CYSCOM VITCC
Knights CTF Walkthrough

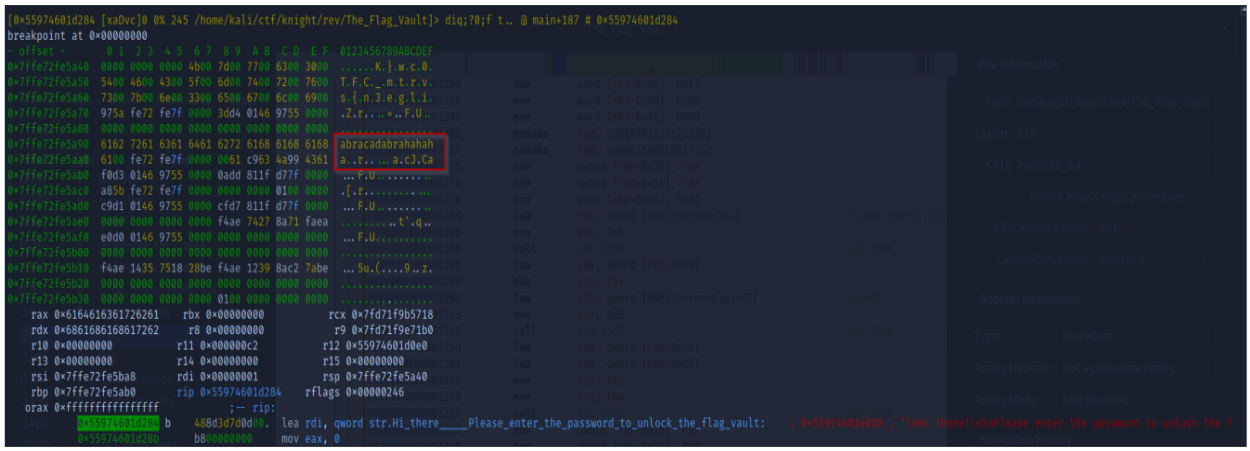
Reverse Engineering 1 - The Flag Vault

It's a simple password binary. By using the strings tool, we can find 2 strings -

abracadaH and brahahahH

After opening up the binary in radare2, I found that it was just concatenating both the strings while also removing the last H and then adds an additional a

So the final answer is abracadabrahahaha



FLAG = KCTF{welc0me_t0_reverse_3ngineering}

Reverse Engineering 2 - The Encoder

The binary was just encoding letters. 'A' = 1402 and so on.. Using this logic, we just find the difference between each number and 1402. According to the ASCII table, we can decode the initial string.

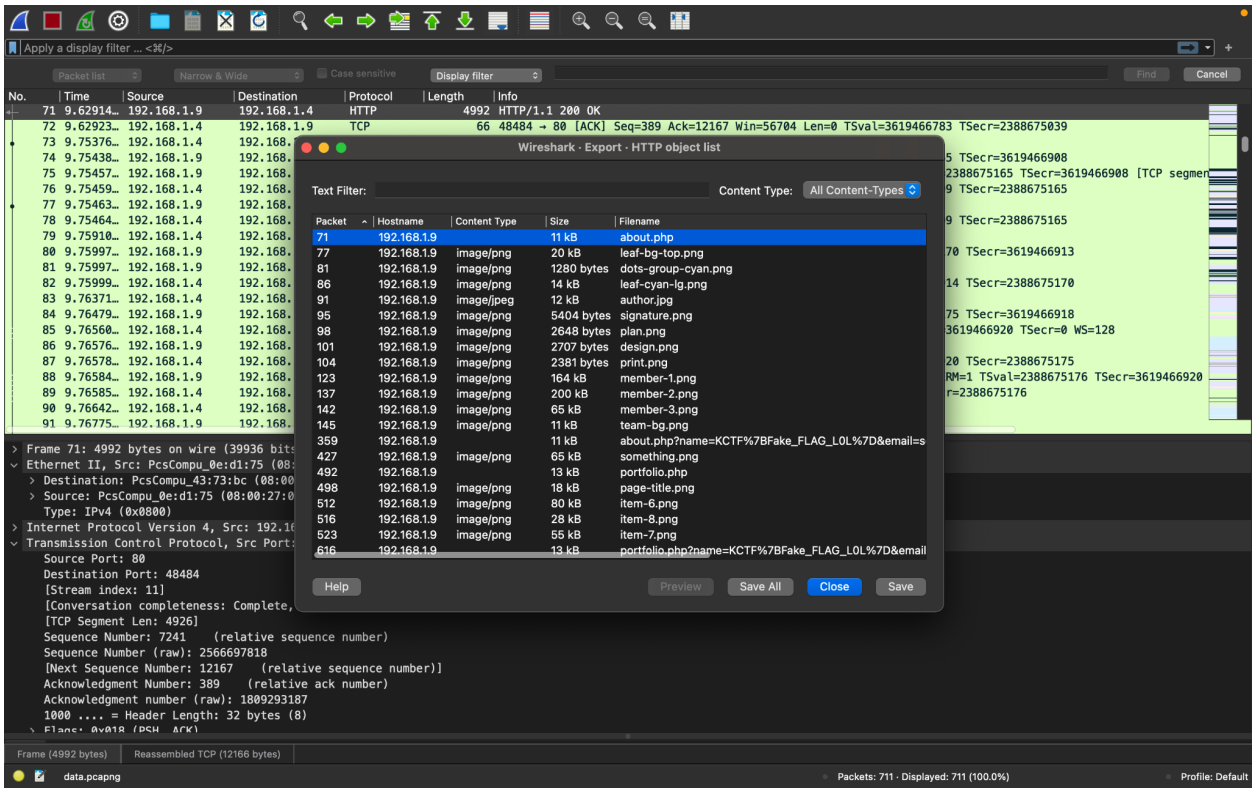
```
tmp = "1412 1404 1421 1407 1460 1452 1386 1414 1449 1445 1388 1432 1388 1415 1436
1385
1405 1388 1451 1432 1386 1388 1388 1392 1462"

tmp = tmp.split()
for i in tmp:
        i = int(i)
         i = i - 1402
         print(chr(ord('A') + i), end = "")
```

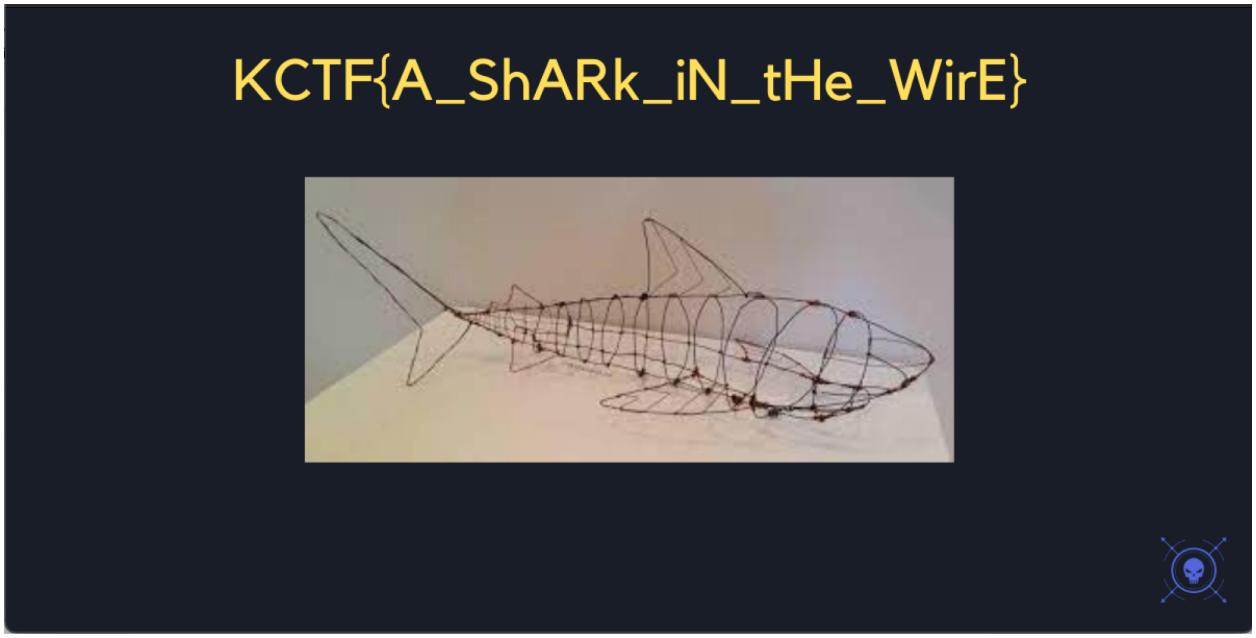
FLAG = KCTF{s1Mpl3_3Nc0D3r_1337}

Networking 1 - Hows the shark:

