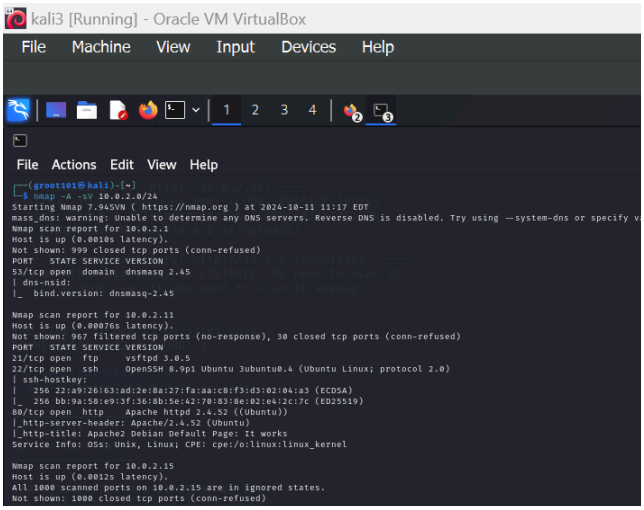
Ethical Hacking Package
Hogwarts Heist - Writeup

22PC02: Abhinivesh
22PC23: Navya
22PC26: Praveshini S B

All the machines are on the NAT Network.
Kali VM: 10.0.2.15

NMap is used to find active hosts on the network. It finds a host with the IP - 10.0.2.16.
The information from the application service versions suggest that it is an Ubuntu Machine.
There are 3 open ports:

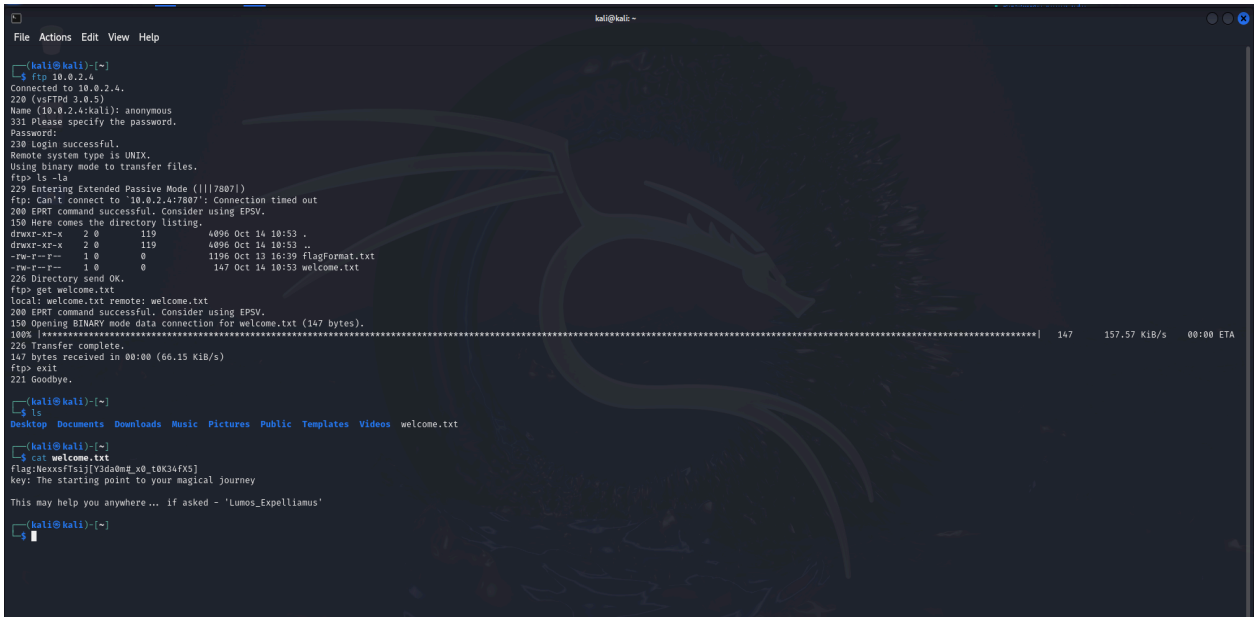
- FTP: Port 21
- SSH: Port 22
- HTTP: Port 80



It is seen that anonymous login is allowed on the FTP Port.
There is a file containing the flag format and another rule.

```
Flag Format: PotterHead[flag]
Passwords for files(if any): all lowercase letters
```

