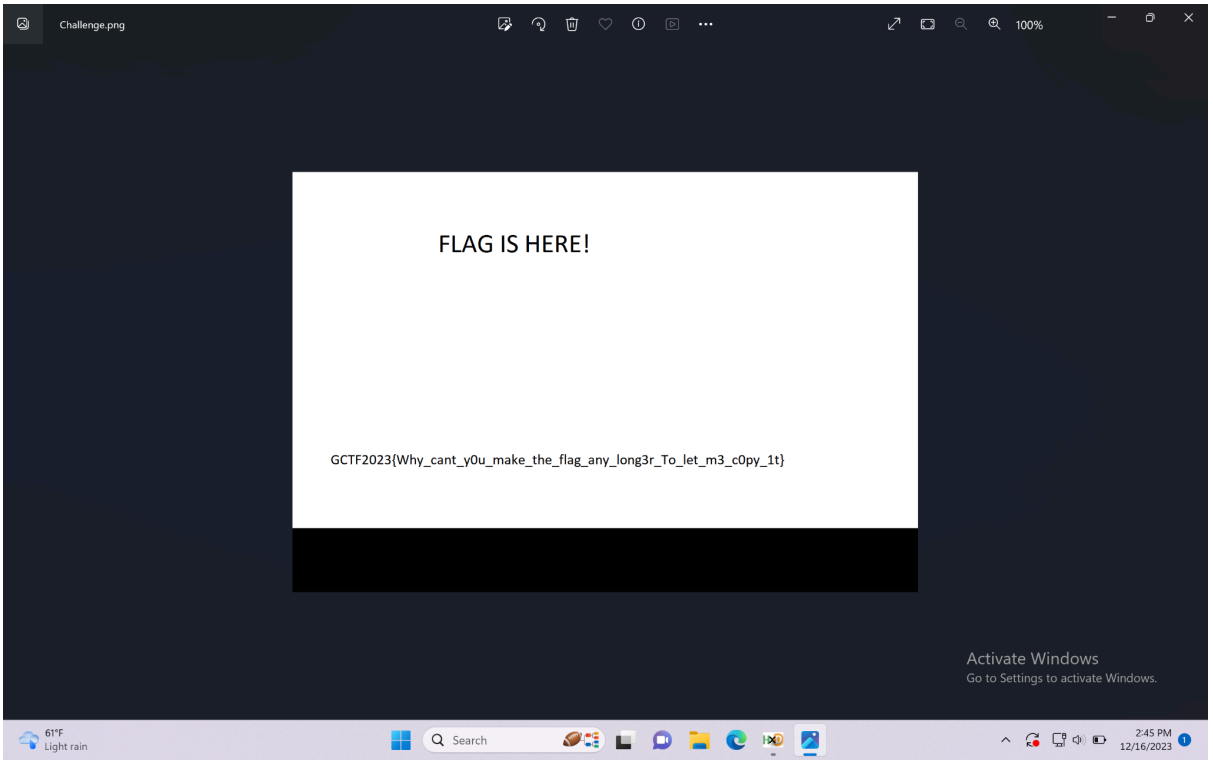
I also converted 1000 to hex value because I'm going to change the image height from 336 to 1000. The new height will be 0x3E8



As I have highlighted, that is the height that we are going to change. To know which value to change, I simply just look for 01 and 50 because the original height is 0x150.



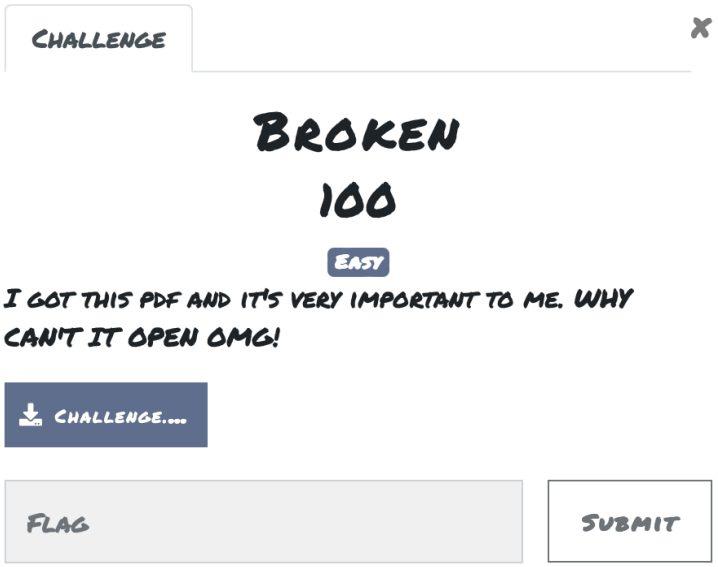
The red hex values are the new height of the image. 03 E8 is the hex value for 1000 which is 0x3E8.

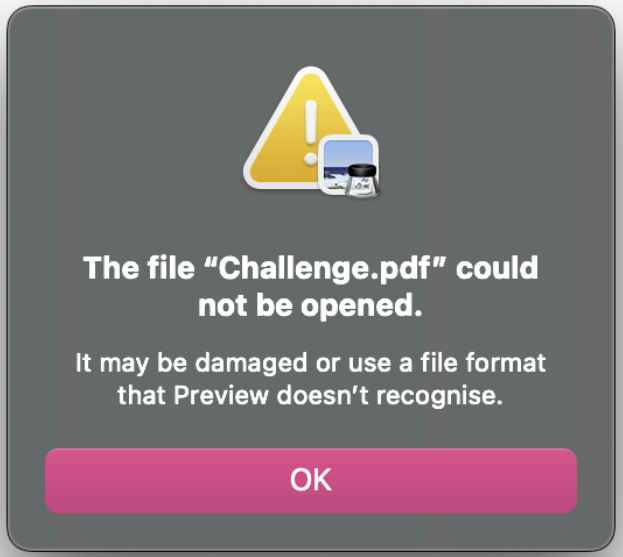


After I edited and saved the new hex value, I reopened the png file and there's the flag!

Flag: GCTF{Why_cant_y0u_make_the_flag_any_long3r_To_let_m3_c0py_1t}

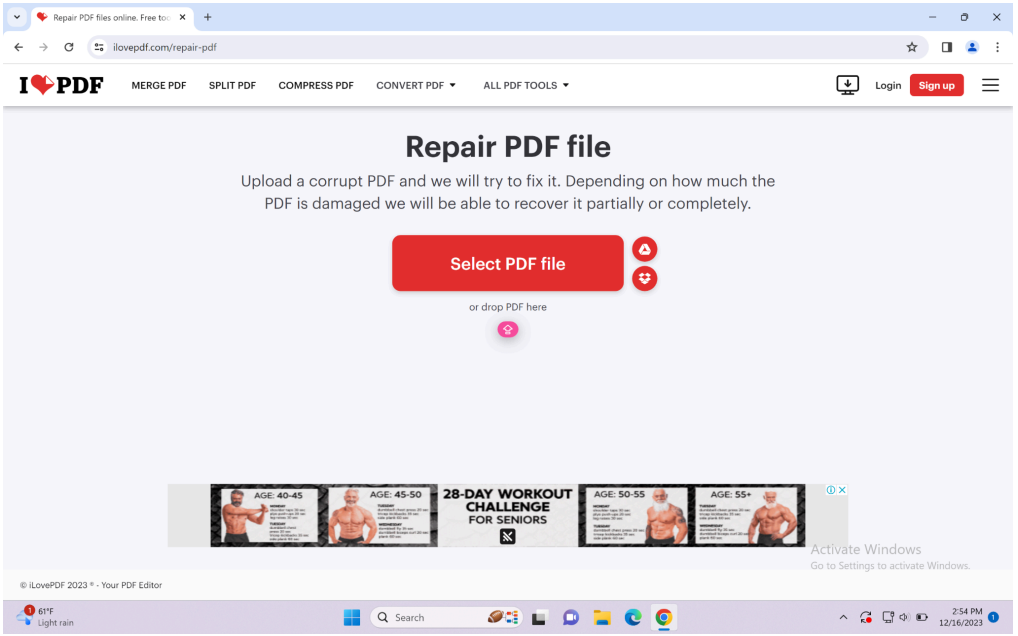
BROKEN



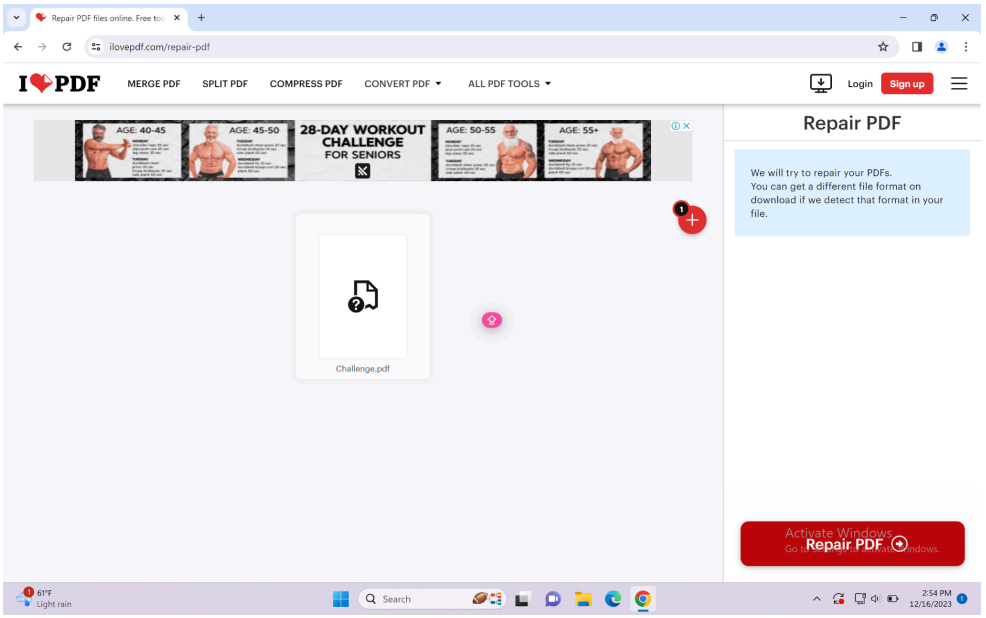


So we were given a broken pdf file. Then I googled and found a website to repair pdf files. The website link is as below:

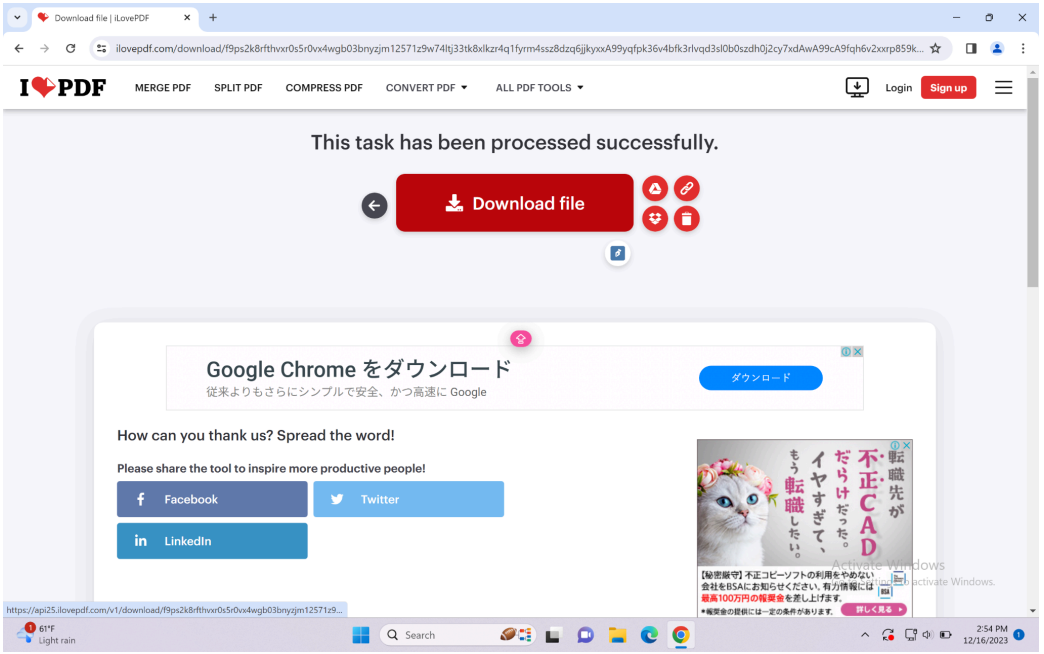
https://www.ilovepdf.com/repair-pdf



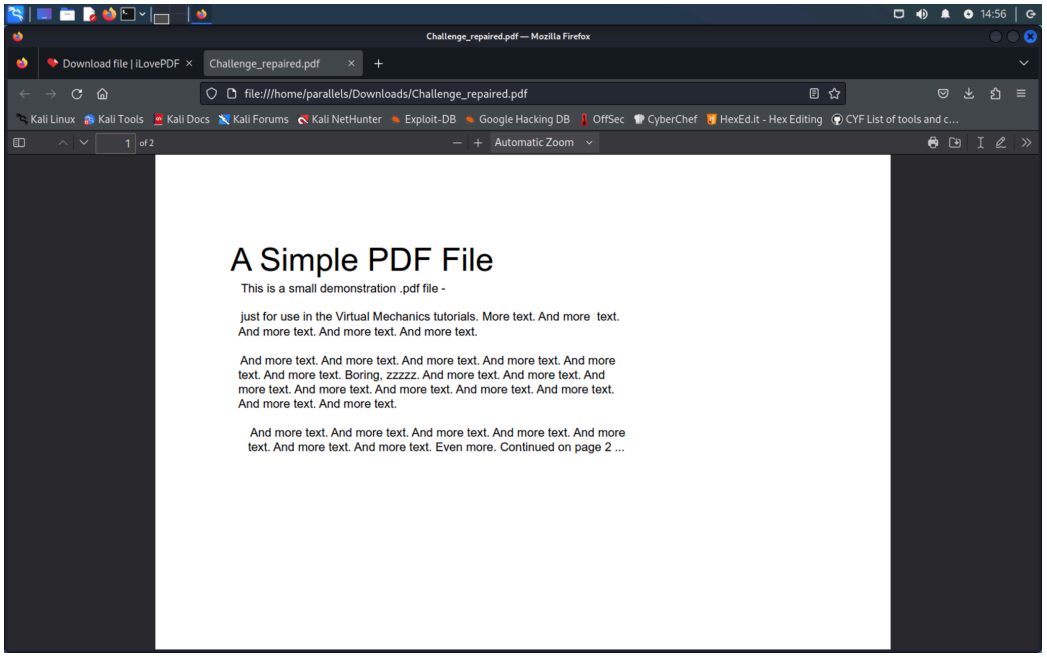
I clicked “select pdf file” to upload a broken pdf file.