Open pcapng file
Go to File -> Export Objects -> HTTP



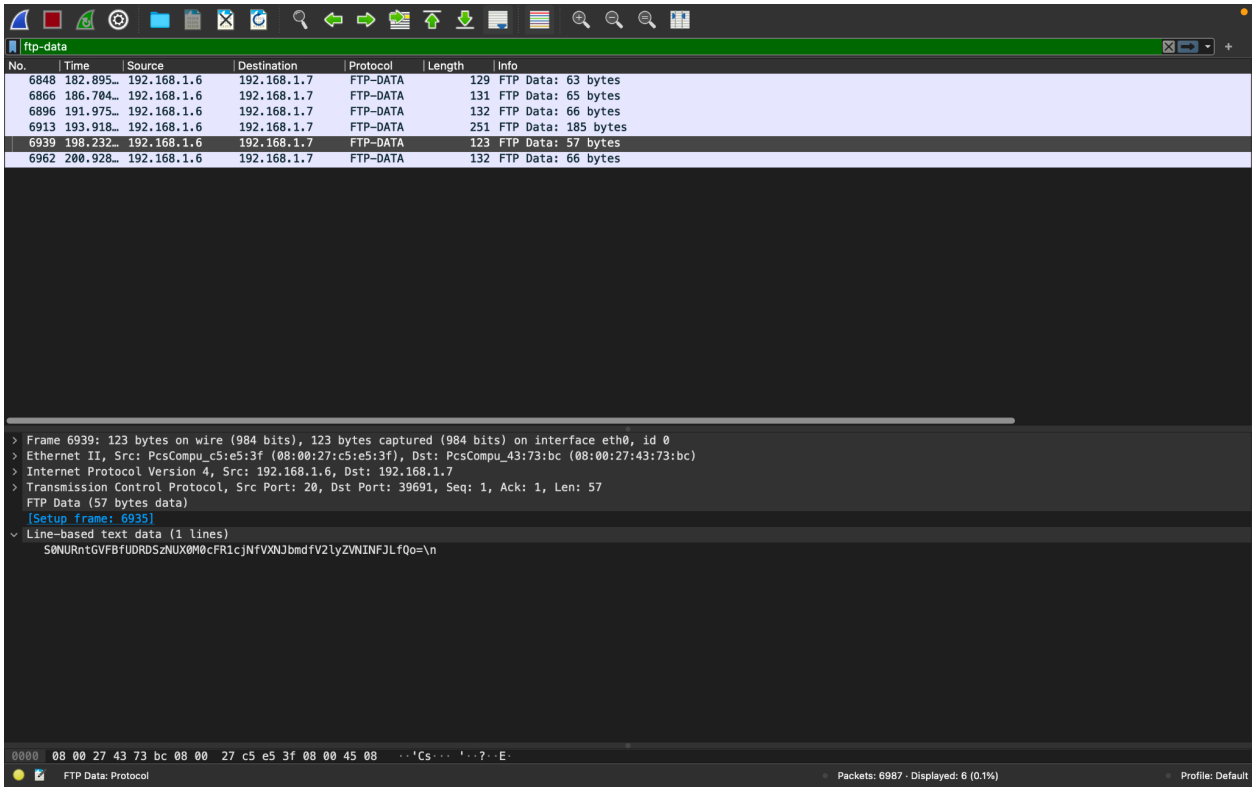
As we can see there are 4 unnamed packets. 2 of them are titled Fake Flags so lets not look there.
Saving the about.png file in my desktop reveals to be a photo which has the flag inside it!



FLAG - KCTF{A_ShARk_iN_tHe_WirE}

Networking 2 - Find the flag:

We can directly see that this pcapng files consists of FTP and TCP packets.
Applying the filter : ftp-data



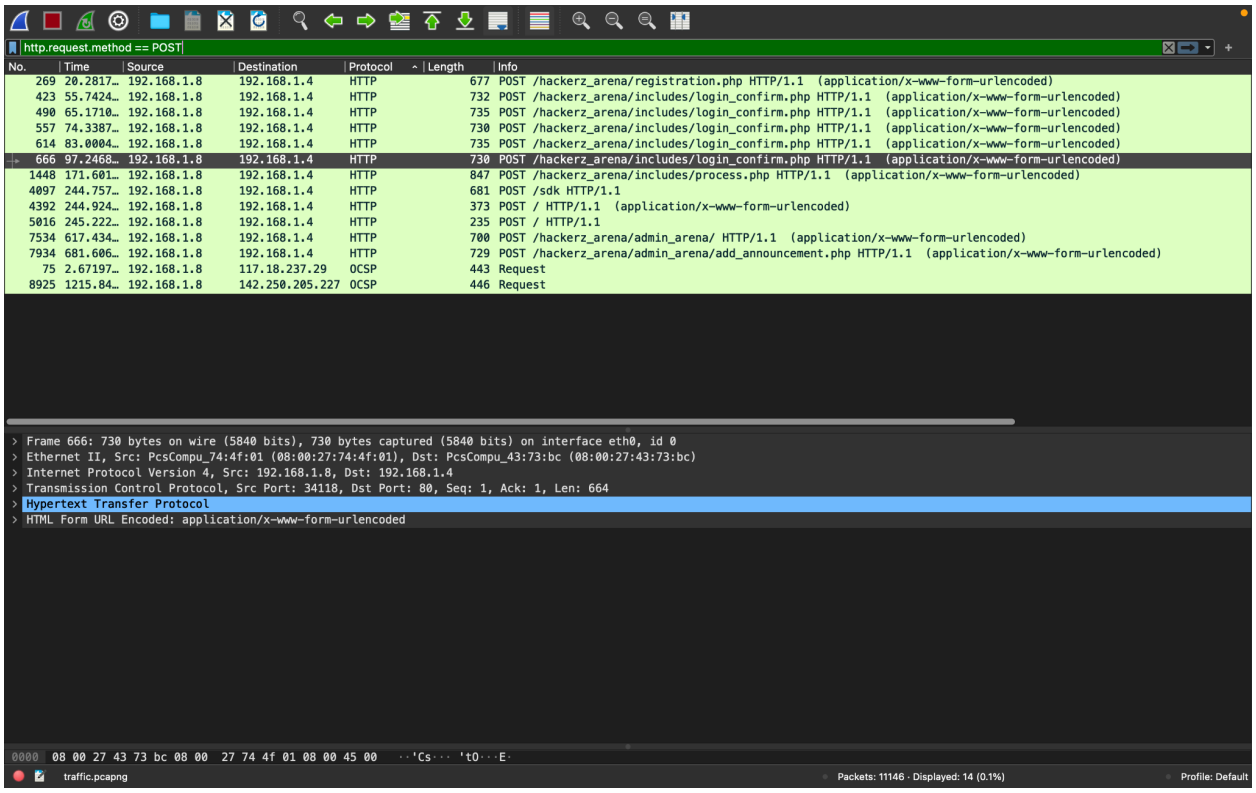
As we can see one packet has a base 64 encoded string. Decoding the string we get the flag.

FLAG - KCTF{FTP_P4CK3T_C4pTur3_UsIng_WireSH4RK}

Networking 3 - Compromised CTF Platform:

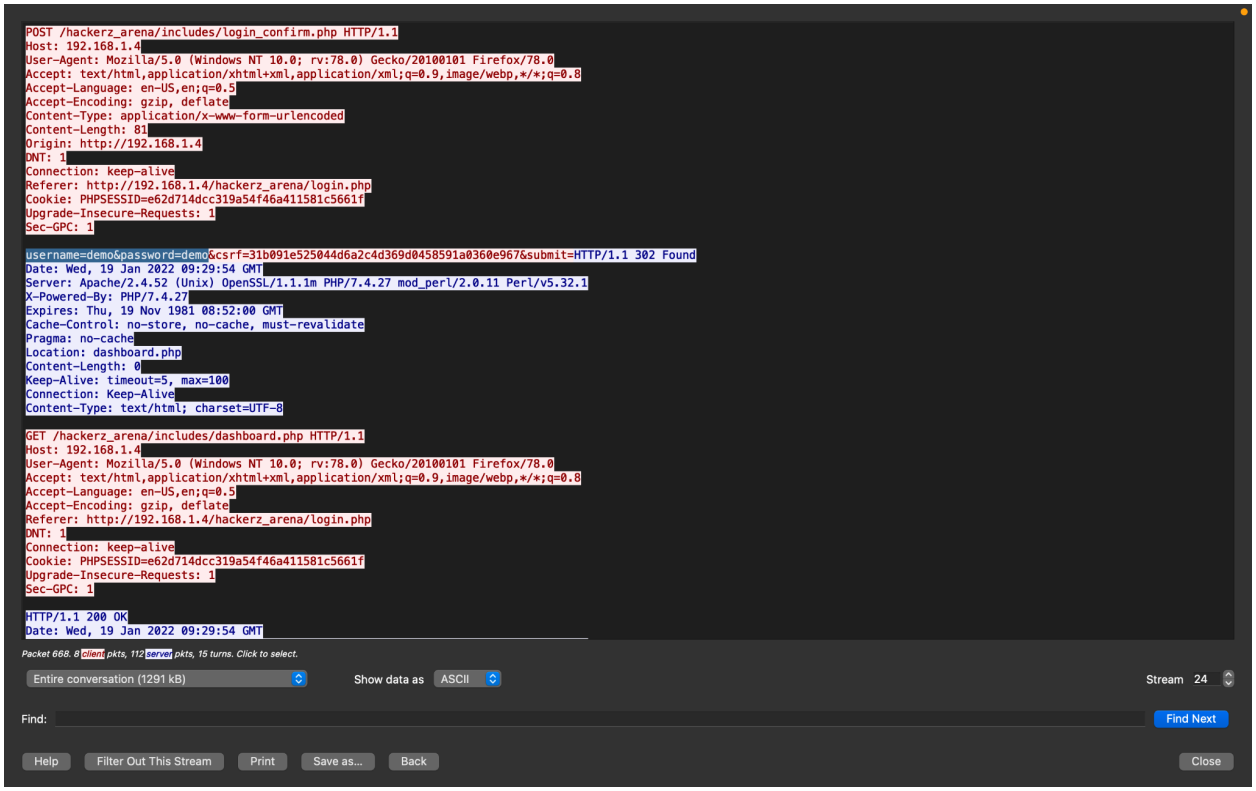
In this challenge we are asked to find the username and password. Since it's most likely to be from a login page, we apply the following filter:

http.requestmethod == POST



We can see multiple login attempts and can assume that the last /login-confirm.php must have accepted the hackers credentials.