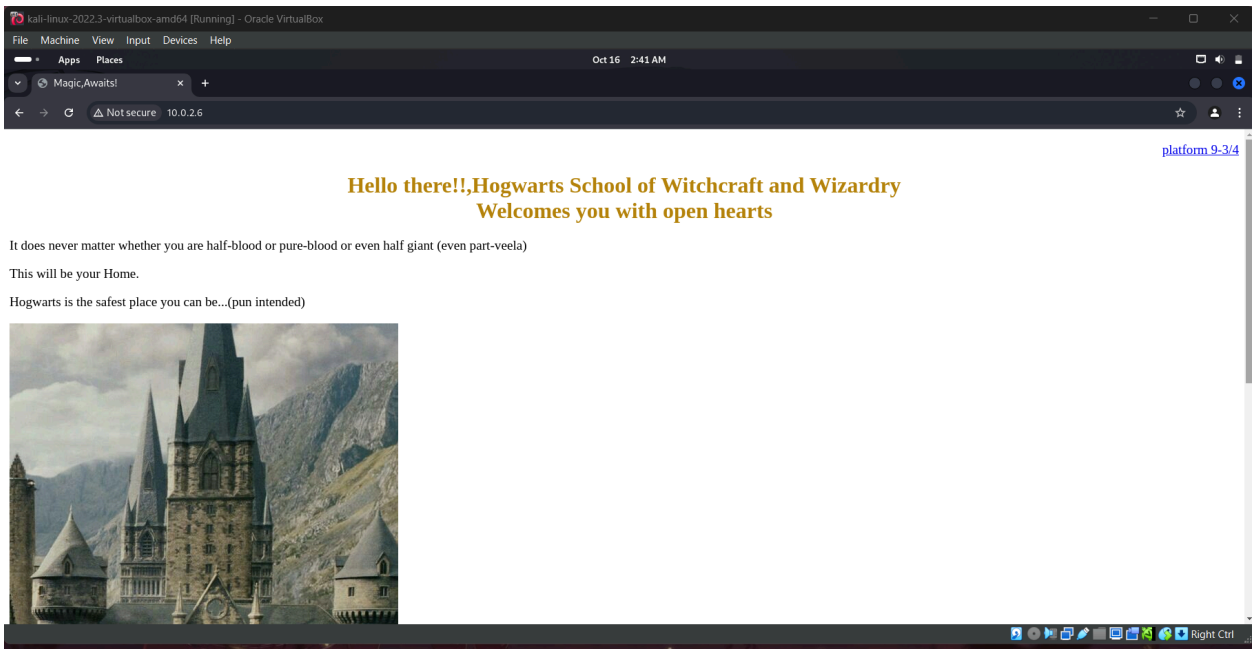
In ftp another file welcome.txt is there



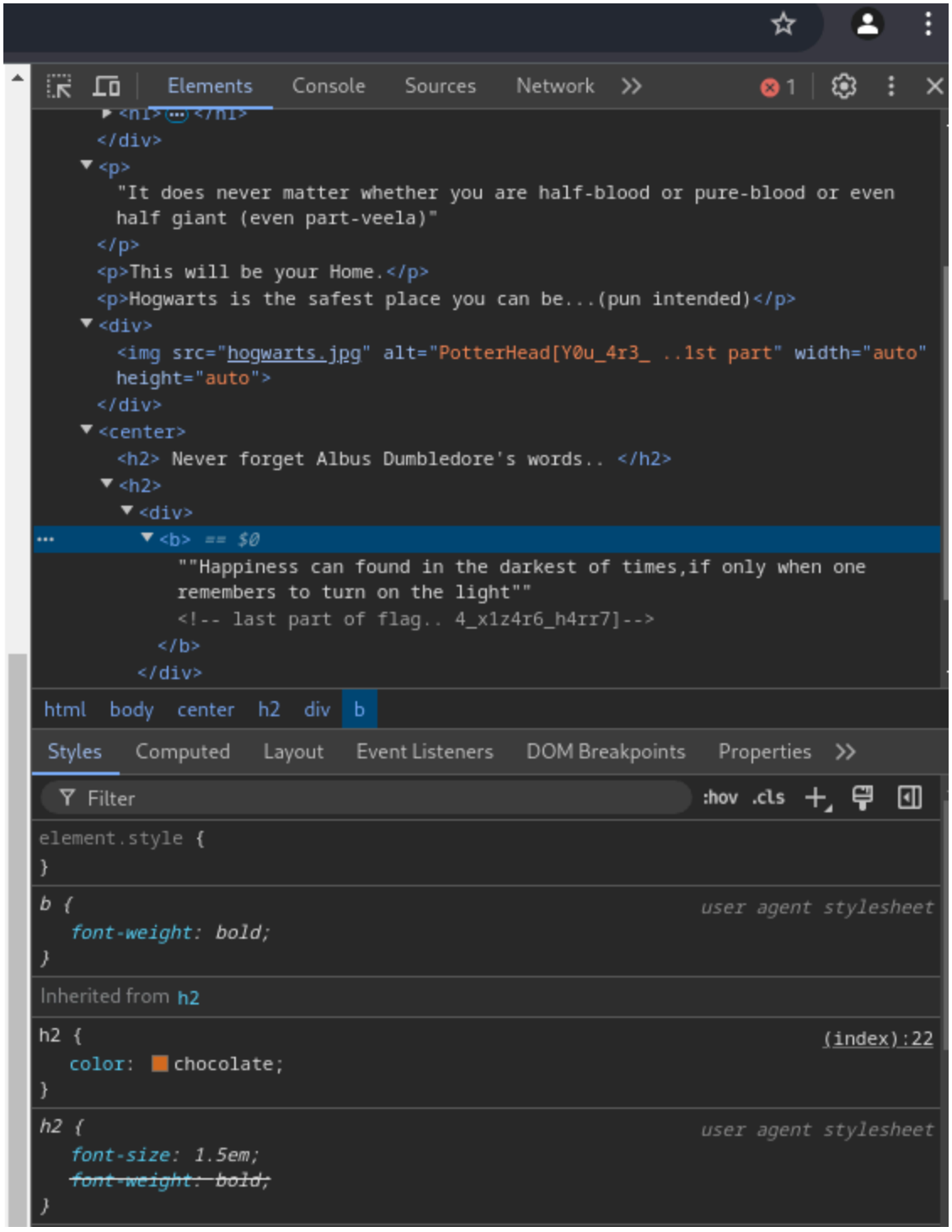
We should decode the affine cipher with key 9,39
We get the flag = **PotterHead[W3lc0m#_t0_h0G34rT5]**