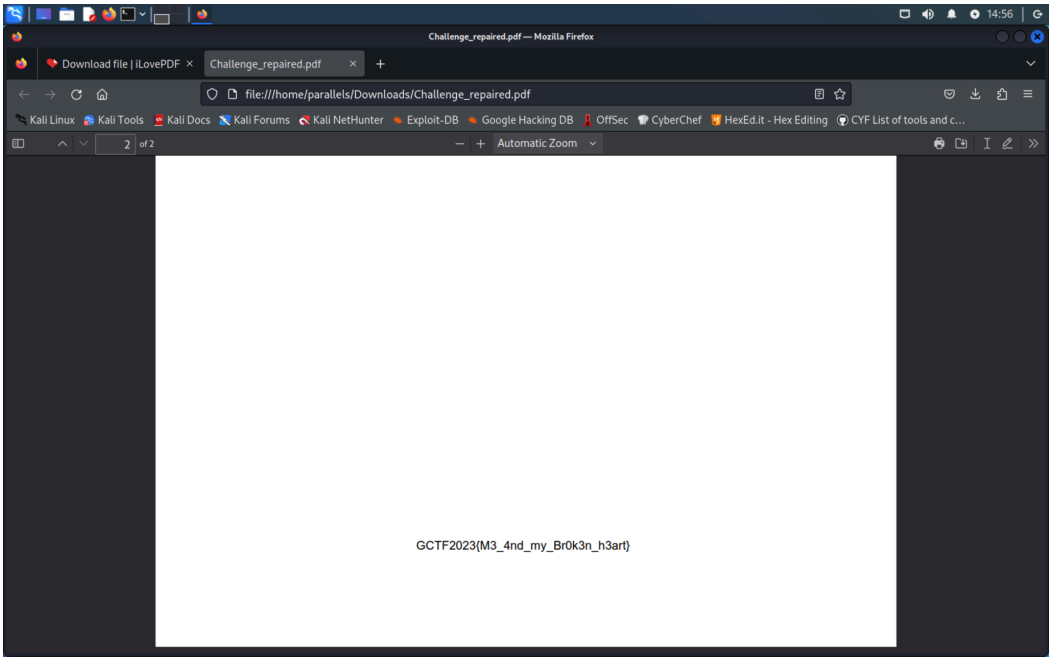


I uploaded Challenge.pdf



After that, I downloaded the repaired file and found the pdf files as below.

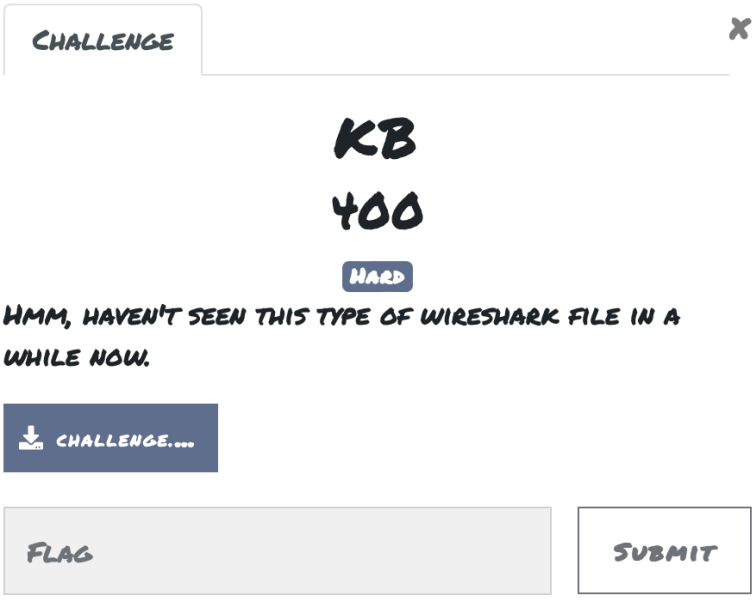


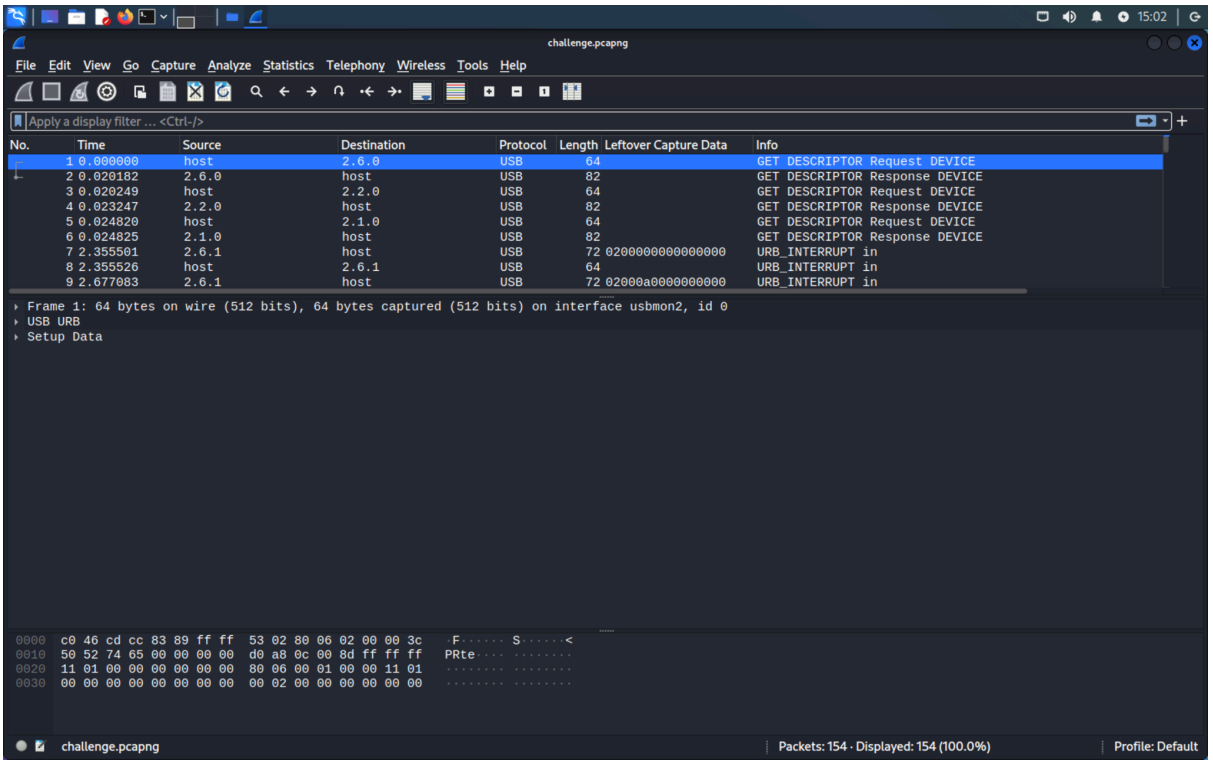


Found the flag!

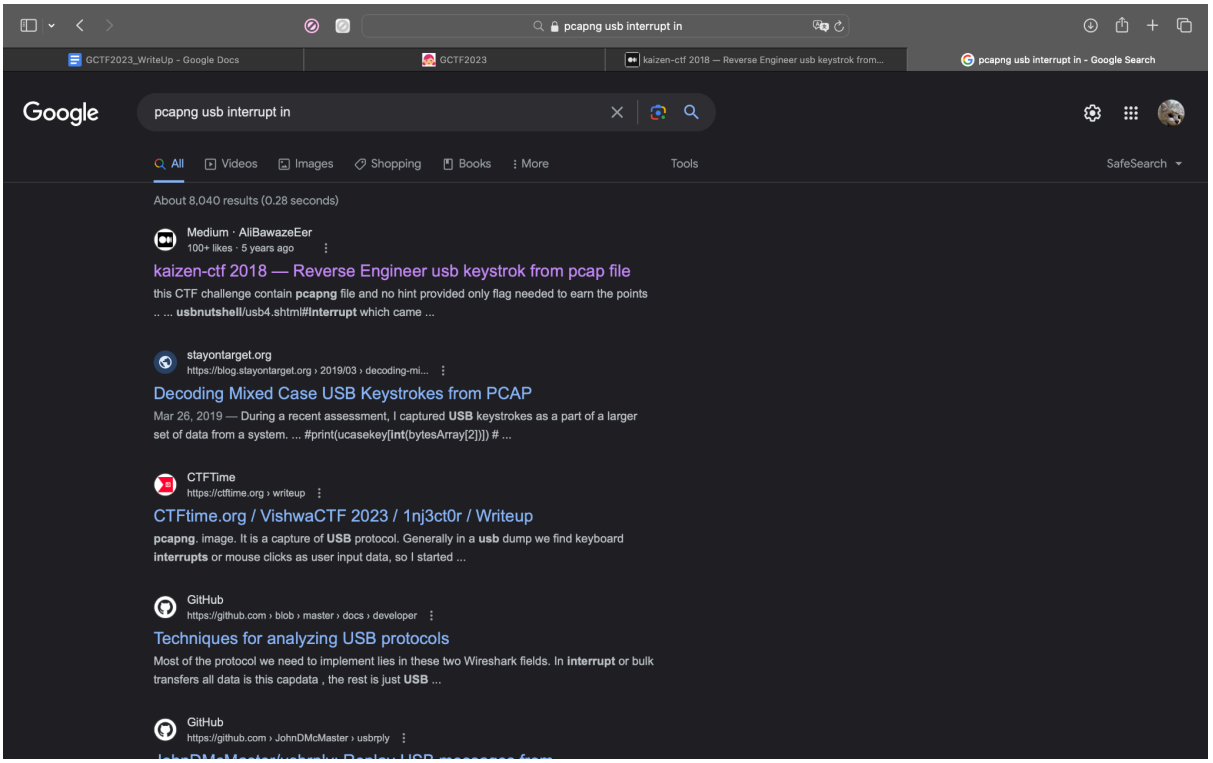
Flag: GCTF2023{M3_4nd_my_Br0k3n_h3art}

KB



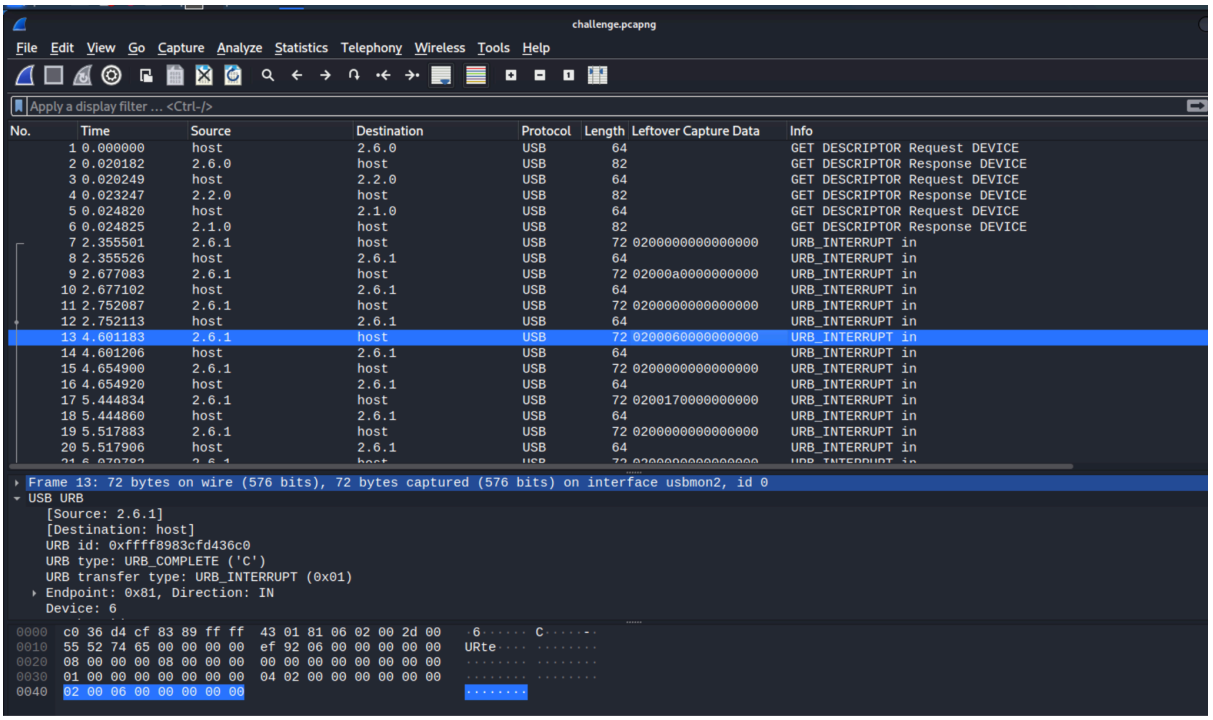


For this challenge,I opened the file using wireshark, and this is what it shows.