Following the TCP stream of packet 666, we find the username and password

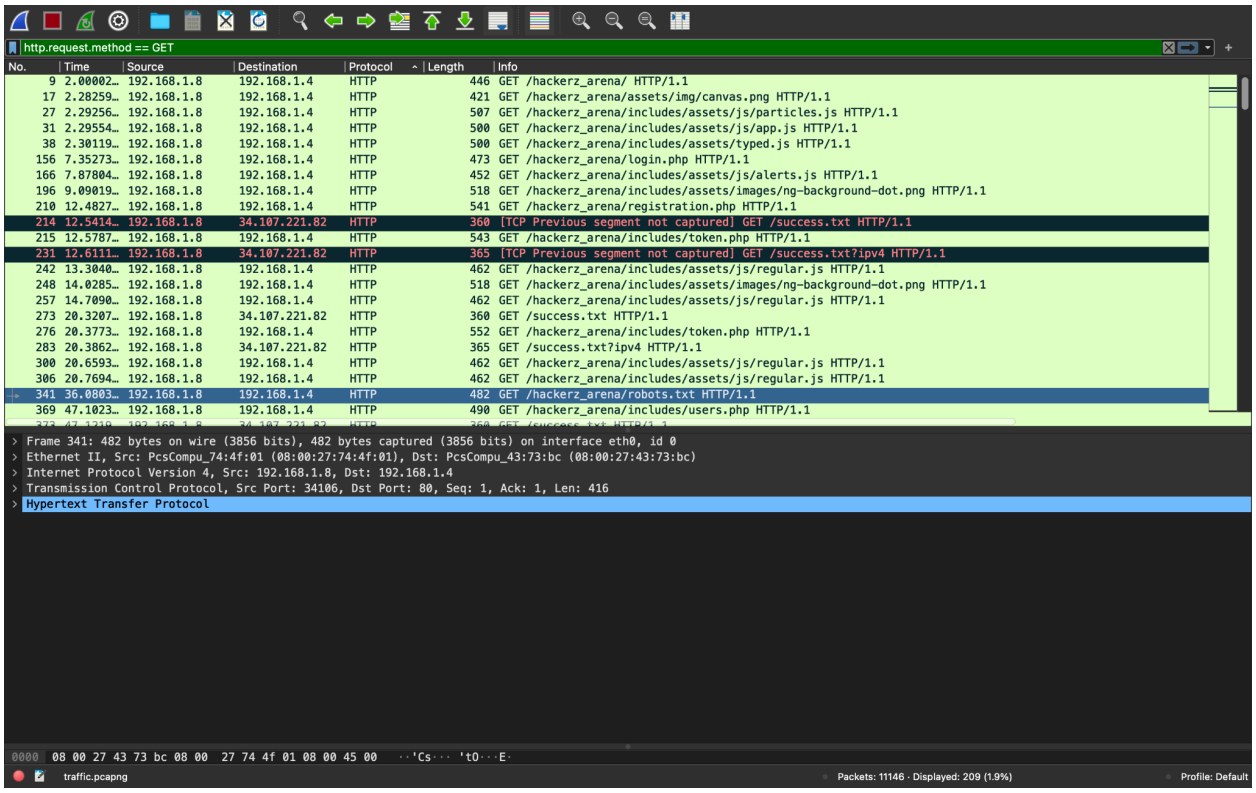


FLAG - KCTF{demo_demo}

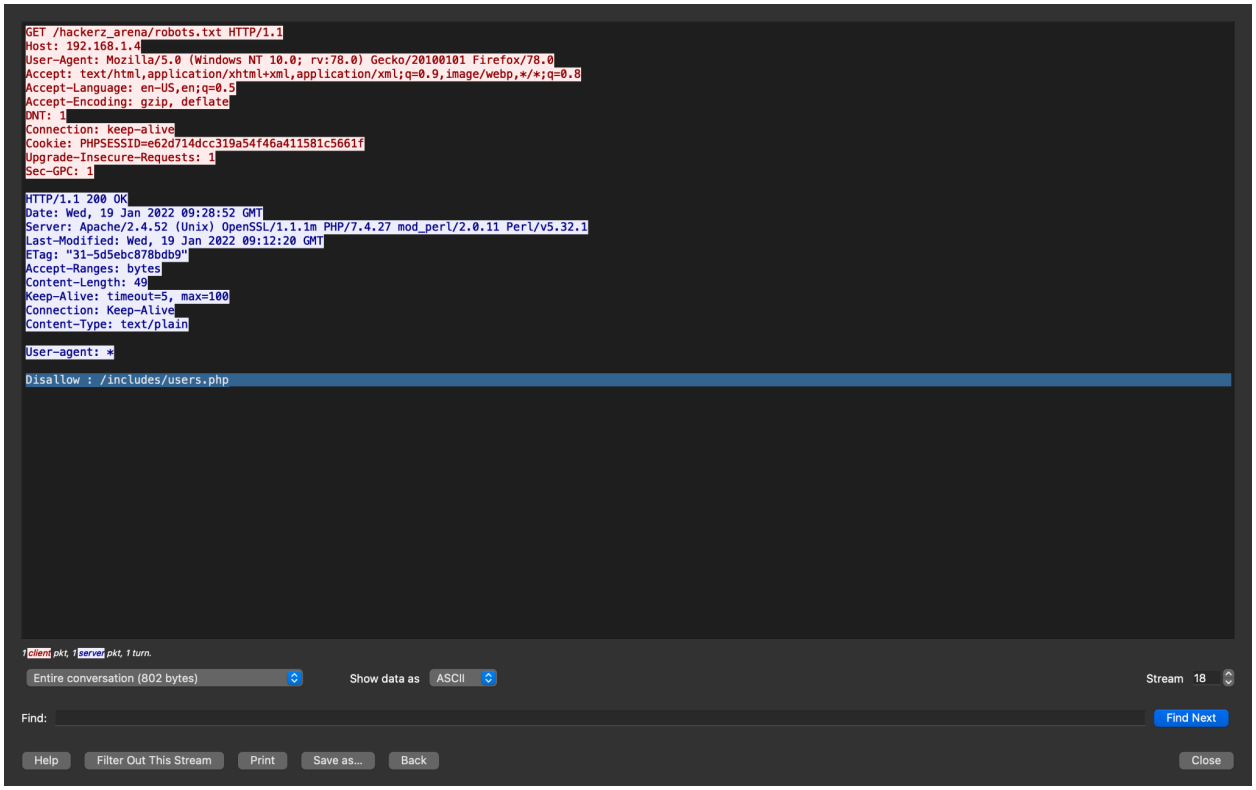
NETWORKING 3.a - Robots.txt

We have to use the same pcapng file from Networking 3. To view the robots.txt file we apply the following filter:

http.requestmethod == GET



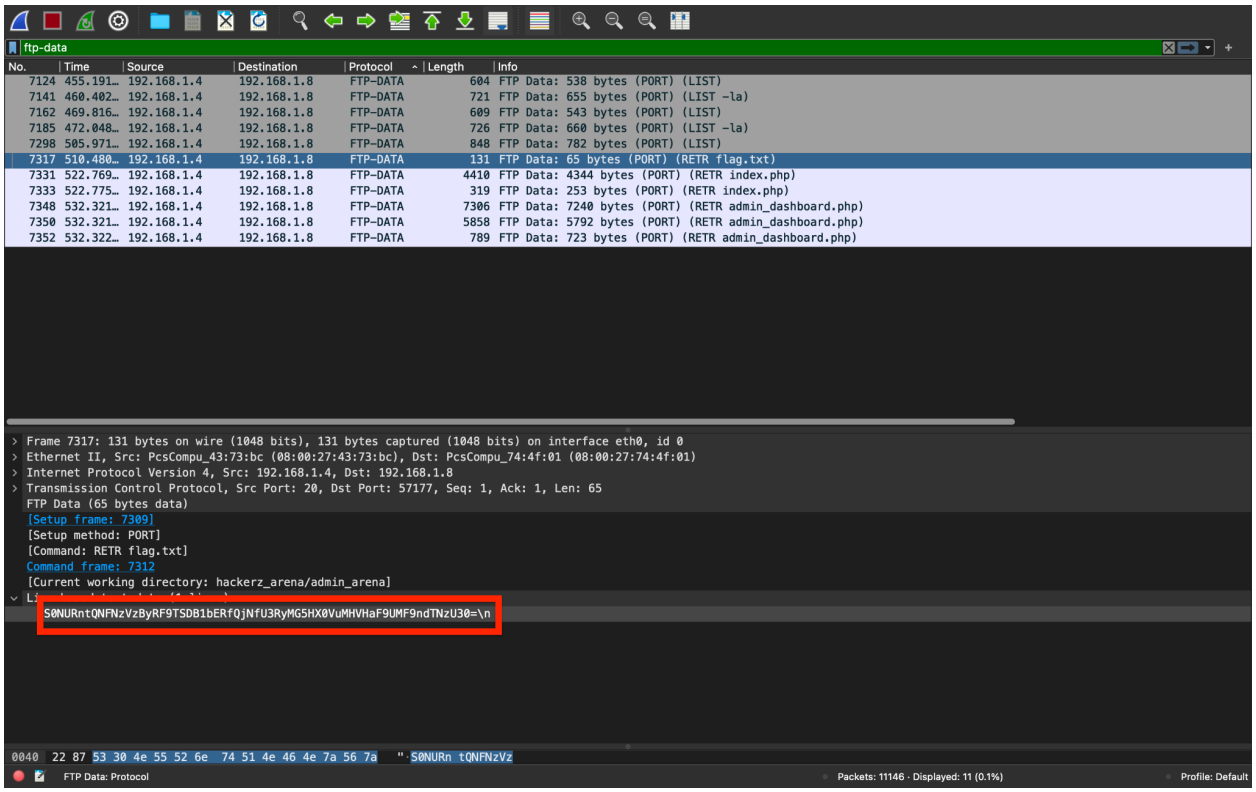
Following the TCP Stream we get the path inside robots.txt



FLAG - KCTF{/includes/users.php}

Networking 3.c- FTP flag

To find the ftp flag we apply the filter : ftp-data

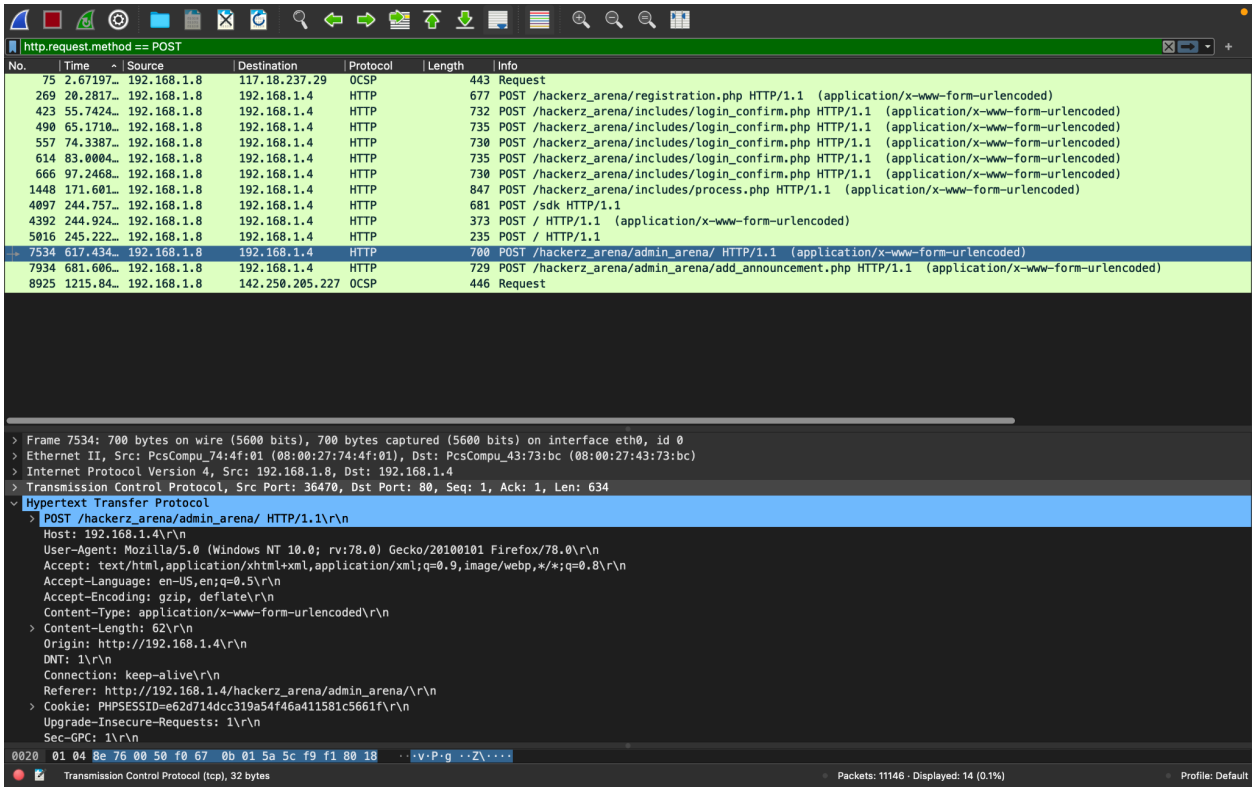


It's base64 encoded again. Decoding it reveals the flag.