With the flag format we have a first flag - that is encoded and encrypted
With hint

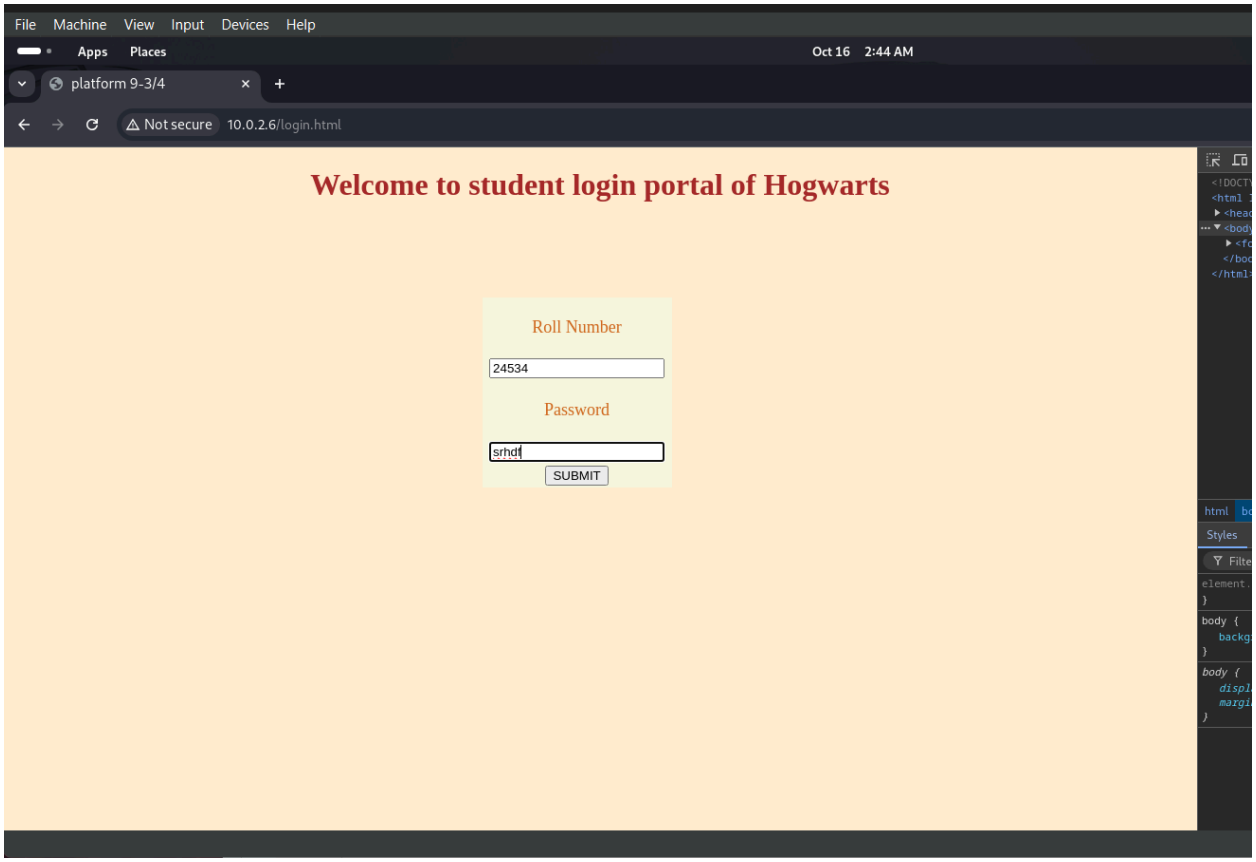
Next we have http port that can be visited