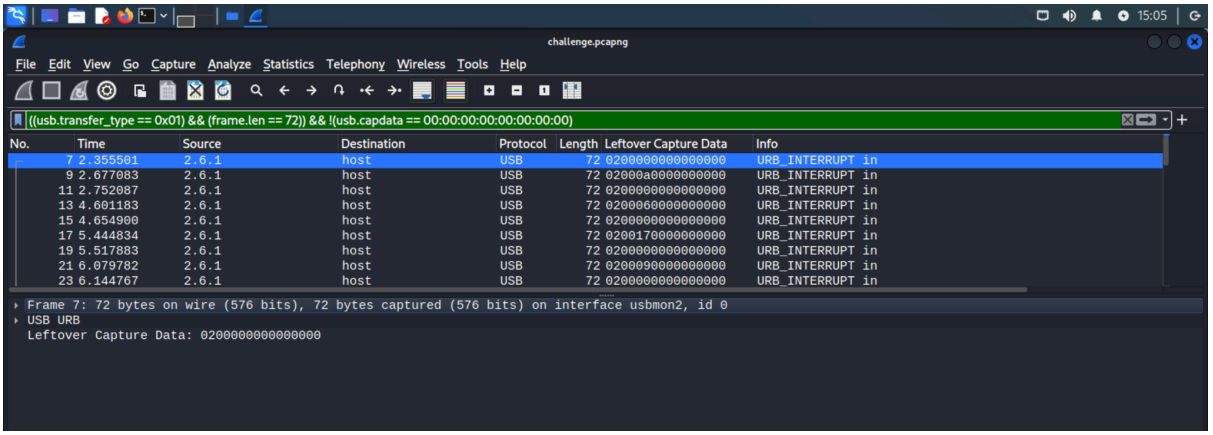


Then, I just googled some keywords and I found this website (link below) and I referred to it for this challenge.

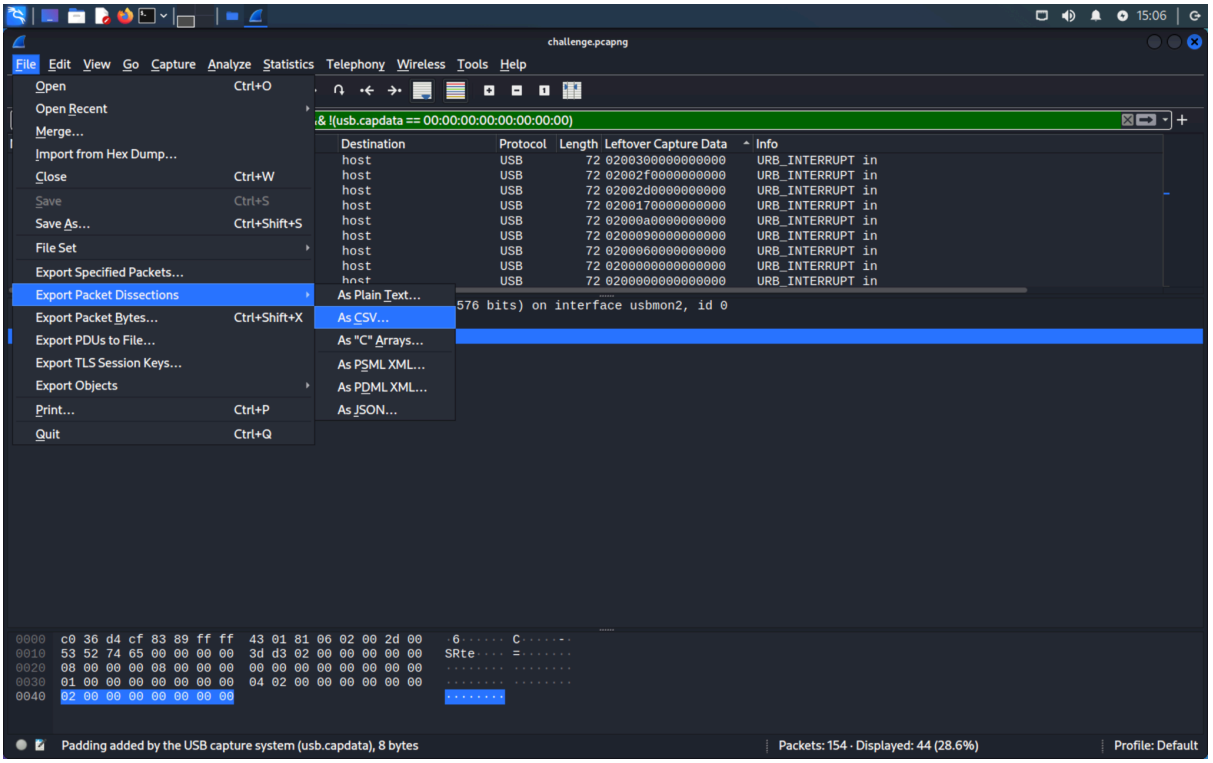
https://abawazeeer.medium.com/kaizen-ctf-2018-reverse-engineer-usb-keystrok-from-pcap-file-2412351679f4



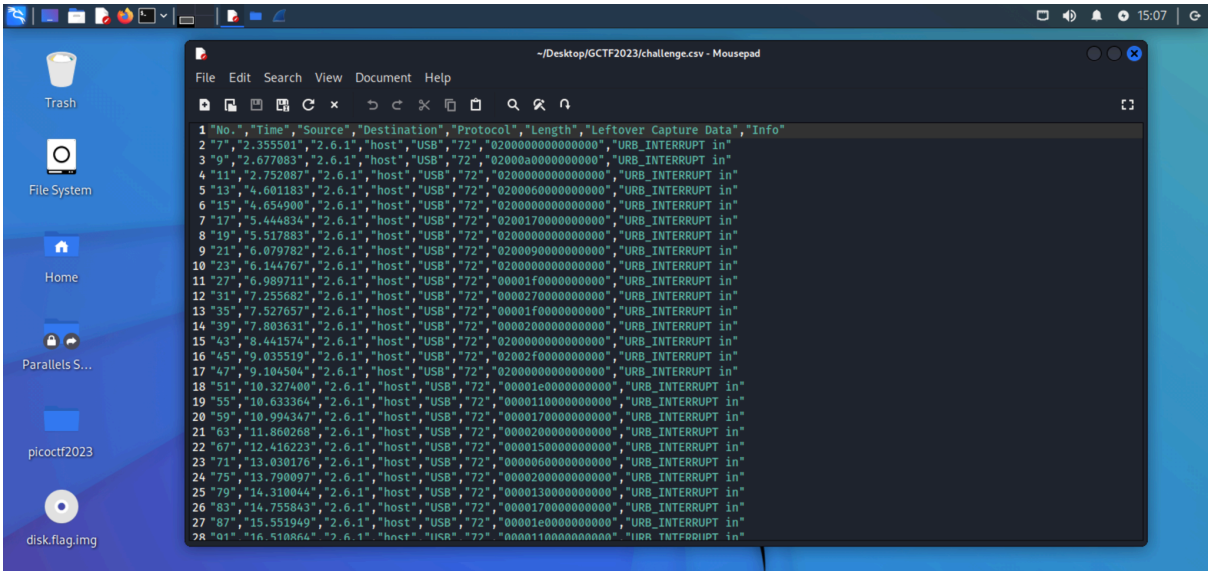
So, as we can see, there is some pattern in Leftover Capture Data, where a certain number changed on that position only, and it occurs on USB transfer type URB_INTERRUPT with frame length 72 only.



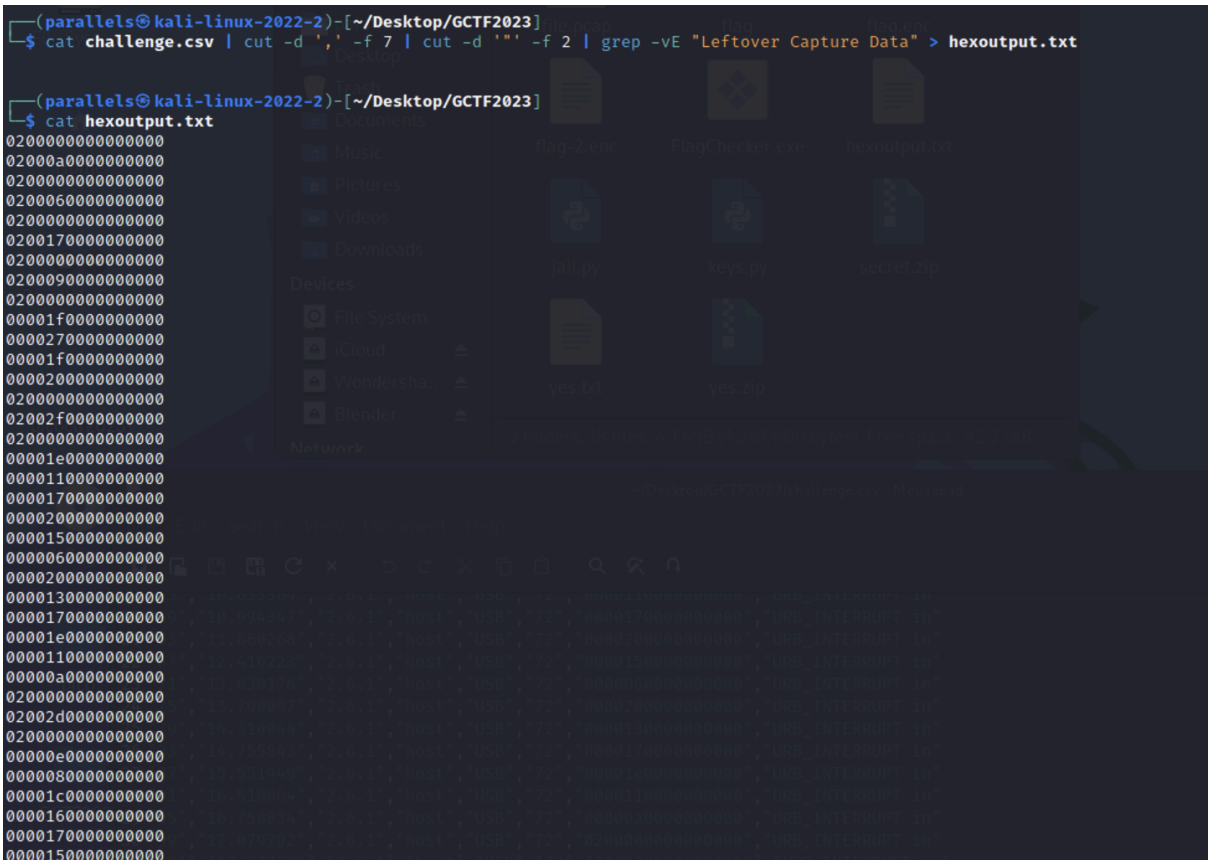
So, we filtered the data using this logic where usb transfer type is 0x01, and frame length 72: **((usb.transfer_type == 0x01) && (frame.len == 72)) && !(usb.capdata == 00:00:00:00:00:00:00:00)**



Then, I exported the filtered data into a csv file and included the Leftover Capture Data into the column because that is the data that we need.



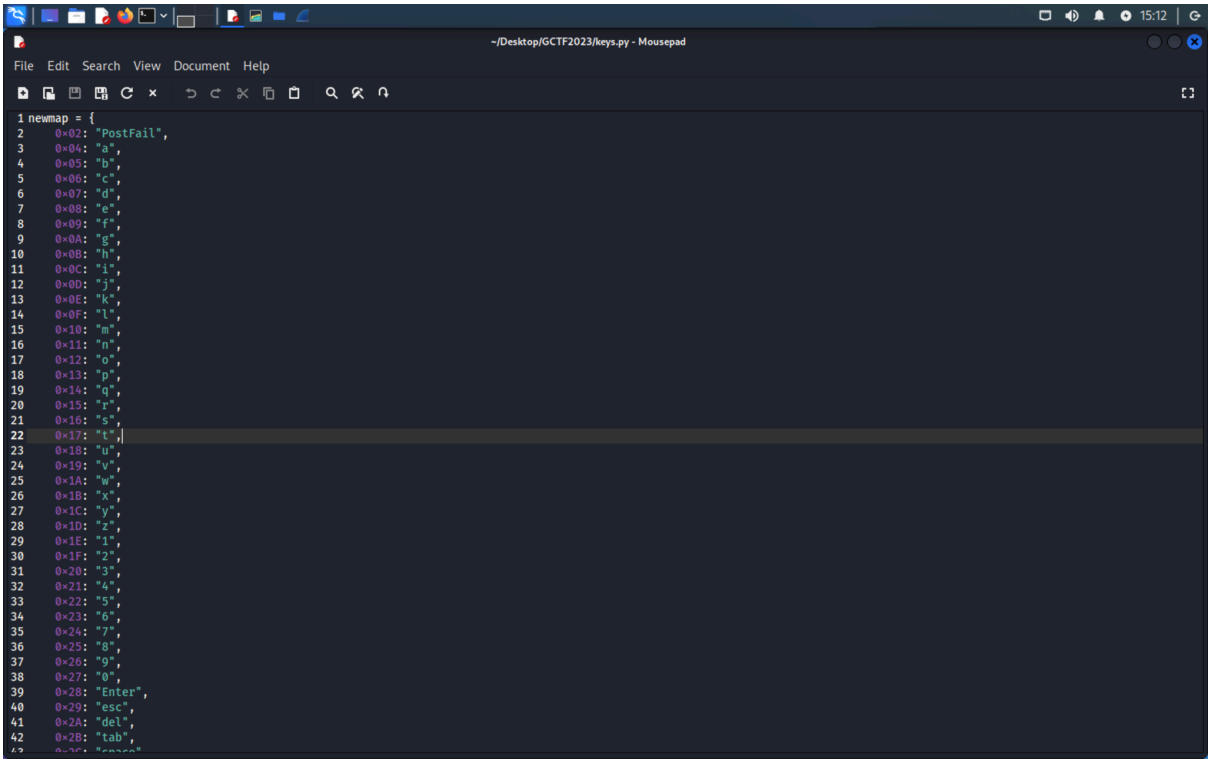
This is how the data looks like in the csv file.