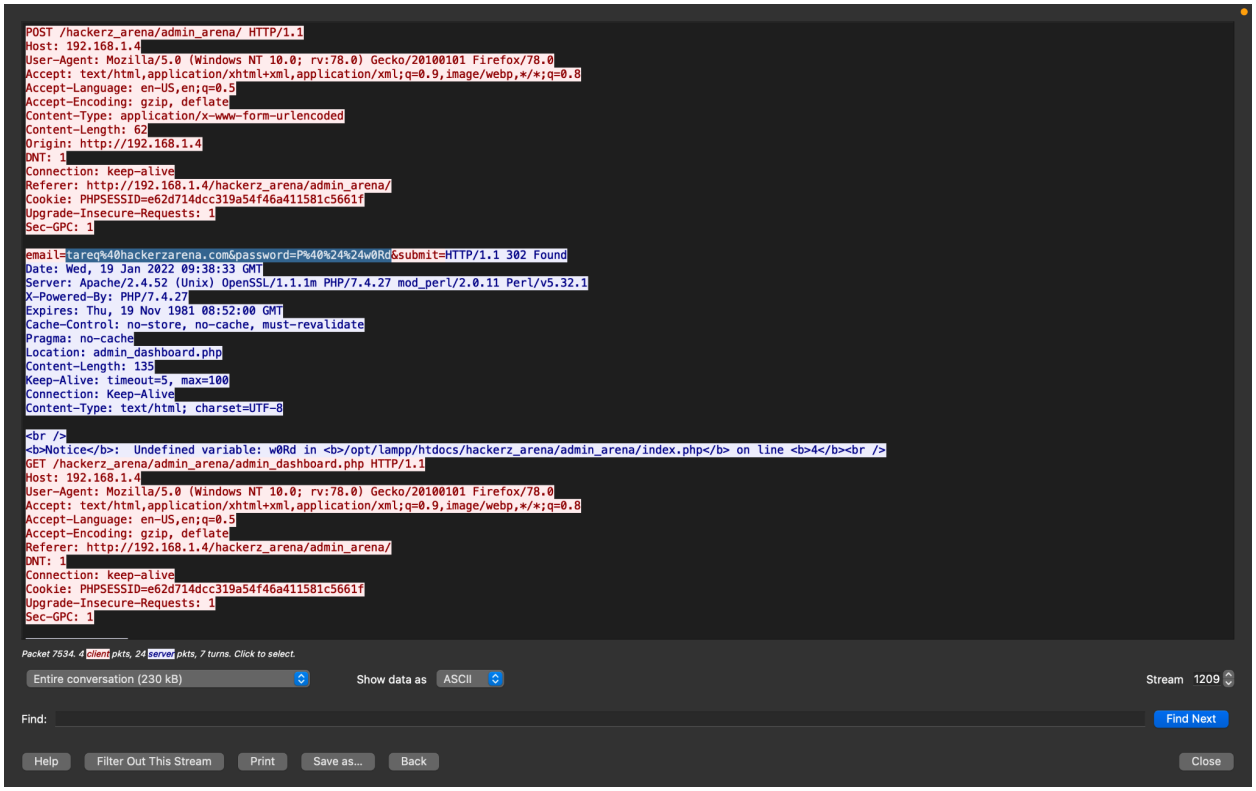
FLAG - KCTF{P4SsW0rD_SH0uID_B3_Str0nG_En0uGh_T0_gu3sS}

Networking 3.4- Admin Arena:

We have to find the email and password inside Admin Arena.
So applying the filter - http.requestmethod == POST, we search for the /admin_arena page:



Following the TCP stream we get the



Email - tareq%40hackerzarena.com
Password - P%40%24%24w0Rd

We can see that these are not the final credentials. Hex text is inserted in the password.
% is blank
40 is @
24 is $

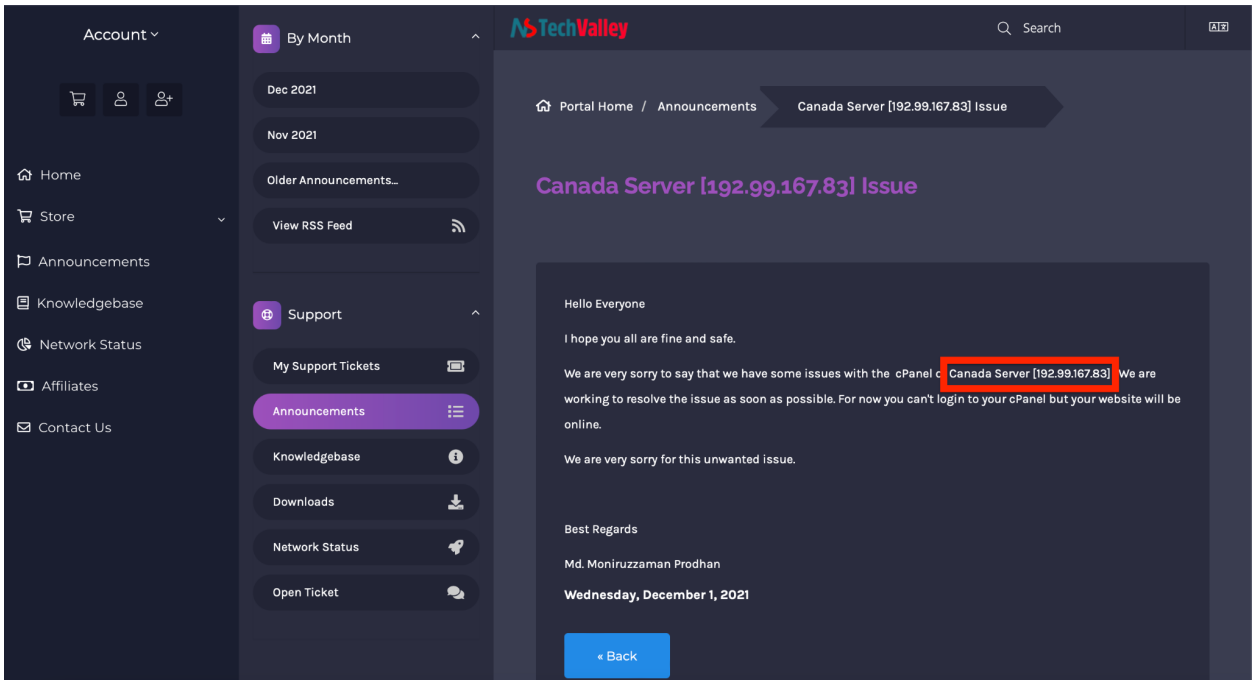
FLAG - KCTF{tareq@hackerzarena.com_P@$$w0Rd}

OSINT 1 - Canada Server:

Searching for NS Tech Valley on google I came across their facebook page. After inspecting all their posts I found a name -

Searching the name on google with the word “canada” to take advantage of google’s search algorithm we found the website -

https://clients.nstechvalley.com/announcements/3



FLAG - KCTF{192.99.167.83}

OSINT 3 - Find the camera:

In the image of the given bus we see the name of the photographer - **JenCh012**

After searching for her online it seemed impossible to find this exact picture as JenCh012 had apparently taken millions of pictures of buses.

So to make things easier this time I searched with - JenCh012 QV 6227 (number plate of the bus), and yes I found the image online.

Luxembourg, Van Hool New A308 # 206

Location:	**Lëtzebuerg – Esch-Uelzecht**
Facility:	**Tramways Intercommunaux dans le Canton d´Esch**
License Plate #:	**QV 6227**
Model:	**Van Hool New A308**
Since...:	**07.2007**
Built:	**07.2007**
Serial number:	**63942**
VIN:	**YE230802N94M63942**
Current state:	Sent to other company (or to the factory) (**10.2017**)
Purpose:	Passenger vehicle

Camera Settings

Model:	DSC-S980
Date and Time:	15.05.2010 15:51
Exposure Time:	1/320 sec
Aperture Value:	5.6
ISO Speed:	100
Focal Length:	23.2 mm
Show all EXIF tags	

So the model name was DSC-S980. Searching the same on google I found out that the manufacturing company was Sony.