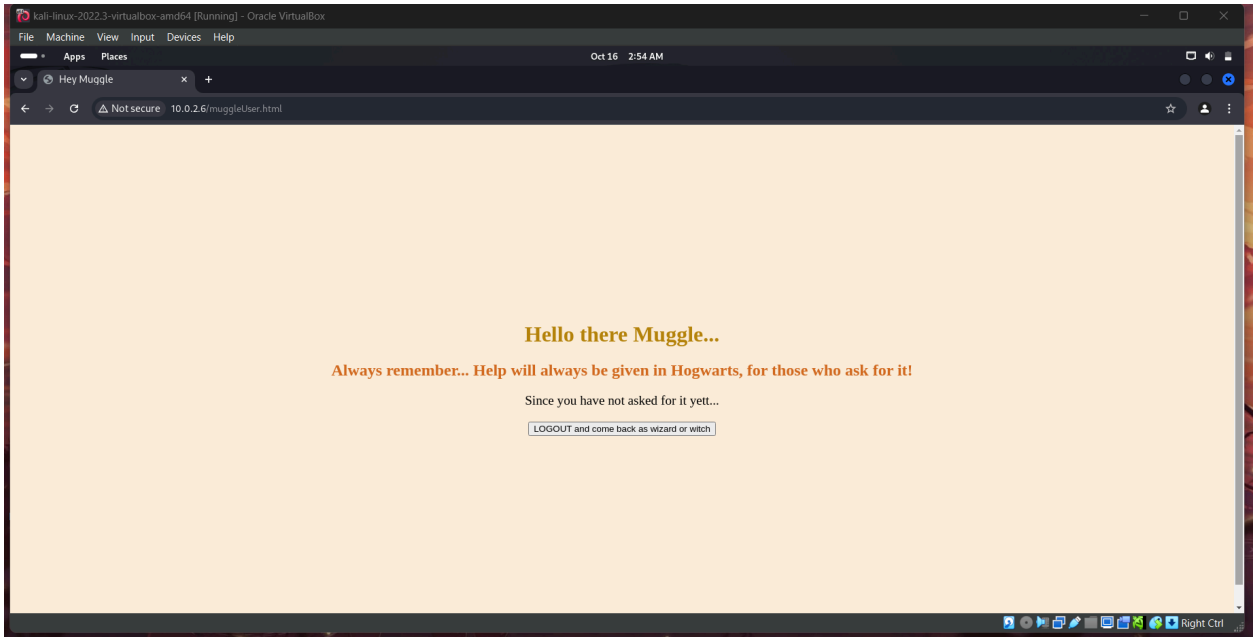
We inspect this page and find 2 parts of a single flag



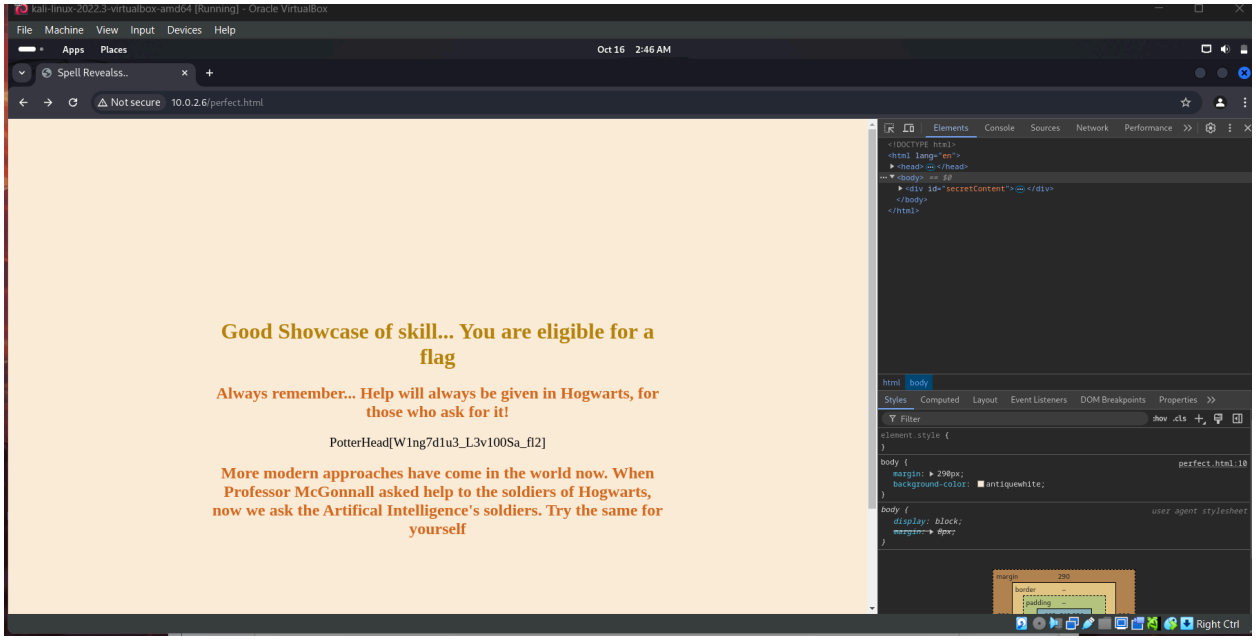
Next we have a link, platform 9 ¾
We go into it,