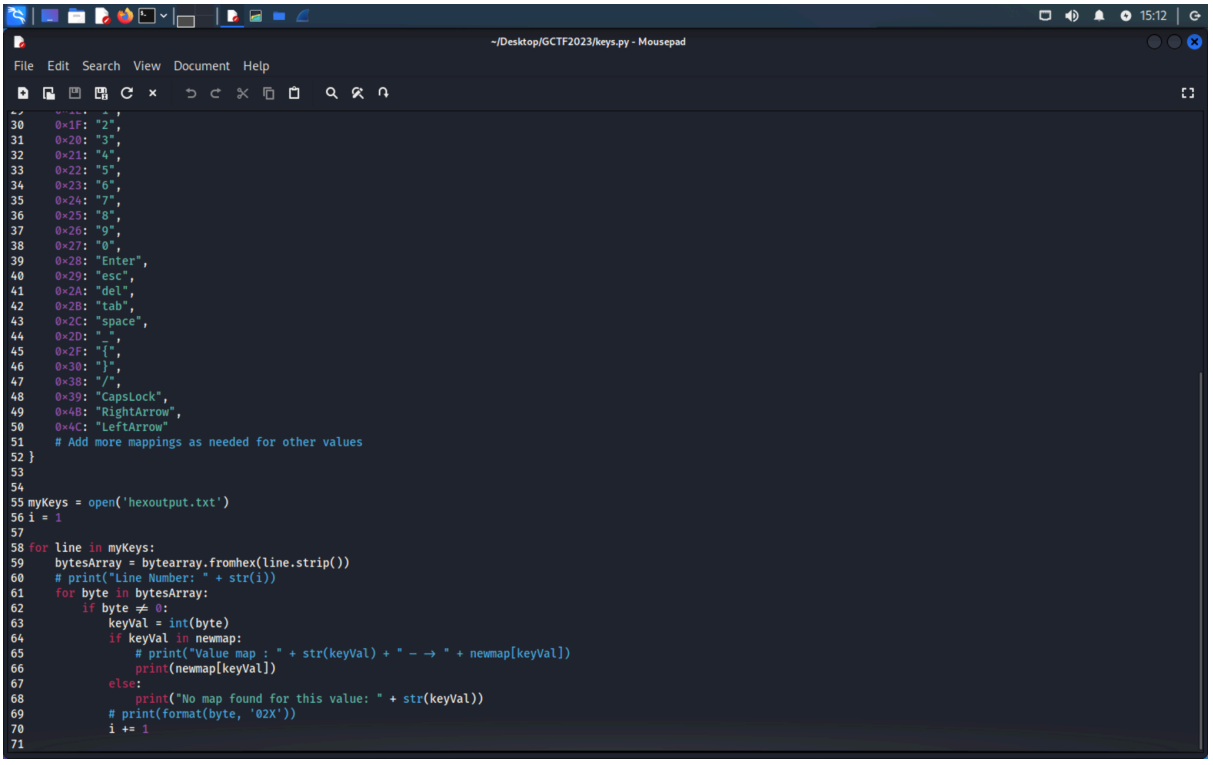


To save the Leftover Capture Data only, into an another txt file (hexoutput.txt) automatically, I use this command:

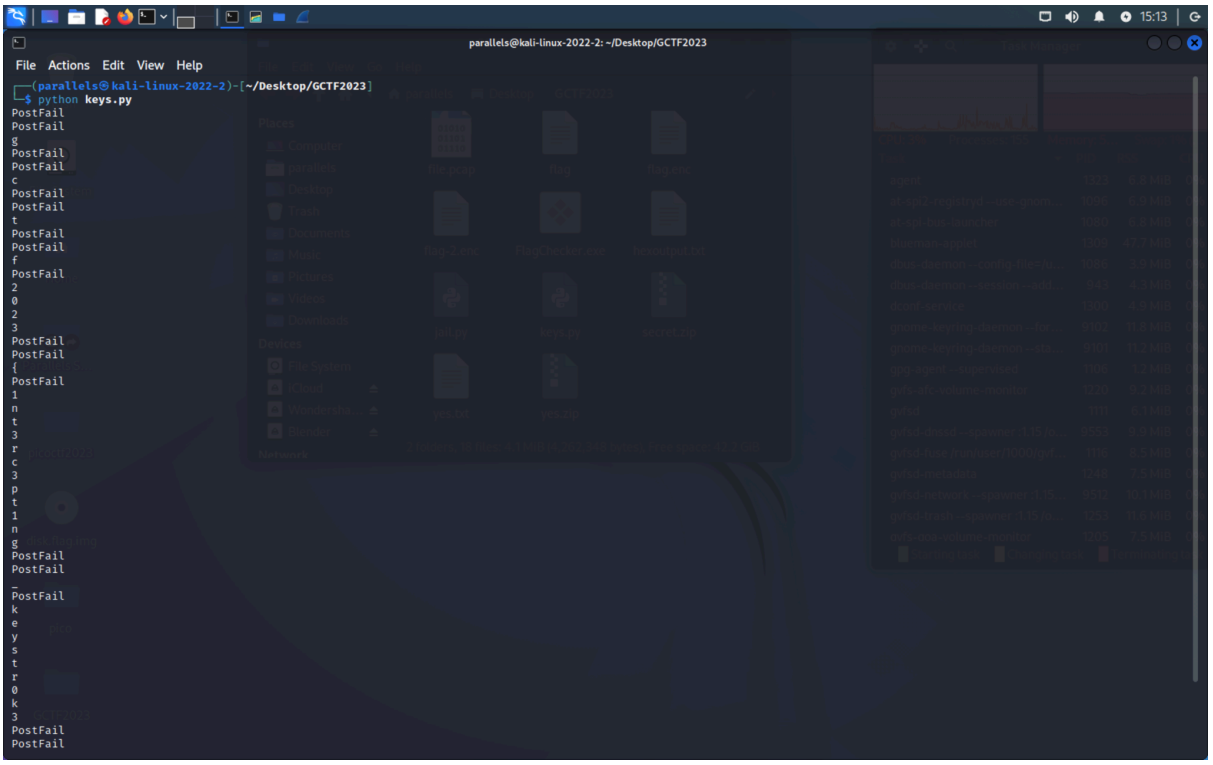
cat challenge.csv | cut -d ',' -f 7 | cut -d '"' -f 2 | grep -vE "Leftover Capture Data" > hexoutput.txt

Then, I read the hexoutput.txt file by using "cat hexoutput.txt" to make sure that everything is written there correctly.





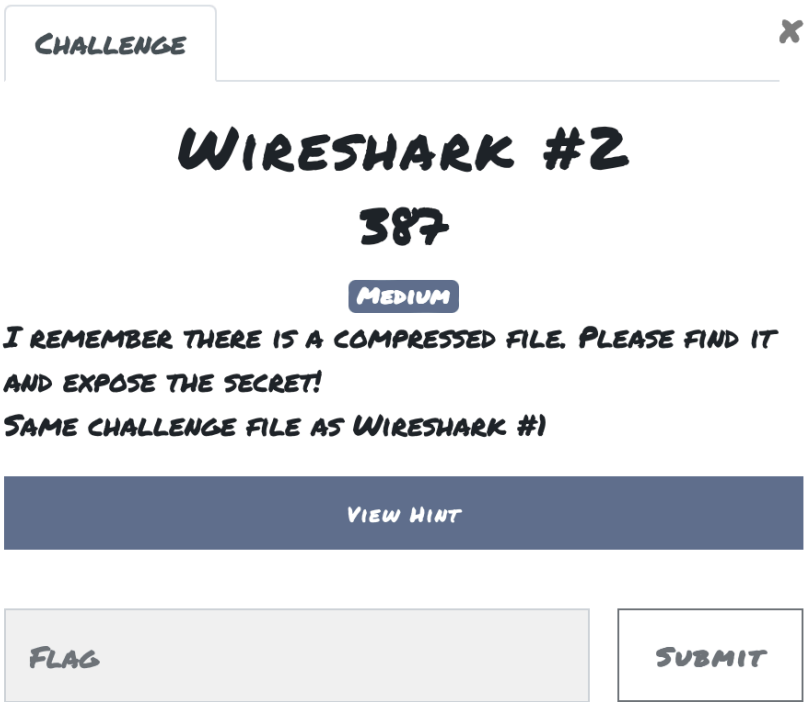
This is the python code that I used to convert the hexoutput.txt into a readable or ASCII character.

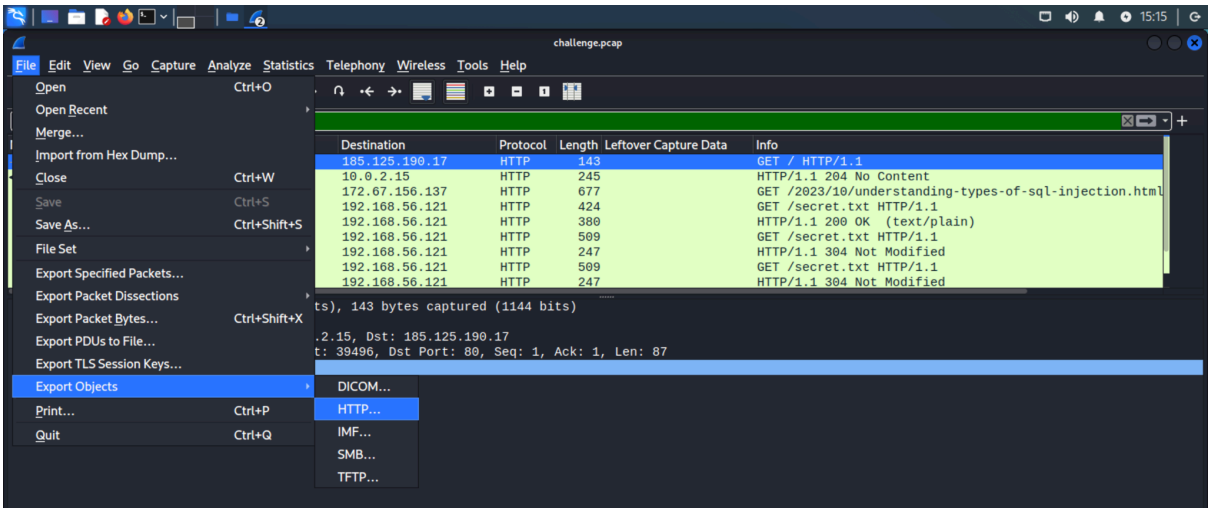


This is the output from hexouput.txt after I run the python code.
After I type all the characters manually, I got the flag:

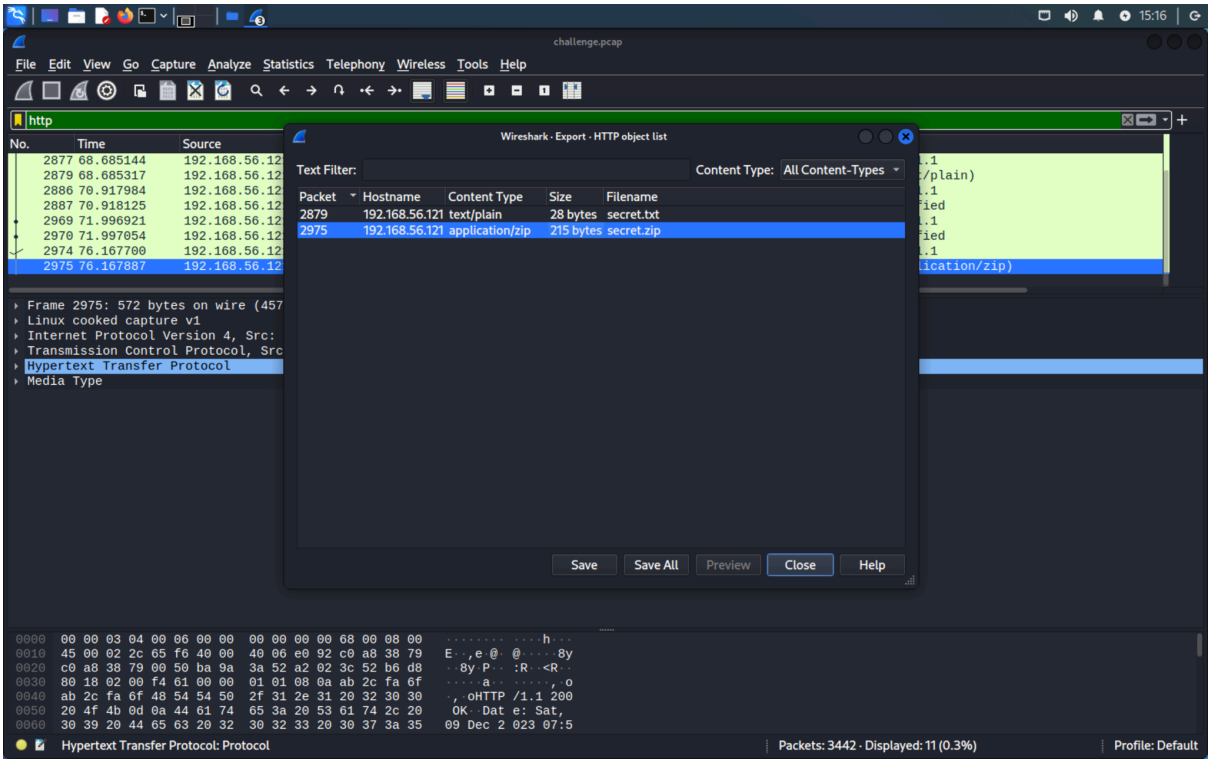
GCTF2023{1nt3c3pt1ng_keystr0k3}

WIRESHARK #2

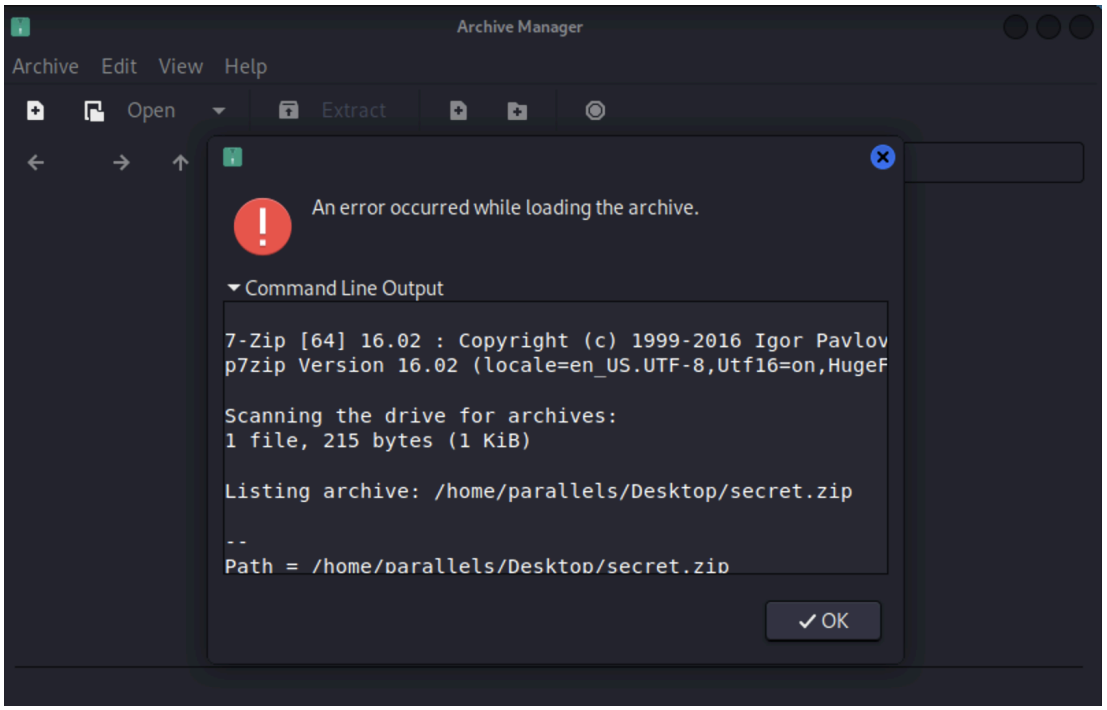




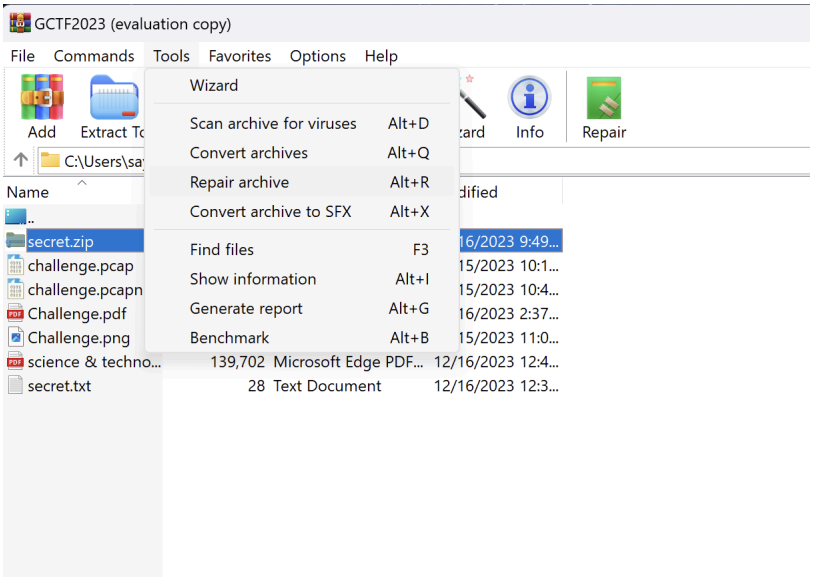
As this is a continuation from challenge Wireshark #1, I am using the same pcap file, then I clicked export objects and clicked http to export the secret.zip file.



As shown above, I exported the file.



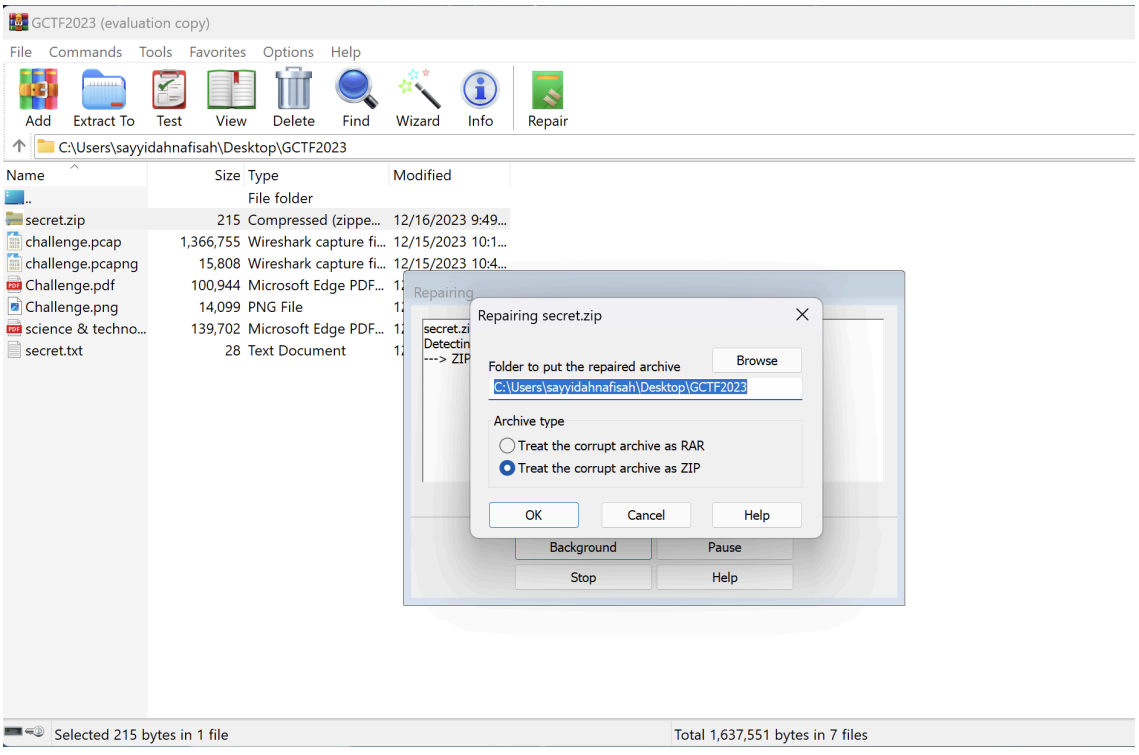
However, the zip file is corrupted and I couldn't opened the file.



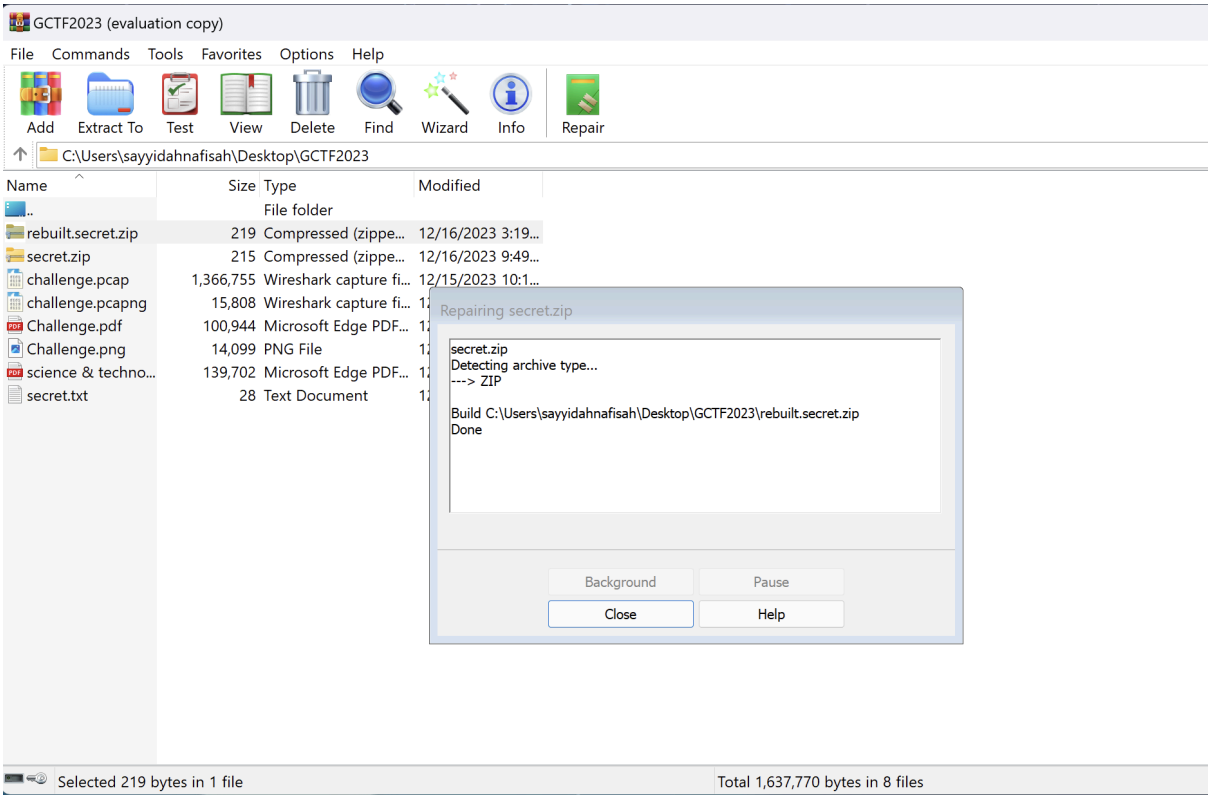
Then I used Win-rar apps to fix the zip file.

The steps to fix the file:

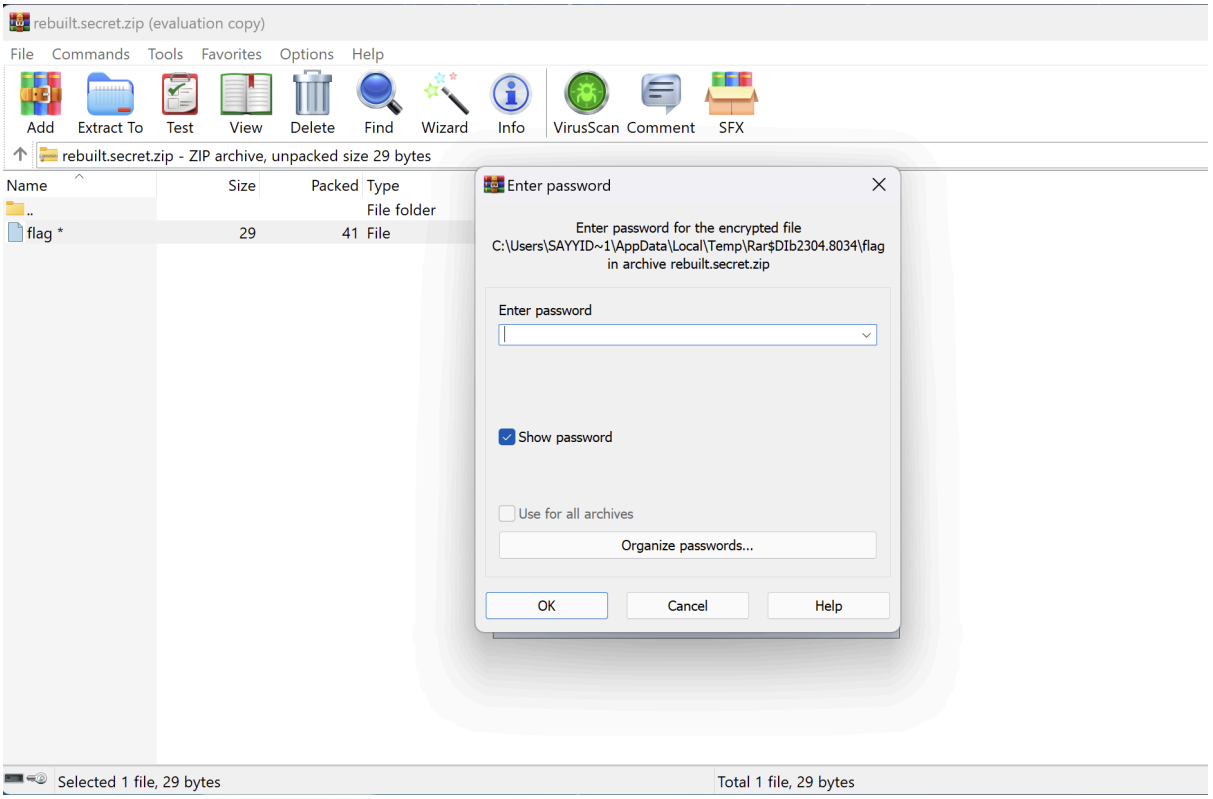
1. select secret.zip file
2. click Repair archive