FLAG - KCTF{SONY_DSC_S980}

Cryptography 1 - Passwd:

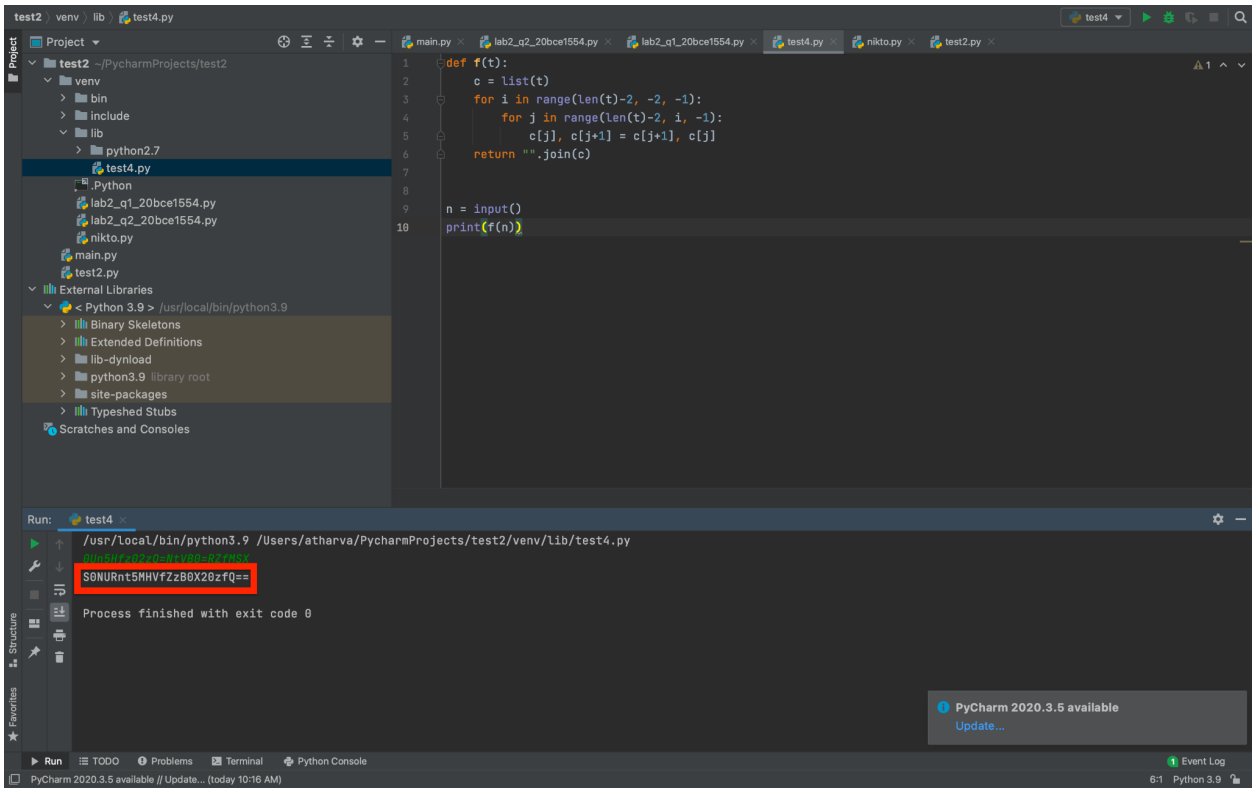
```
root:x:0:0:root:/root:/usr/bin/zsh
bin:x:1:1::/:/usr/bin/nologin
daemon:x:2:2::/:/usr/bin/nologin
mail:x:8:12::/var/spool/mail:/usr/bin/nologin
ftp:x:14:11::/srv/ftp:/usr/bin/nologin
http:x:33:33::/srv/http:/usr/bin/nologin
nobody:x:65534:65534:Nobody:/:/usr/bin/nologin
dbus:x:81:81:System Message Bus:/:/usr/bin/nologin
systemd-journal-remote:x:988:988:systemd Journal Remote:/:/usr/bin/nologin
systemd-network:x:987:987:systemd Network Management:/:/usr/bin/nologin
systemd-oom:x:986:986:systemd Userspace OOM Killer:/:/usr/bin/nologin
systemd-resolve:x:984:984:systemd Resolver:/:/usr/bin/nologin
systemd-timesync:x:983:983:systemd Time Synchronization:/:/usr/bin/nologin
systemd-coredump:x:982:982:systemd Core Dumper:/:/usr/bin/nologin
uuidd:x:68:68::/:/usr/bin/nologin
avahi:x:980:980:Avahi mDNS/DNS-SD daemon:/:/usr/bin/nologin
named:x:40:40:BIND DNS Server:/:/usr/bin/nologin
brltty:x:979:979:Braille Device Daemon:/var/lib/brltty:/usr/bin/nologin
colord:x:978:978:Color management daemon:/var/lib/colord:/usr/bin/nologin
cups:x:209:209:cups helper user:/:/usr/bin/nologin
dhcpcd:x:977:977:dhcpcd privilege separation:/:/usr/bin/nologin
dnsmasq:x:976:976:dnsmasq daemon:/:/usr/bin/nologin
git:x:975:975:git daemon user:/:/usr/bin/git-shell
mpd:x:45:45::/var/lib/mpd:/usr/bin/nologin
nbd:x:974:974:Network Block Device:/var/empty:/usr/bin/nologin
nm-openvpn:x:973:973:NetworkManager OpenVPN:/:/usr/bin/nologin
nvidia-persistenced:x:143:143:NVIDIA Persistence Daemon:/:/usr/bin/nologin
openvpn:x:972:972:OpenVPN:/:/usr/bin/nologin
partimag:x:110:110:Partimage user:/:/usr/bin/nologin
polkitd:x:102:102:PolicyKit daemon:/:/usr/bin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/usr/bin/nologin
rtkit:x:133:133:RealtimeKit:/proc:/usr/bin/nologin
sddm:x:971:971:Simple Desktop Display Manager:/var/lib/sddm:/usr/bin/nologin
tss:x:970:970:tss user for tpm2:/:/usr/bin/nologin
usbmux:x:140:140:usbmux user:/:/usr/bin/nologin
junior:[redacted]bin/zsh
knight:[redacted]:708697c63f7eb369319c6523380bdf7a:[redacted]home/junior:/bin/zsh
```

We can see that the password of knight is a hash. It is of the type MD5. Using any of the tools to convert the hash into simple text we get the flag.

FLAG - KCTF{exploit}

Cryptography 3 - Jumble:

We are provided with the encoded string and an enc.py file. It's obvious that the enc.py file was used to encode the string. Understanding the code, I wrote a python file to reverse or decode the string.



The decoded string received looks base64 encoded. Decoding the string gives the flag.

FLAG - KCTF{y0u_g0t_m3}

Programming 1 - Keep Calculating

```
x= 1

y= 2

ans = 0

for x in range(1,666+1):

      num = (10*x) + y

      ans += (x*y) + num
print(ans)
```

Ans = 2666664

FLAG = KCTF{2666664}

Programming 2 - Time complexity

```
procedure max (a1, a2, …, an: integers)
max := a1
for i :=2 to n
if max < a1 then max := ai
{max is the largest element}
```

FLAG - KCTF{O(n)}

Programming 3 - Reverse the answer

```
x= 1

calc = 0

ans = 0

for x in range(1, 543+1):

       calc = (x*(x+1)) + (2 *(x+1))

        tmp = str(calc)

        rev_calc = int(tmp[::-1])

        if(rev_calc%4 == 0):

           ans += rev_calc

print(ans)
```

Ans = 12252696

FLAG - KCTF{12252696}

Programming 4- Square Sum

Limit is 159 as its square is just above 25000.

```
for x in range(159):
       for y in range(159):

             if ((x*x) + (y*y)) == 25000:
                    print(str(x) + " " + str(y))
```

FLAG - KCTF{90,130}

Programming 5 - Something in common

Found the GCD using online calculator😓

GCD = 4305125
Sum of its integers = 20
Ans = 20*1234

Ans = 24860

FLAG = KCTF{24860}

Programming 6 - Find the number

```
def G_sum(n):
    if n < 0:

        return 0
    else:

        return (1/pow(2,n)) + G_sum(n-1)
print(G_sum(25))
```

Ans = 1.9999999701976776

FLAG = KCTF{1.9999999701976776}

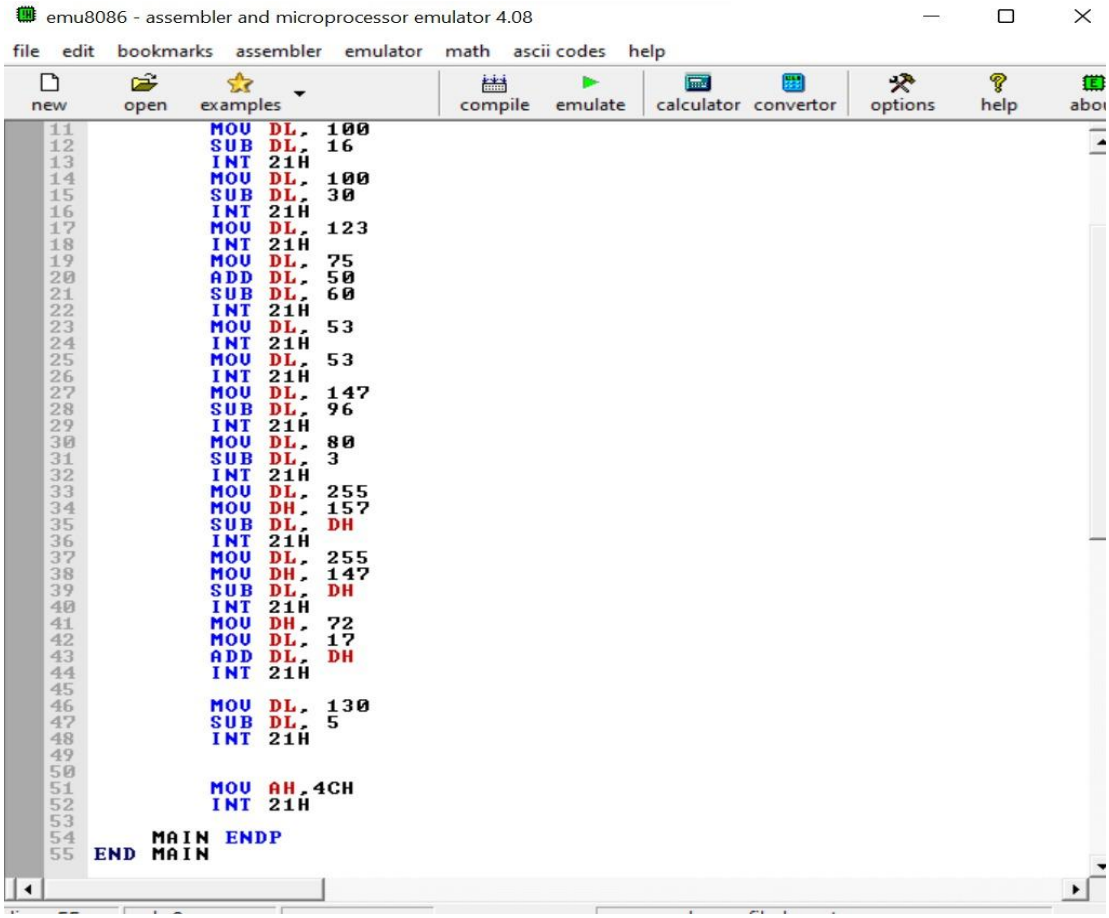
Programming 8 - Loop in a loop

I knew the flag format, hence I knew the basic positioning scheme. So i could decode it manually 😞