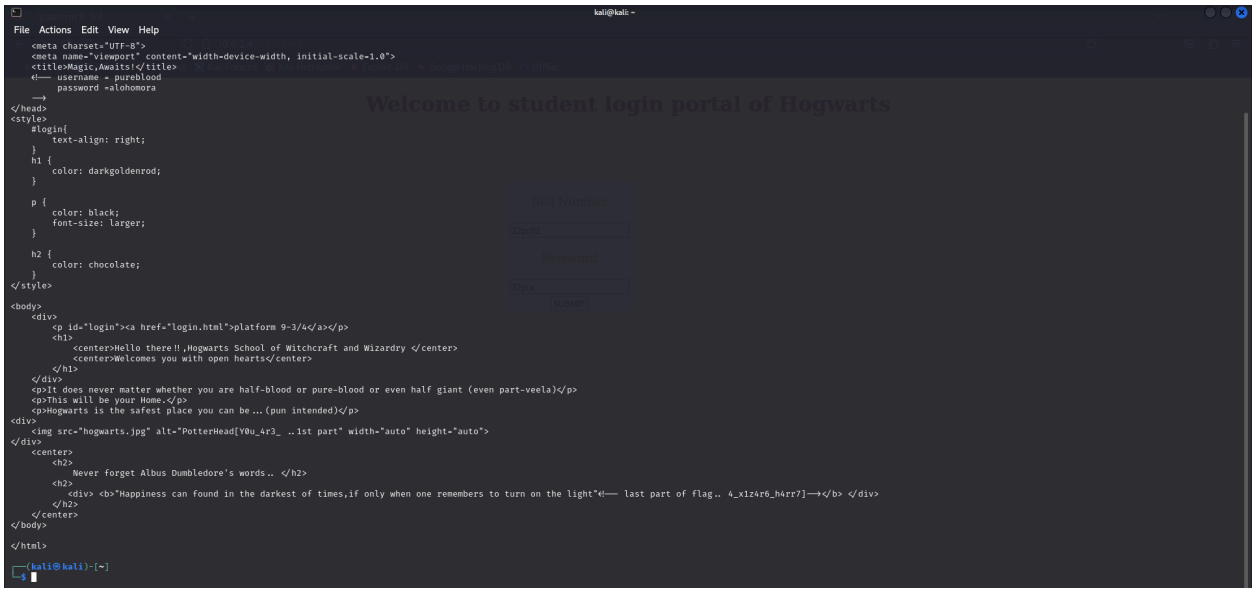
And submit random rollno and password



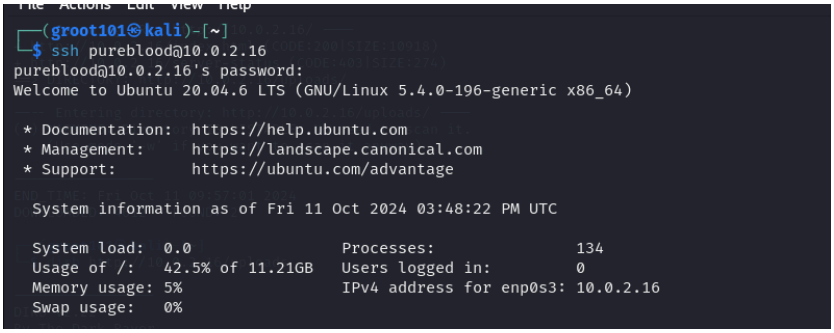
So we understand that special username password or cookie needs to be given.. So we use the welcome.txt hint “Lumos_Expelliamus” as cookie



With the obtained ssh credentials, access is gained by using curl 10.0.2.4 and the username and password are obtained



Username:pureblood
Password: alohomora



In the home folder, the draco folder is searched through. There are a few files found.

```
pureblood@hogwarts:/home$ ls
draco  pureblood
pureblood@hogwarts:/home$ cd draco
pureblood@hogwarts:/home/draco$ ls
conversations
pureblood@hogwarts:/home/draco$ cd conversations
pureblood@hogwarts:/home/draco/conversations$ ls -a
.  ..  ffthlzhzhragzh.zip  hint.txt  hiss.py  translate.py
pureblood@hogwarts:/home/draco/conversations$
```

A python file called hiss.py is found. Another zipped file called ffthlzhzhragzh.zip is present which is password protected.