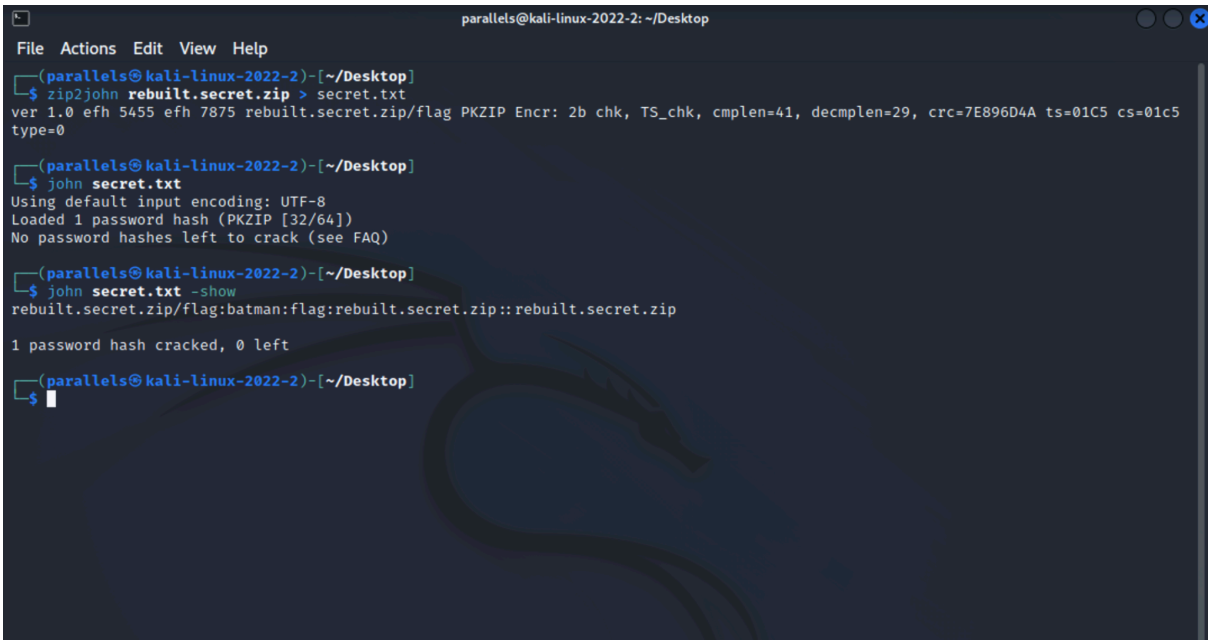
3. Click OK



4. Click close



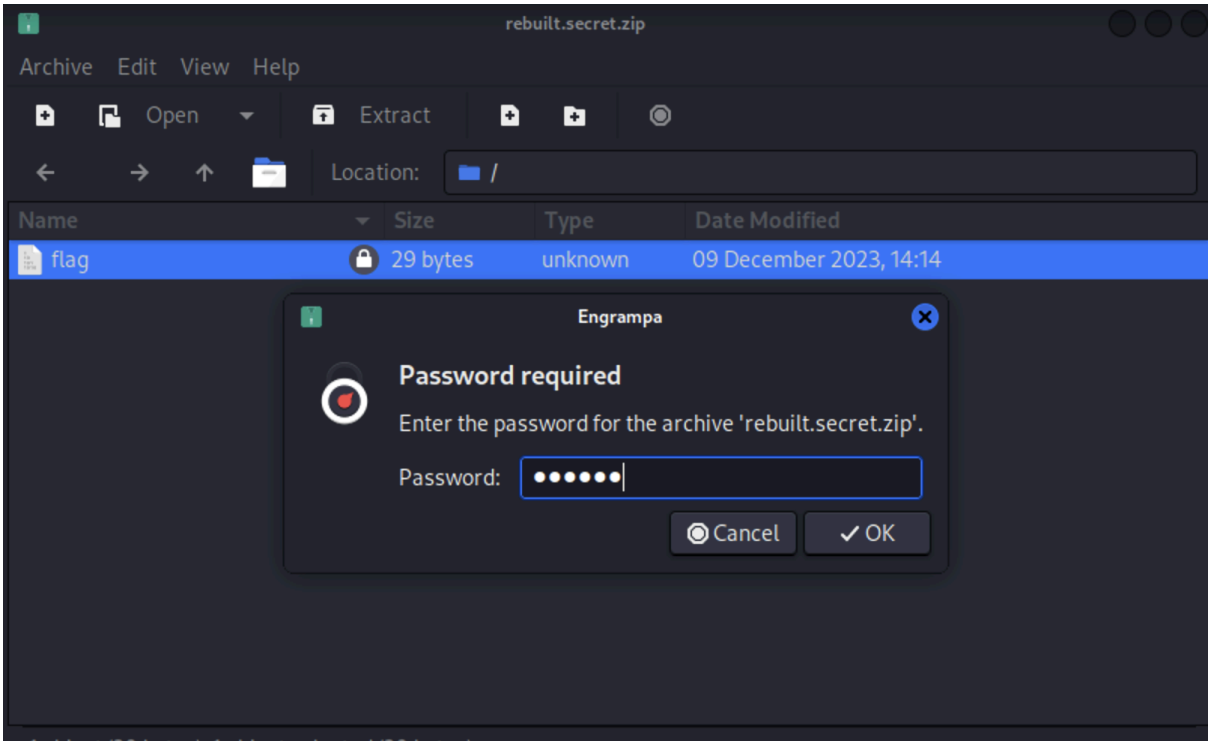
5. The file is successfully repaired but we are still not done yet because the file is protected by password.



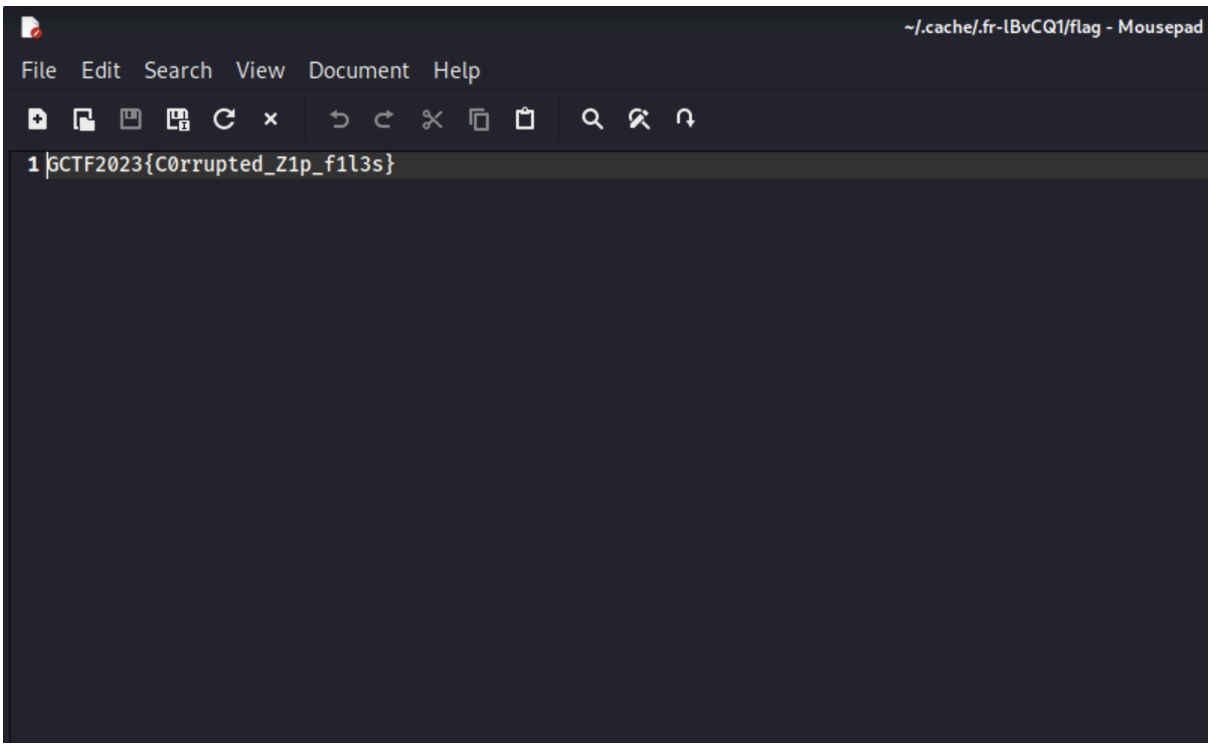
To crack the password, I referred to this video:
https://www.youtube.com/watch?v=yyIoX0QT6QM
I cracked it using John the ripper.

The linux command is as follows:
zip2john rebuild.secret.zip > secret.txt
john secret.txt
john secret.txt -show

Then, as shown in the picture, the password for the file is batman.



I entered the password "batman"

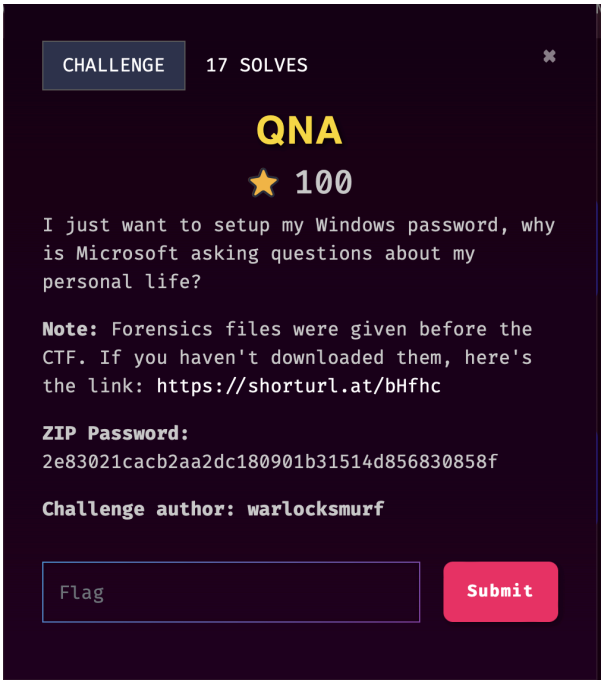


I successfully unlocked the file and found the flag!

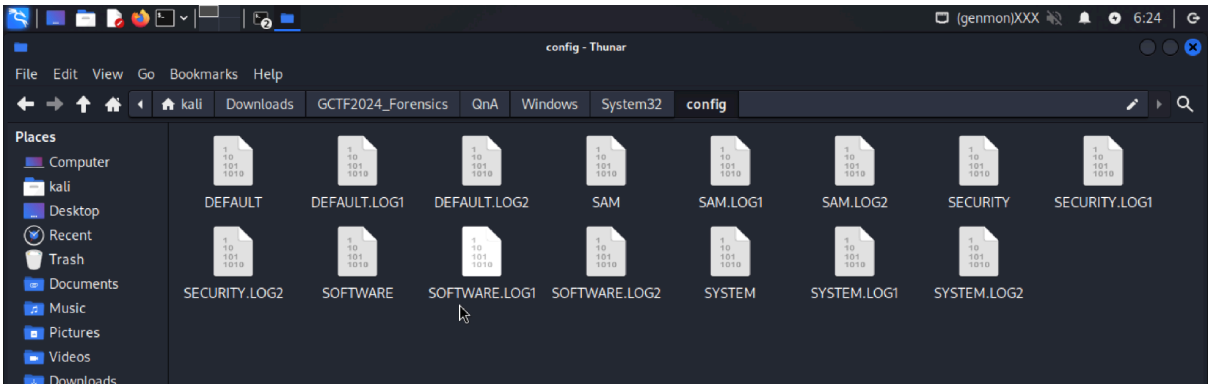
Flag: GCTF2023{C0rrupted_Z1p_f1l3s}

FORENSIC

QnA



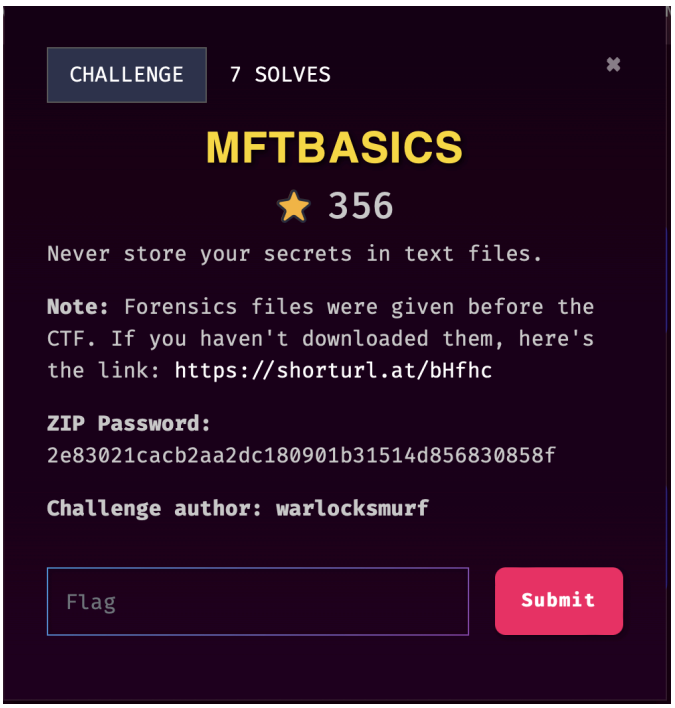
In this directory, I just look for the User's name which is SAM.



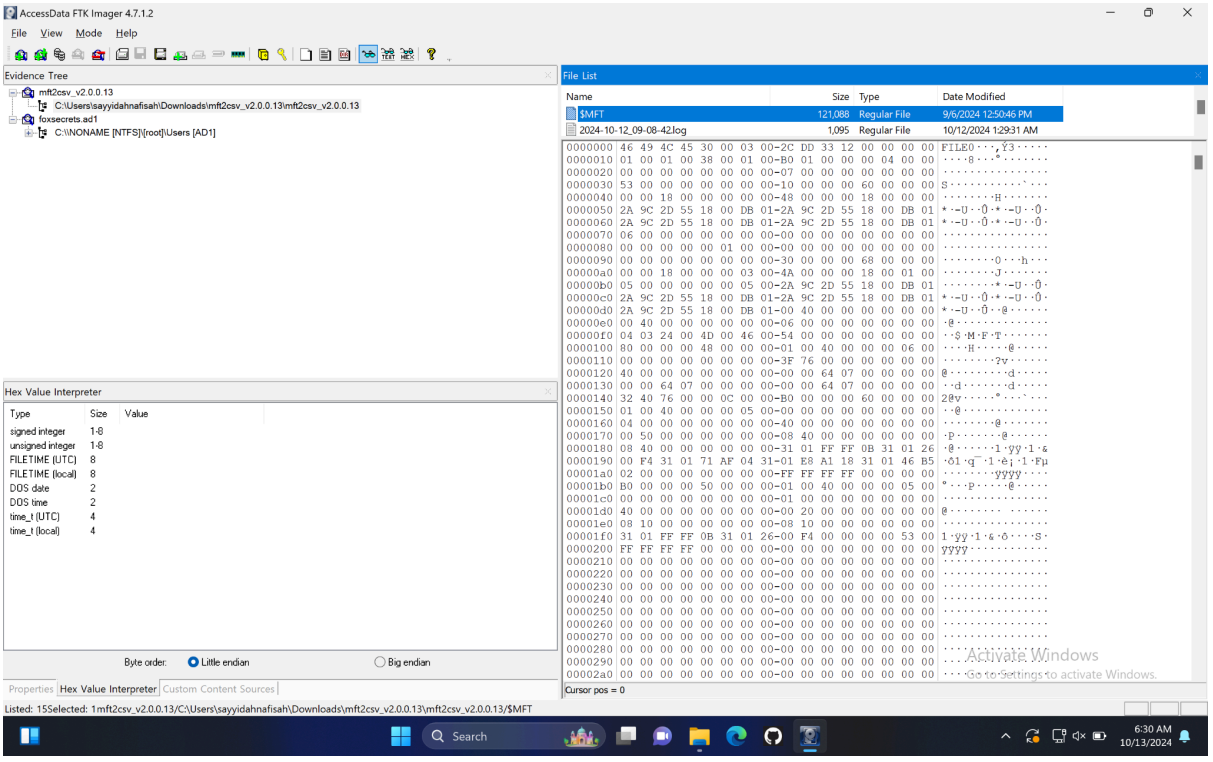
I used the command cat SAM to see the content and finally found the flag which was divided into 3 parts of the questions.