FLAG = KCTF{b451c_pr06r4mm1ng}

Programming 7 - Run the code

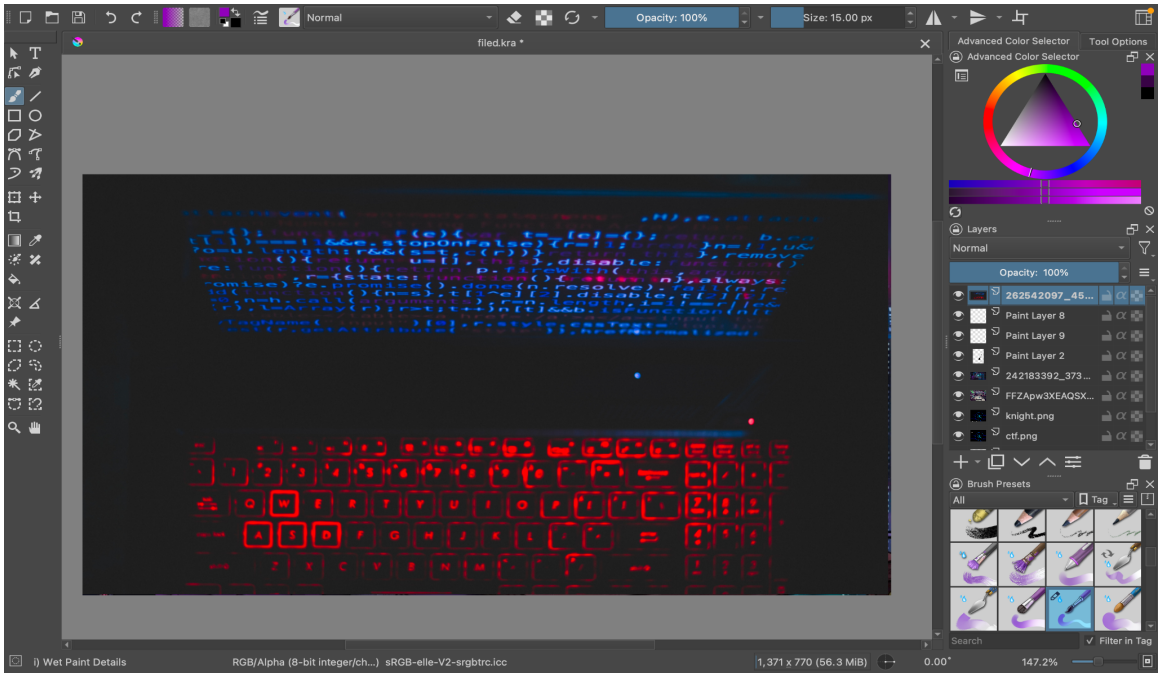
The code looks like a 8086 code. So running it in my 8086 emulator gives me the desired output.



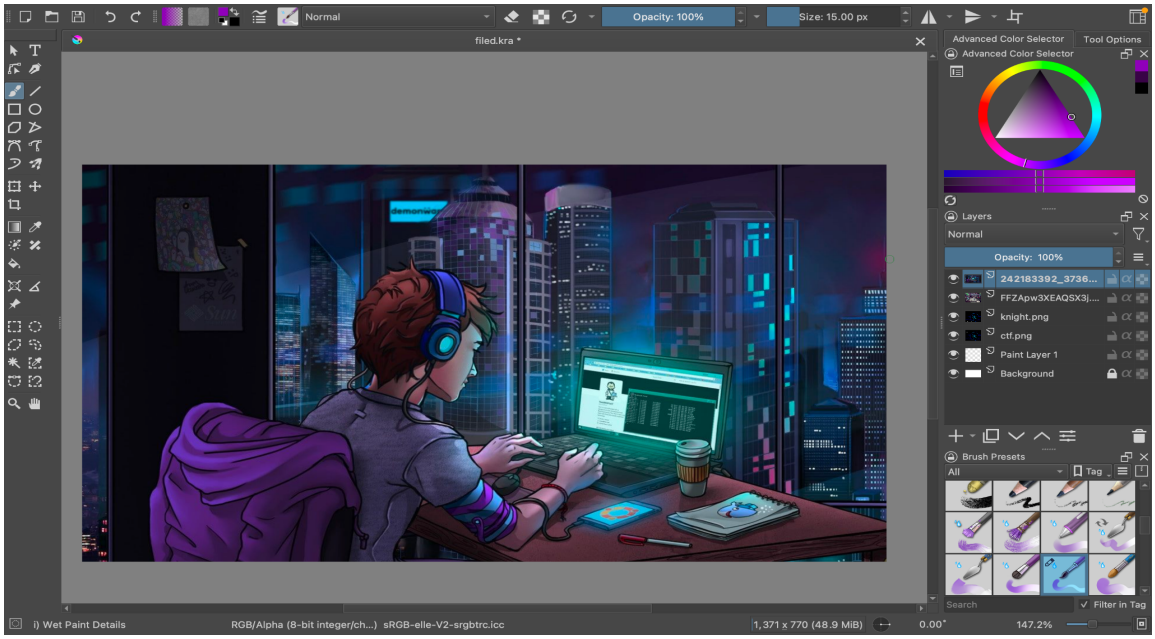
FLAG - KCTF{A553Mb1Y}

Steganography 1 - File D

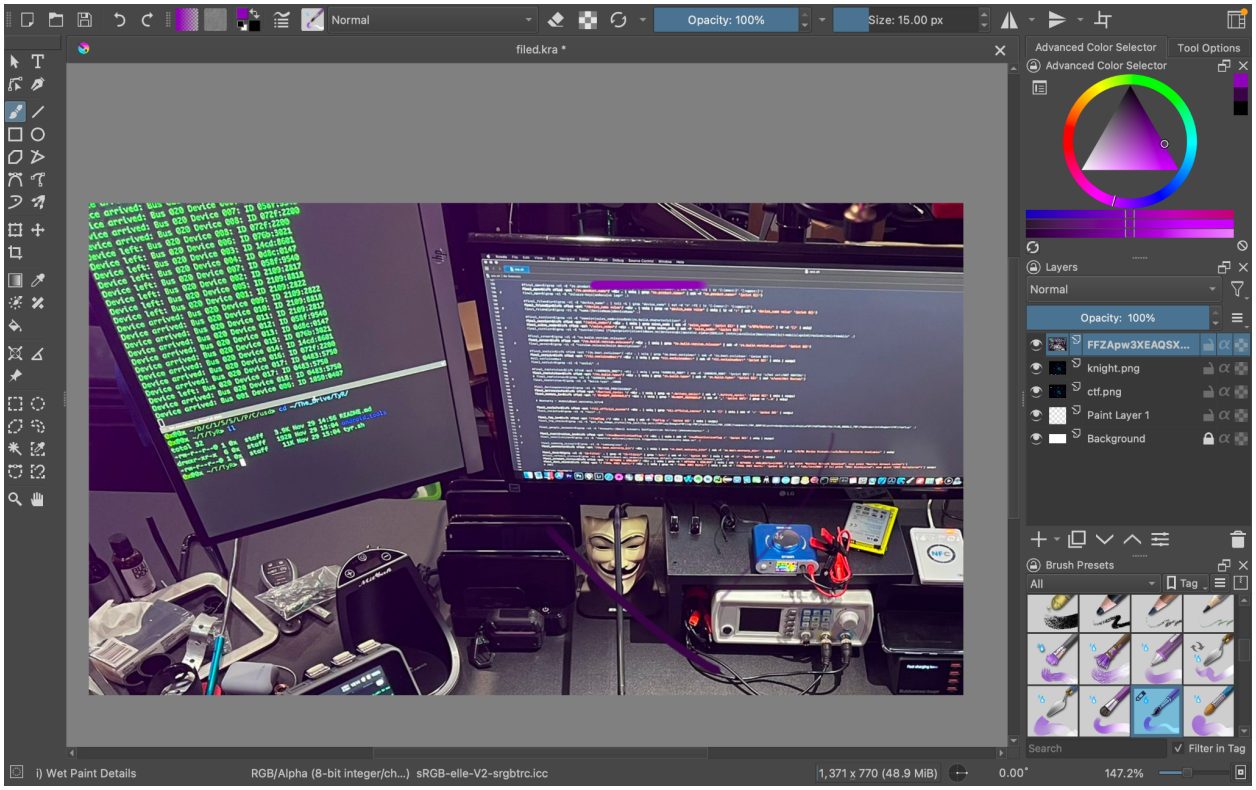
We get a .kra file. I installed krita, an application specifically made to open .kra files. Opening the file -