The hint mentions that only parselmouths understand the conversation. The hint also suggests that the password lies in the conversation.
On opening the python file, there are certain comments and strings in the print statements that don't look normal.

```
    print("A piece of the puzzle falls into place...")

print("Harry's understanding of the Horcruxes grows.")
print("nzhzhragzhiizhnzhiizh. iizhtth mzhzhraktheezhssss uuzhssss vzhuuzhlzhnzheezhrzhzhraksslzheezh. wzheezh mzhuuzhssstth")
f print_harry_returns_to_life():
print("Harry Potter returns to the world of the living, his spirit and determination driving him back to the mortal realm, where he will fa
print("The warmth of life returns to Harry's body...")
time.sleep(1)  # pause for 1 second
print("He opens his eyes, ready to face whatever comes next.")

f print_voldemort_final_death():
print("Lord Voldemort is killed by his own Killing Curse, which rebounds back to him, a fitting end to a life of evil and tyranny, and a te
print("The curse turns back on Voldemort, striking him down...")
for i in range(5):
    #print("    mzhyizh ffthrzhiizheezhnzhdzh, mzhuuzhssstth sssstthzhrayizh tthlzhoozhsssseezh ")
```

On collecting such statements and arranging them, the conversation is found.

```
tthhsseezh kssoozhyizh hsszhrassss dzhiizhsssstthoozhvzheezhrzheezhdzh oozhuuzhrzh sssseezhtthrzheezhtth,
nzhzhragzhiizhnzhiizh. iizhtth mzhzhraktheezhssss uuzhssss vzhuuzhlzhnzheezhrzhzhraksslzheezh. wzheezh mzhuuzhssstth
dzheezhpthlzhoozhyizh zhralzhlzh oozhuuzhrzh ffthoozhrzhttheezhssss nzhoozhwzh tthoozh ffthiizhnzhdzh hssiizhmzh. zhranzhdzh yizhoozhuuzh,
mzhyizh ffthrzhiizheezhnzhdzh, mzhuuzhssstth sssstthzhrayizh tthlzhoozhsssseezh
```

There is also a function voldemort_talks_to_nagini() which prints the whole conversation when called.

```
def voldemort_talks_to_nagini():
    print('''tthhsseezh kssoozhyizh hsszhrassss dzhiizhsssstthoozhvzheezhrzheezhdzh oozhuuzhrzh sssseezhtthrzheezhtth,
nzhzhragzhiizhnzhiizh. iizhtth mzhzhraktheezhssss uuzhssss vzhuuzhlzhnzheezhrzhzhraksslzheezh. wzheezh mzhuuzhssstth
dzheezhpthlzhoozhyizh zhralzhlzh oozhuuzhrzh ffthoozhrzhttheezhssss nzhoozhwzh tthoozh ffthiizhnzhdzh hssiizhmzh. zhranzhdzh yizhoozhuuzh,
mzhyizh ffthrzhiizheezhnzhdzh, mzhuuzhssstth sssstthzhrayizh tthlzhoozhsssseezh''')
```

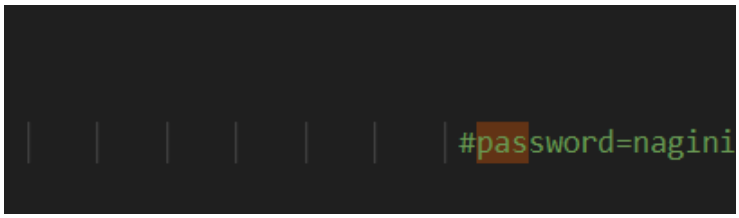
Also a translator file - translate.py is present which accepts language as the parameter and asks the language of the text that is needed to be converted. Parselmouths only understand Parseltongue.

On giving the obtained text as input to the program, the output is obtained as follows:

“che boy has discovered our secrec,
nagini. ic makes us vulnerable. we musc
deploy all our forces now co find him. and you,
my friend, musc scay close”

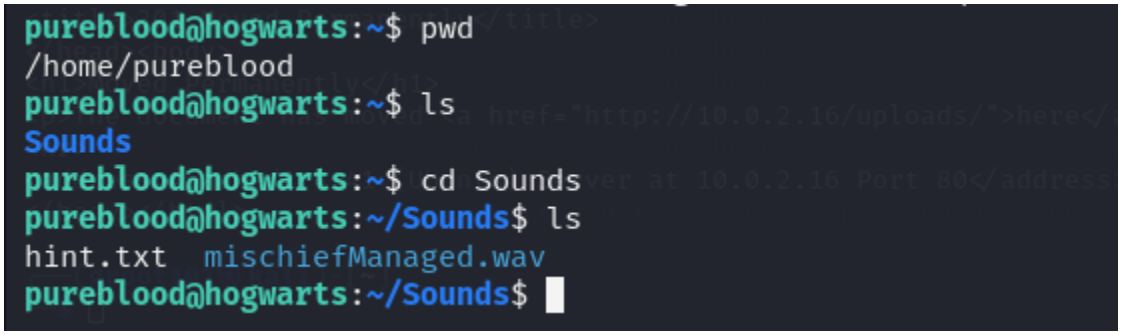
The password protected file - ffthlzhzhragzh.zip - turns out to be ‘flag’.(encoded in parseltongue).

Now the password is guessed as nagini as the file name is hiss.py.
Also a comment in the python file is found revealing the password.