Flag: GCTF{p3rs0nal_s3cr3ts_r3g1stry}

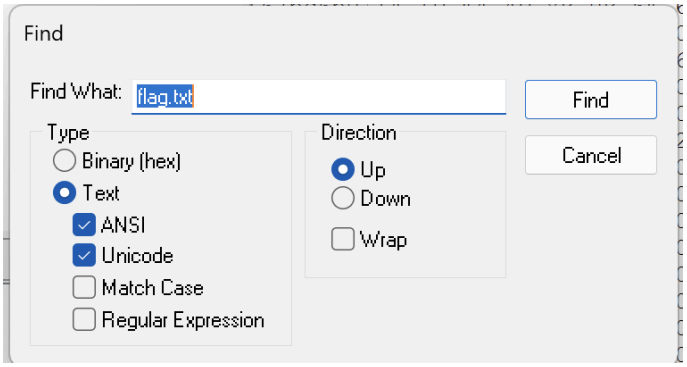
MFTBasics



To perform analysis on the $MFT file, I put it in a folder and opened it using FTK Imager.



Right click and click find to find a keyword for the secret text file which is flag.txt. Just find it until I find the secret code.

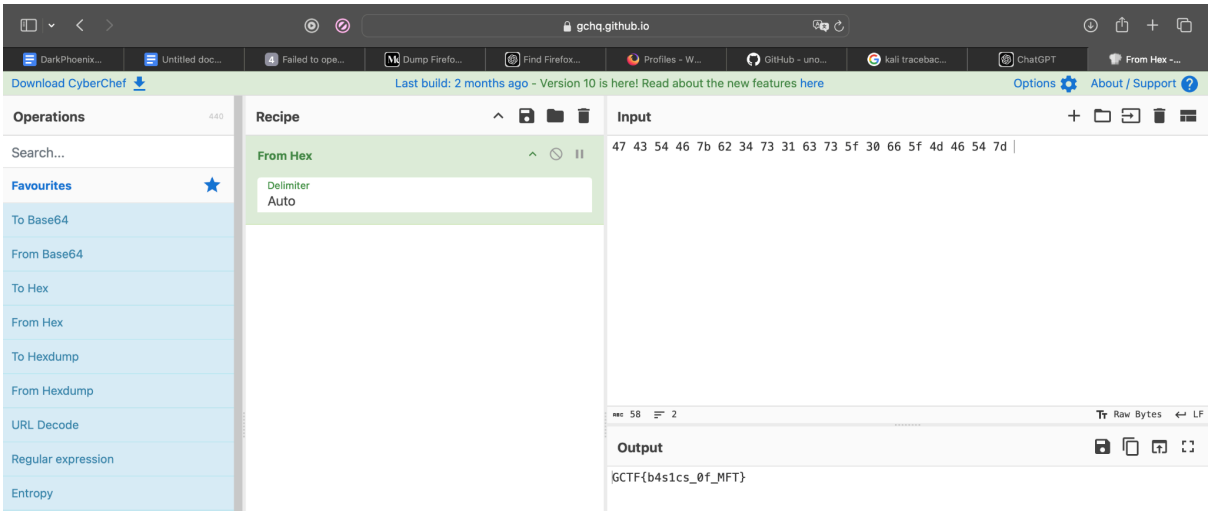


Finally found the secret code:

The secret code is: 47 43 54 46 7b 62 34 73 31 63 73 5f 30 66 5f 4d 46 54 7d 4d

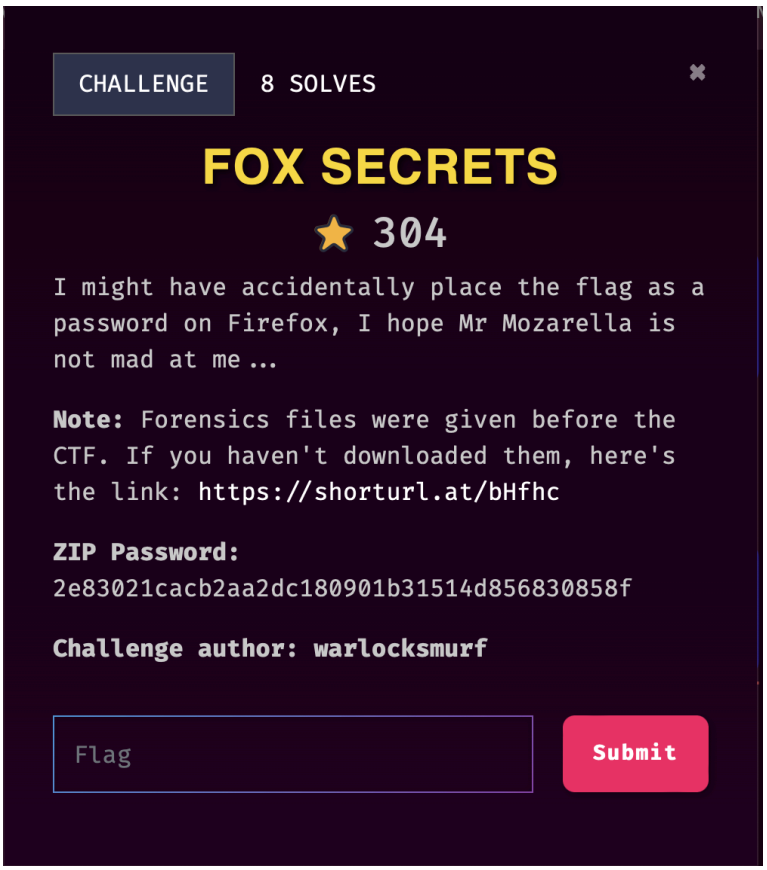
```
6fadcf0 08 03 66 00 6C 00 61 00-67 00 2E 00 74 00 78 00 ··f·l·a·g·.·t·x·
6fadd00 74 00 00 00 00 00 00 00-40 00 00 00 28 00 00 00 t······@···(···
6fadd10 00 00 00 00 00 00 05 00-10 00 00 00 18 00 00 00 ················
6fadd20 87 8D 61 04 88 6B EF 11-AB D7 00 0C 29 22 8E 27 ··a··kï·«×··)"·'
6fadd30 80 00 00 00 68 00 00 00-00 00 18 00 00 00 01 00 ····h···········
6fadd40 4C 00 00 00 18 00 00 00-54 68 65 20 73 65 63 72 L·······The secr
6fadd50 65 74 20 63 6F 64 65 20-69 73 3A 20 34 37 20 34 et code is: 47 4
6fadd60 33 20 35 34 20 34 36 20-37 62 20 36 32 20 33 34 3 54 46 7b 62 34
6fadd70 20 37 33 20 33 31 20 36-33 20 37 33 20 35 66 20  73 31 63 73 5f
6fadd80 33 30 20 36 36 20 35 66-20 34 64 20 34 36 20 35 30 66 5f 4d 46 5
6fadd90 34 20 37 64 34 64 20 34-FF FF FF FF 82 79 47 11 4 7d4d 4ÿÿÿÿ·yG·
6fadda0 FF FF FF FF 82 79 47 11-00 00 00 00 18 00 00 00 ÿÿÿÿ·yG·····
6faddb0 FF FF FF FF 82 79 47 11-00 00 00 00 00 00 00 00 ÿÿÿÿ·yG·····
6faddc0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
```

To decrypt that i just threw it on cyberchef and finally cracked it.



Flag: GCTF{b4s1cs_0f_MFT}

Fox Secrets

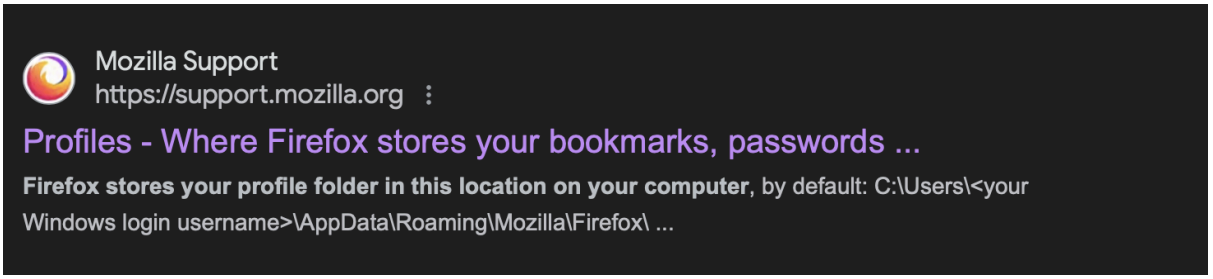


We were given an image file foxsecrets.ad1. To do an analysis on the image file, I opened it using FTK imager.

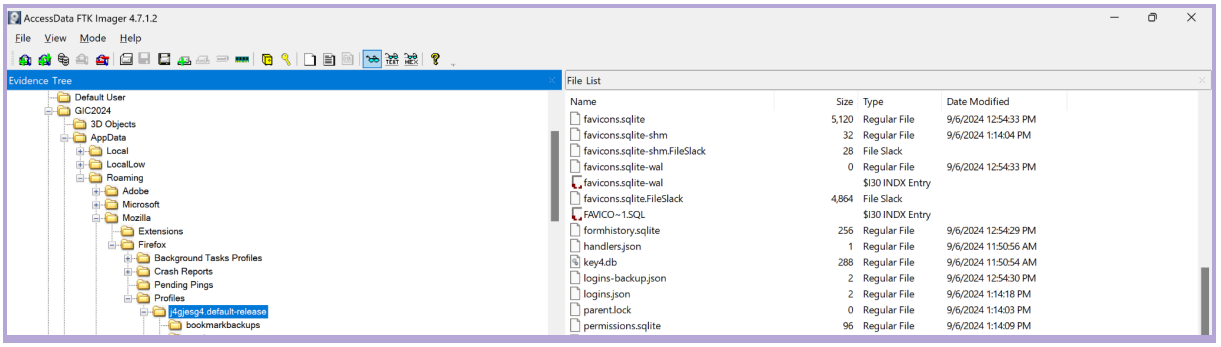
Since the clue said they stored the password on firefox, I must look for `key4.db` and `logins.json` according to https://support.mozilla.org/en-US/kb/profiles-where-firefox-stores-user-data,

So I looked for the path of the stored password on mozilla which is:

`username\AppData\Roaming\Mozilla\Firefox\Profiles`



So to look for the files, I just opened the directories on FTK imager GIC2024\AppData\Roaming\Mozilla\Firefox\Profiles\j4gjesg4.default-release and as shown below, there are key4.db and logins.json in that folder.