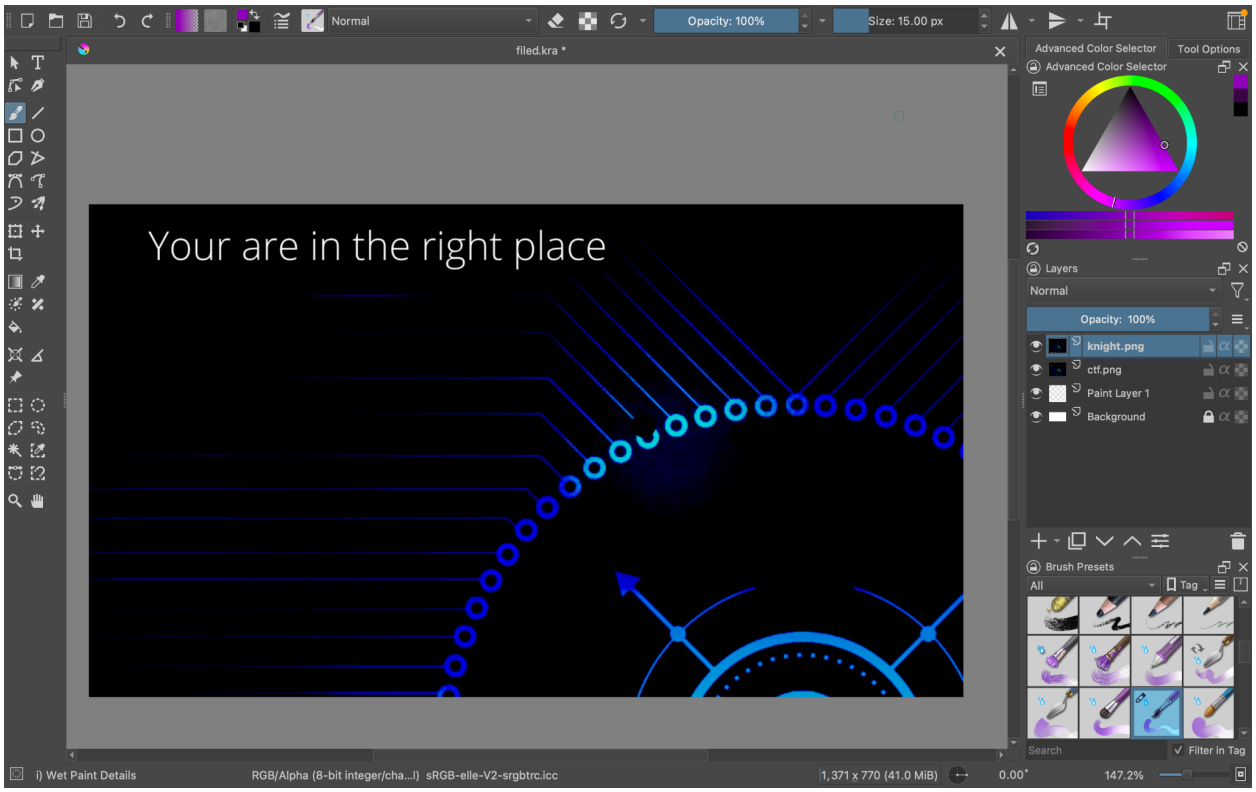


I can see on the extreme right side of the image, there seems to be another image. It seems like there's an image beneath this image. So removing the image on top -

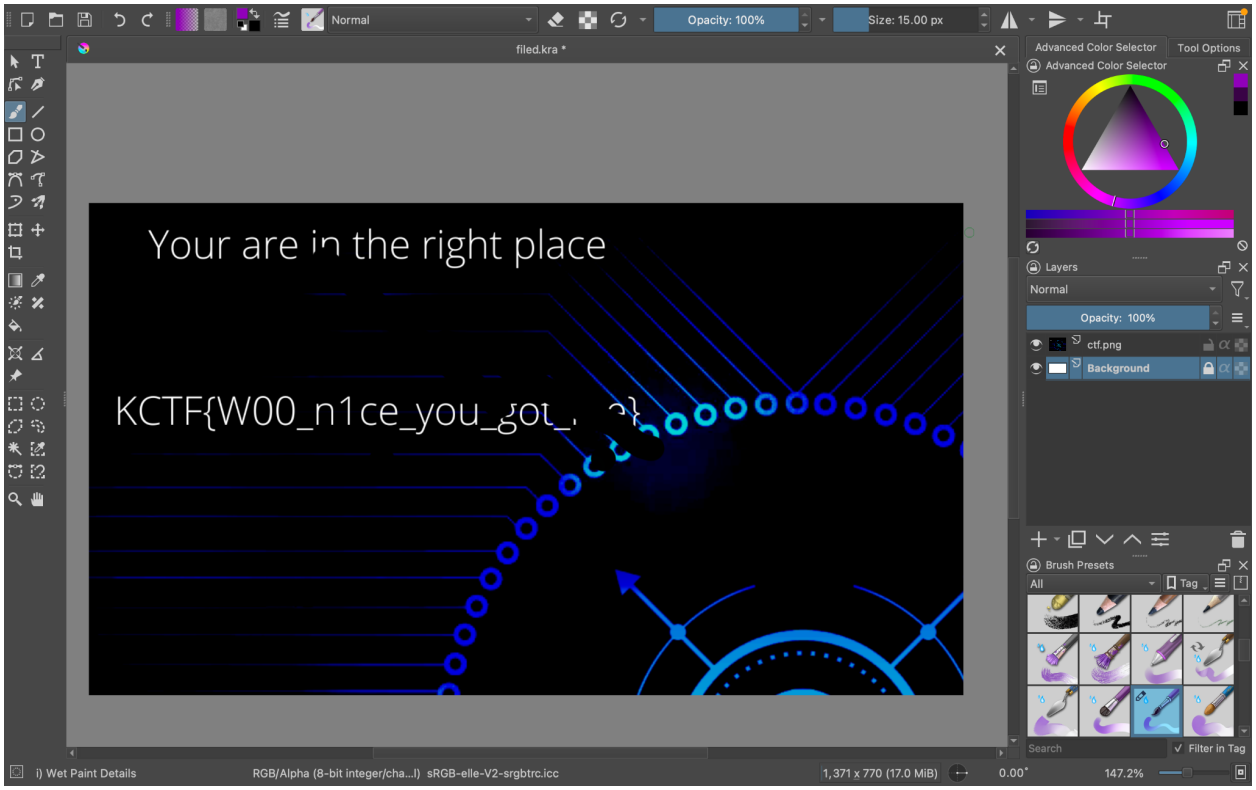


Having gotten the idea on how to move forward I start removing layers of pictures one by one -


filed.kra *
Normal
Opacity: 100%
Size: 15.00 px
Advanced Color Selector
Tool Options
Layers
knight.png
ctf.png
Paint Layer 1
Background
Brush Presets
i) Wet Paint Details
RGB/Alpha (8-bit integer/ch...) sRGB-elle-V2-srgbtrc.icc
1,371 x 770 (48.9 MiB)
147.2%


filed.kra *
Your are in the right place
Normal
Opacity: 100%
Size: 15.00 px
Advanced Color Selector
Tool Options
Layers
knight.png
ctf.png
Paint Layer 1
Background
Brush Presets
i) Wet Paint Details
RGB/Alpha (8-bit integer/cha...l) sRGB-elle-V2-srgbtrc.icc
1,371 x 770 (41.0 MiB)
147.2%

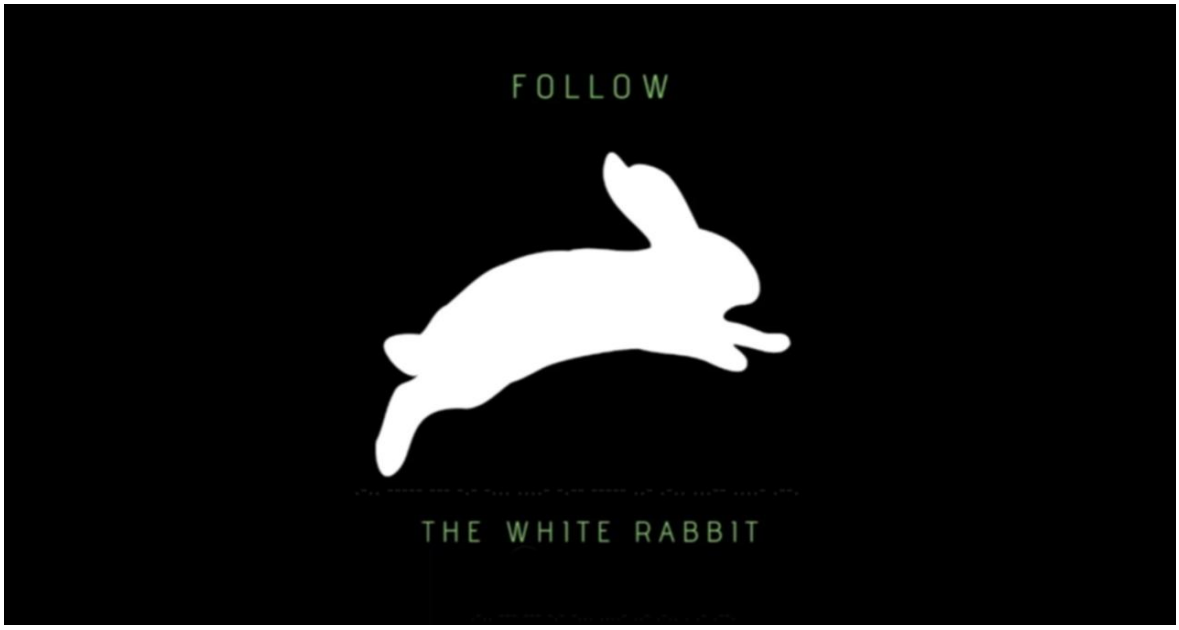
And finally -



FLAG - KCTF{W00_n1ce_you_got_me}

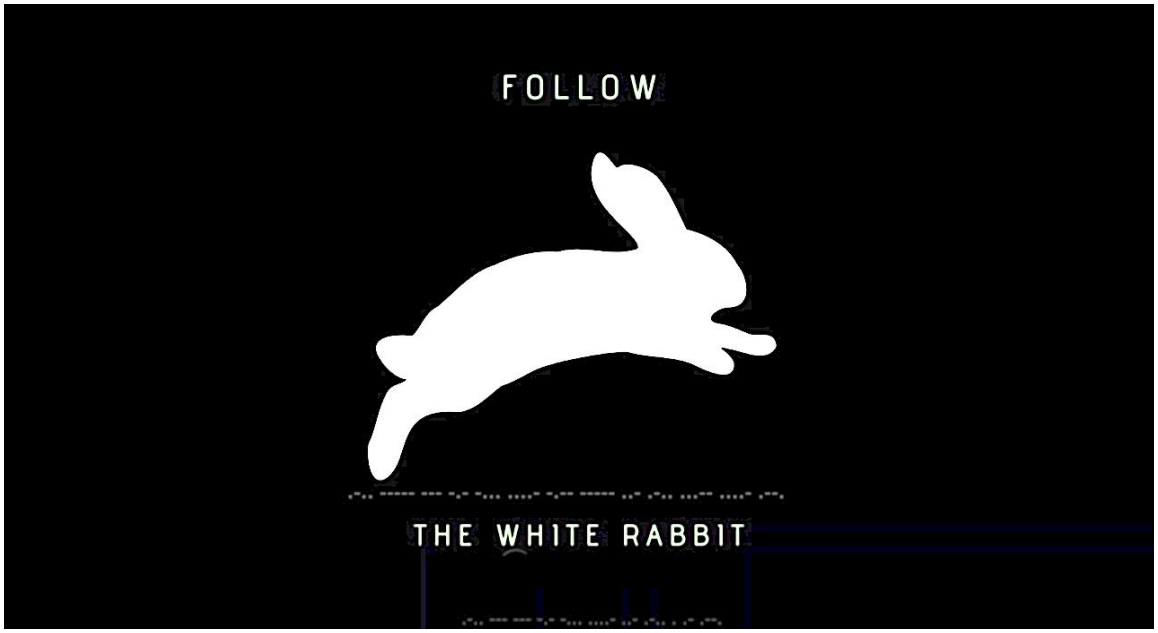
Steganography 2 - Follow the White rabbit

A photo with a white rabbit was given to us.