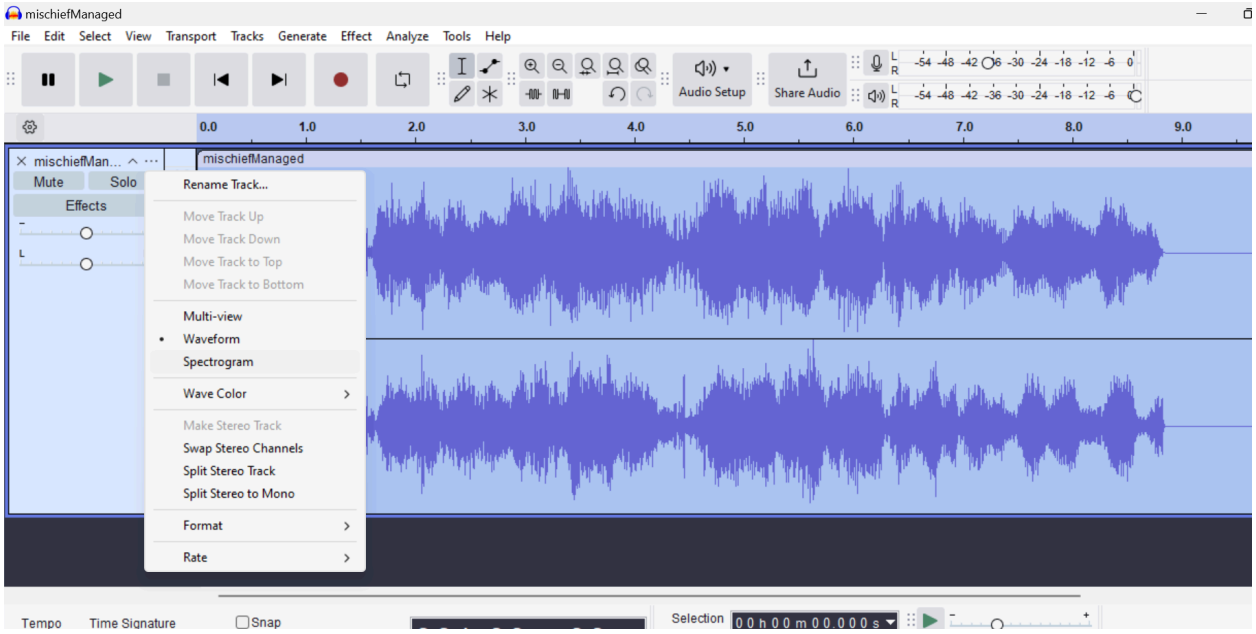
On opening the flag.txt file, a flag is found - **PotterHead[N@g1n1_th3_H0rC#ux]**.

A folder called Sounds is found navigating through the home folder. The files are transferred to the local machine - Kali VM.

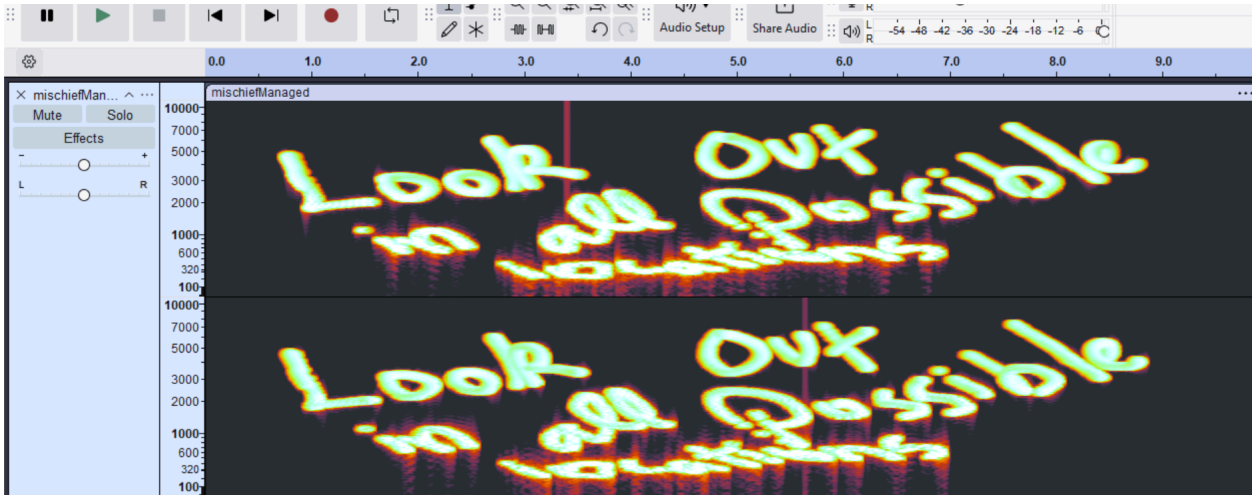


There is an audio file - mischiefManaged.wav - it runs for 8s.
But the audio is heard as a random noise.
The hint suggests that the sound isn’t audible as it isn’t in the ghosts’ SPECTRUM.

So a steganography technique called - Spectrogram Analysis can be tried.
For this the wav file is opened in Audacity.