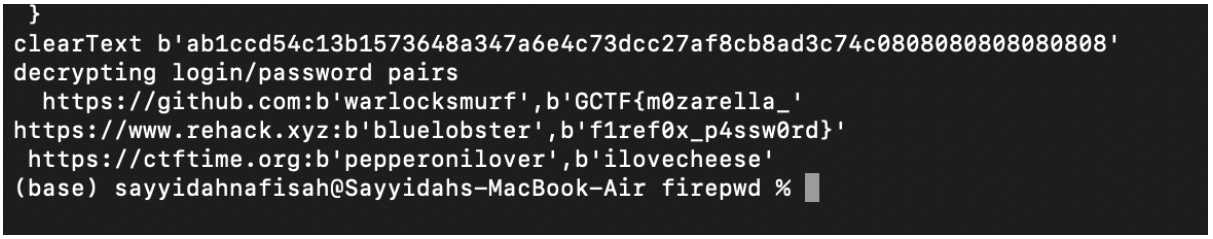


To retrieve the password, I used this website as my reference: https://medium.com/@s12deff/dump-firefox-passwords-with-firepwd-and-firefox-decrypt-65350fd74503

Since we already have key4.db and logins.json, the step to find the passwords are

1. git clone https://github.com/lclevy/firepwd
2. Put the key4.db and logins.json in the same folder as firepwd
3. python3 firepwd.py

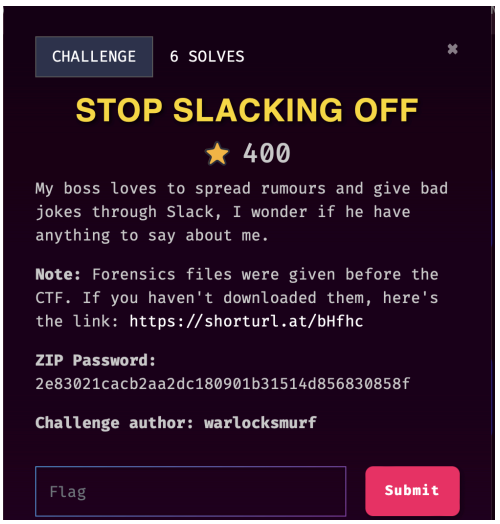
```
zsh: no such file or directory: https://github.com/lclevy/firepwd
(base) sayyidahnafisah@Sayyidahs-MacBook-Air fox secret % git clone https://github.com/lclevy/firepwd

Cloning into 'firepwd'...
remote: Enumerating objects: 88, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 88 (delta 2), reused 3 (delta 0), pack-reused 80 (from 1)
Receiving objects: 100% (88/88), 239.08 KiB | 2.28 MiB/s, done.
Resolving deltas: 100% (41/41), done.
(base) sayyidahnafisah@Sayyidahs-MacBook-Air fox secret % ls
firepwd         key4.db         logins.json
(base) sayyidahnafisah@Sayyidahs-MacBook-Air fox secret % cd firepwd
(base) sayyidahnafisah@Sayyidahs-MacBook-Air firepwd % ls
LICENSE                 logins.json             mozilla_pbe.svg
firepwd.py              mozilla_db              readme.md
key4.db                 mozilla_pbe.pdf         requirements.txt
(base) sayyidahnafisah@Sayyidahs-MacBook-Air firepwd % python3 firepwd
python3: can't open file '/Users/sayyidahnafisah/Desktop/shared/fox secret/firepwd/firepwd': [Errno 2] No such file or directory
(base) sayyidahnafisah@Sayyidahs-MacBook-Air firepwd % python3 firepwd.py
globalSalt: b'b5dbfec66b891e193f516eccaf39209a93634332'
 SEQUENCE {
   SEQUENCE {
     OBJECTIDENTIFIER 1.2.840.113549.1.5.13 pkcs5 pbes2
     SEQUENCE {
       SEQUENCE {
         OBJECTIDENTIFIER 1.2.840.113549.1.5.12 pkcs5 PBKDF2
         SEQUENCE {
           OCTETSTRING b'eaa234484d176f2f091e1a9b2162b550a9874bf9ced92daa19c43e058b1328cf'
           INTEGER b'01'
           INTEGER b'20'
           SEQUENCE {
             OBJECTIDENTIFIER 1.2.840.113549.2.9 hmacWithSHA256
           }
         }
       }
       SEQUENCE {
         OBJECTIDENTIFIER 2.16.840.1.101.3.4.1.42 aes256-CBC
         OCTETSTRING b'c361eb13322fd6004ad463de0ef2'
       }
     }
   }
   OCTETSTRING b'c2cb6eb12879f4c649458e59d634f355'
 }
clearText b'70617373776f72642d636865636b0202'
password check? True
 SEQUENCE {
   SEQUENCE {
     OBJECTIDENTIFIER 1.2.840.113549.1.5.13 pkcs5 pbes2
     SEQUENCE {
       SEQUENCE {
         OBJECTIDENTIFIER 1.2.840.113549.1.5.12 pkcs5 PBKDF2
         SEQUENCE {
           OCTETSTRING b'ae2720e0964ce4beabedba40345712df4234f9ac9cd86e53a08982b65abbbd9c'
           INTEGER b'01'
           INTEGER b'20'
           SEQUENCE {
             OBJECTIDENTIFIER 1.2.840.113549.2.9 hmacWithSHA256
           }
         }
       }
       SEQUENCE {
         OBJECTIDENTIFIER 2.16.840.1.101.3.4.1.42 aes256-CBC
```



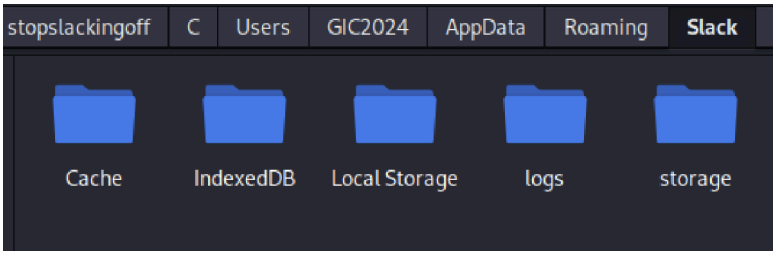
Finally we cracked the password and retrieved the flag after doing all the steps.

Flag:GCTF{m0zarella_f1ref0x_p4ssw0rd}

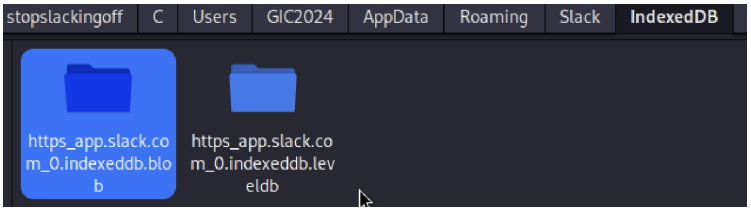
Stop Slacking Off



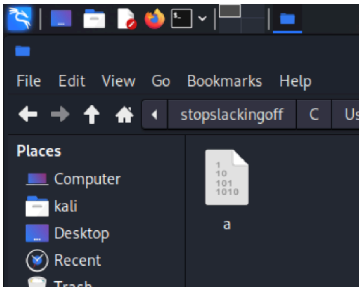
There are 5 folders in the Slack directory. I actually checked and `cat` all the files in those folders.



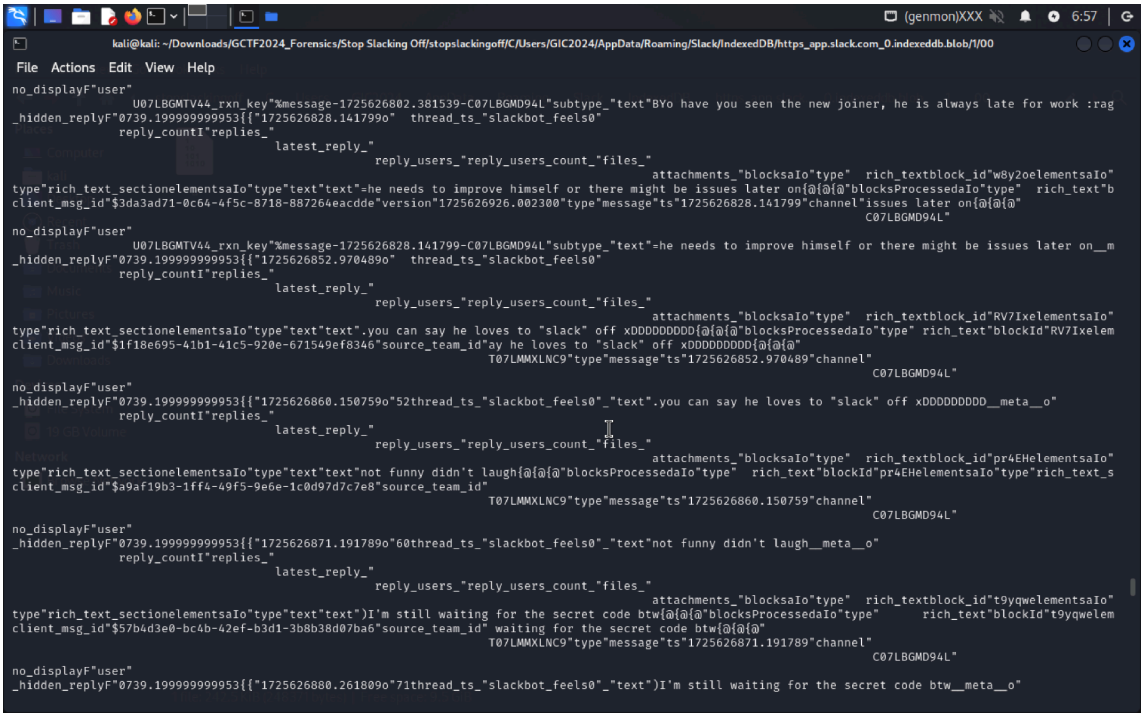
After carefully checked the content of the files, I finally found the conversations on `/Slack/IndexedDB/https_app.slack.com_0.indexeddb.blob`



Below was the only file in that folder and I checked the content using `cat a`.

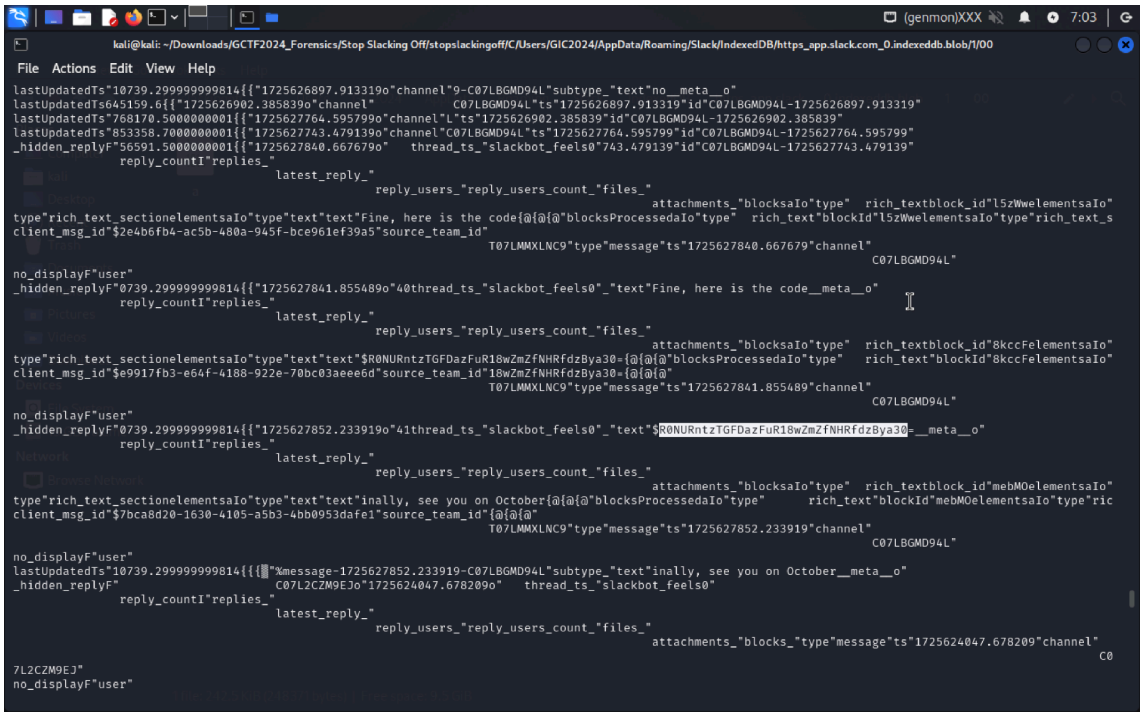


Found the conversation history on slack in that file.

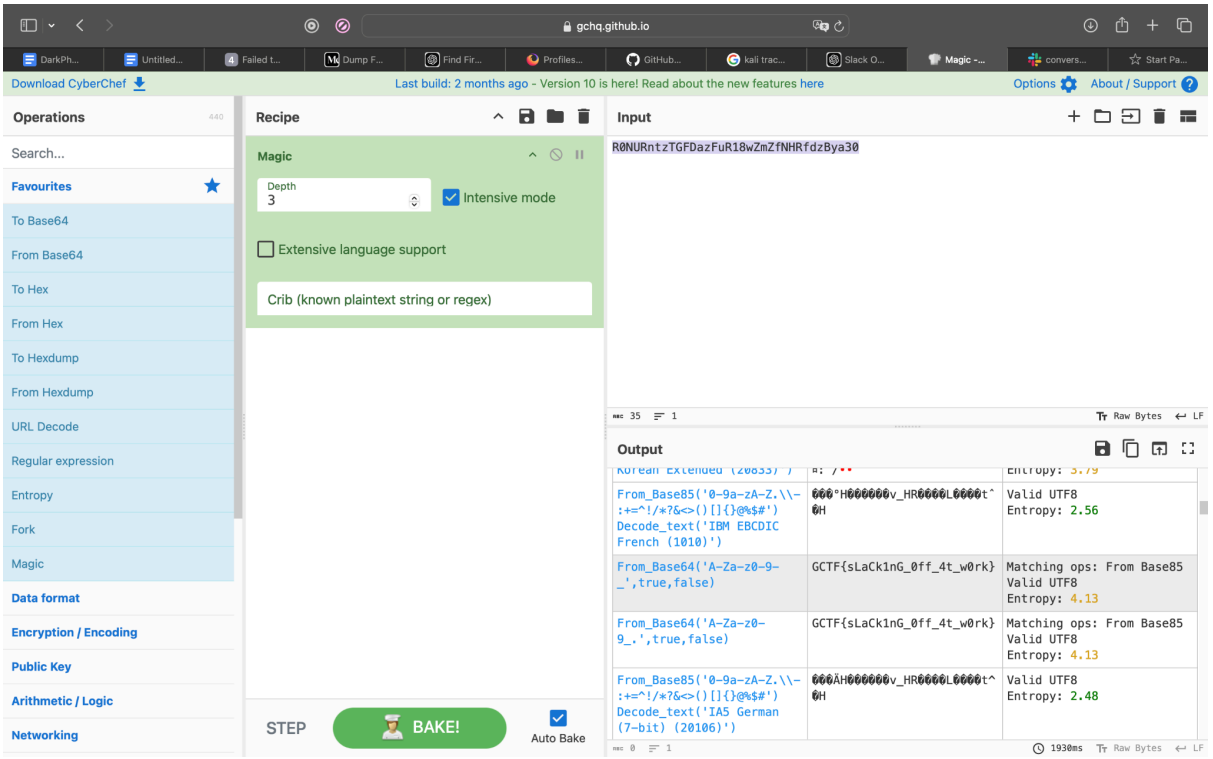


After carefully reading the conversations, I found the secret code which is:

R0NURntzTGFDazFuR18wZmZfNHRfdzBya30



As usual, to decrypt it, I just put in the cyberchef and let it do the work.



Flag: GCTF{sLaCk1nG_0ff_4t_w0rk}