On closer inspection I felt like there was something in the background, just above the text “THE WHITE RABBIT”

So I changed the contrast and sharpness of the image and got something interesting.



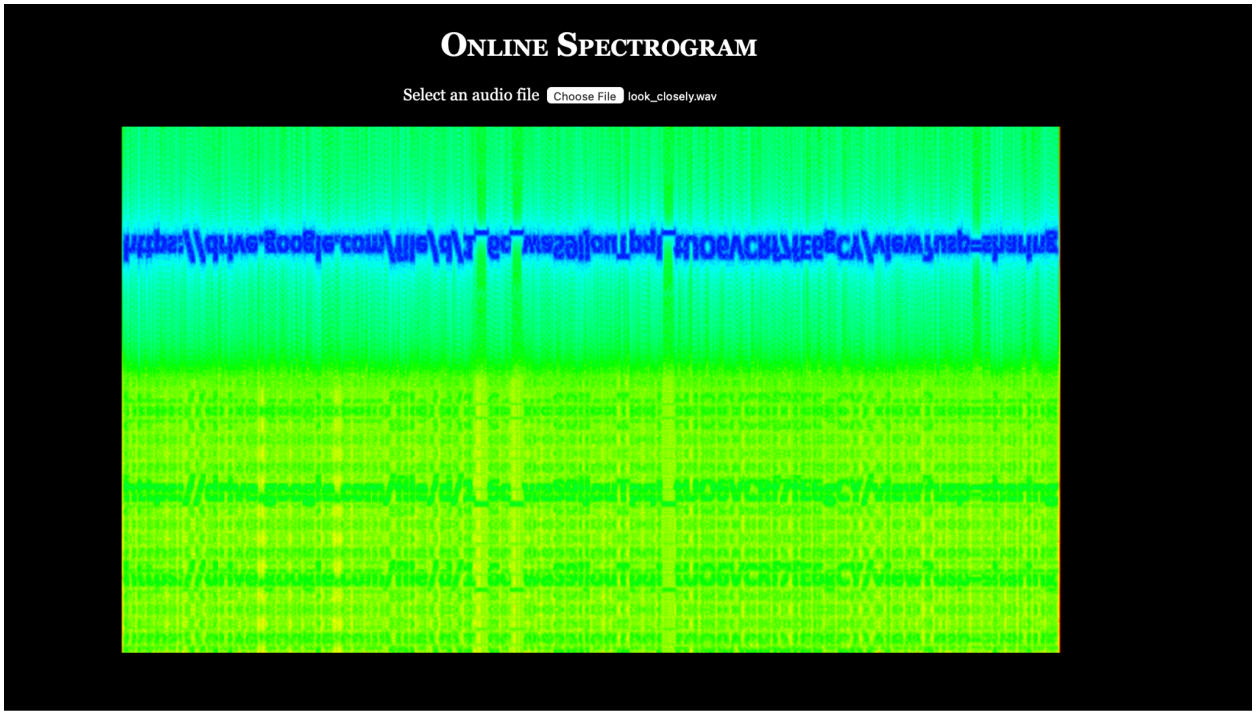
It was morse code. Translating the morse code above the text, “THE WHITE RABBIT” we got - LOOKB4ULEAP

And translating the one below the text we got - L0OKB4Y0UL34P

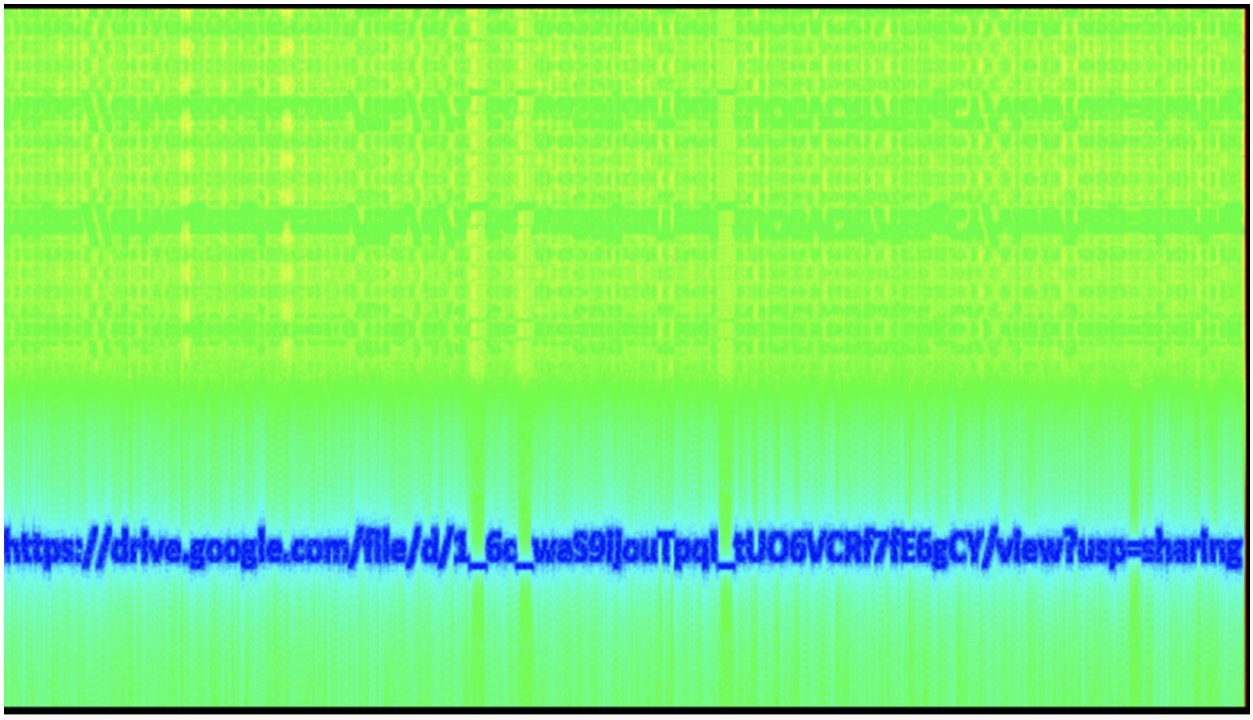
FLAG - {L0OKB4Y0UL34P}

MISC 4 - Look closely:

We were given a .wav file. I went forward to an online spectogram analyzer and uploaded the .wav file there.

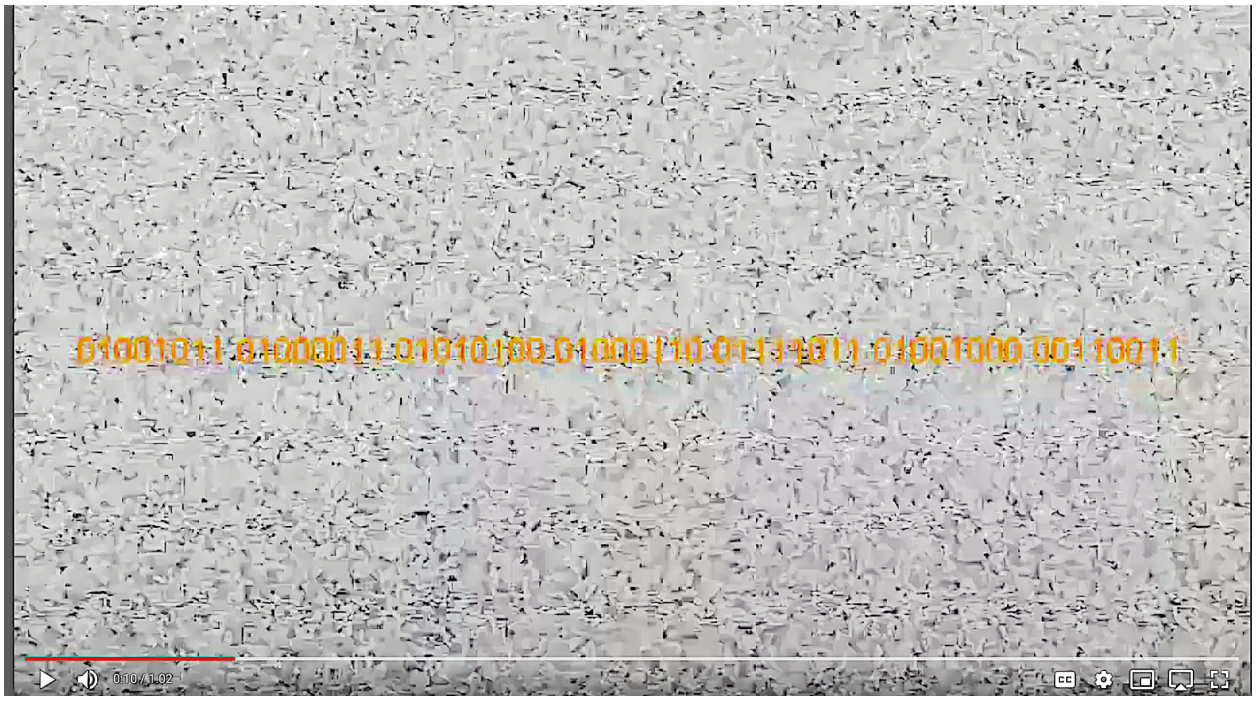


Since it was an inverted image, I had to mirror image it vertically -



I got a link! - https://drive.google.com/file/d/1_6c_waS9ijouTpqI_tUO6VCRf7fE6gCY/view

There we had a video where upon closer look we could see some blurred yellow text coming for a second. I screenshoted the 2 occurrences and adjusted the colours to reveal the text.



I got a binary - 0100101101000011010101000100011001111011010010000011001101001100010011000100111101011111010010100011001101001100010011000100111101111101

Converting it to text got me the flag.

FLAG - KCTF{H3LLO_J3LLO}