The audio should be viewed as spectrograms.

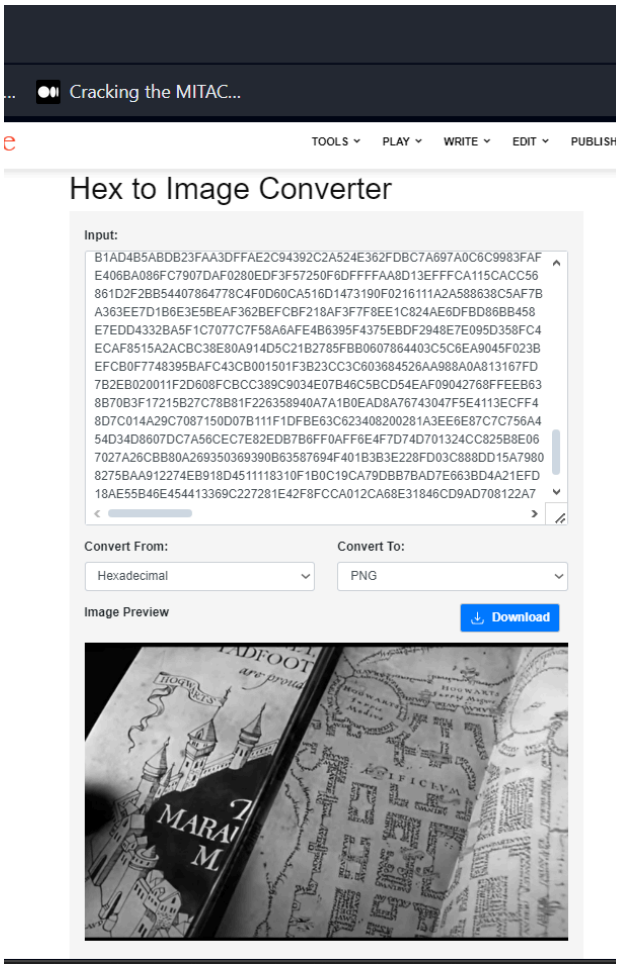


A clue - “Look out in all possible locations” - is obtained.
So the hidden files are explored and a text file- hexicallyDumpedImage.txt is found.
It appears to have some hexdump values as its name suggests.

hexicallyDumpedImage.txt

```
0A1A0A0000000D49484452000005280000033208000000004B67709300010000494441547 89CECFDD
0D97C0CC3D00C4D51144288240982C0710CE33CF8253C52525F86BDBBE770269A56499971F170375F
7FA945FF39FE73FCE7F8CFF17FEFF8B78325FEFB35E93FC77F8EFF1CFF7F7EFC3BA212DFFCF53FC77
03FE839BC91E83701D0C733EB19FFE681A3F61EBFA195EC2F88FD15DFBCEEB6B3FEEF1F7D38FC6668
40FC414079DB9677DBB693496F48F9E79386F7C19ABFF8E79BA7776DC0F76EFDBB66D070F29FD84FF
1E193E3493CFCF29B04C0F1C63BB07DF54E1B5EDFE9D5B35E29C5A1DDBF6920FB98104E3FBED24022
F2CEE593772EF6626CFBEFE9A03FE0B86DED1B92F8AB33F5DDE376D6BFF1E92F1FBF83D3FEDA5DDE1
1F88D78F06F791CE74767167FDA5429C3F4D6D8FBB1937F47A7FBFF69187080FFCAA0FFAB40F93B9E
37FD57A0F5771EBF6E8ABFA64FBF9F441CA7F9FF01F3FD5FA6187F82A4FD0B56D36FEBA1579E94370
DEE1D71FF1781F2F7CC821D287FF92AEE14A2B2CF12CC1968C4ADE9E1FFE70CF3D7FDF3BAB5AFC4A7
5F4E9B115078A07A7CD32747D05F0BE76EAAF99B2FD94FFB05FCBA912CEE9F542E7A18B0DF751CFBE
F7A4BF08E0D6FE71F6FFCB687E015FB1954657BFB3704CC7938B9DE65982FBF6461BF7938FA3BFA3B
1DE5DBEEFD75FE5B4FFB2DD38FE328D0FB98BE7E7EB9F788369D62733FD5398AD9A1DDC349BFA1556
333BD3DFC3DB83BDCE337C81D0FAFF5EA59B7C2F996C27DDBA2191BD15EBB0F078FCCED38C86FB6FE
E9555BDA25FF0250FE5FC7C9F2826498A7E065FE107C052C6FAB8D2108D4EFF6C633471ED7FE64D5E
1770042127037737712D8D6E41AA618E0859695A620A7B419400DAA6CB7A70030877761E1AFCDDBDE
79CB0F01DF36034314B87969114B537BCFD0D67533004E50047418DCE04E10100ADDCDBC0C4B57370
1DEA5AD19666EE6660094F0ECA42A72763046A5396839B94C41DC1D6079634FEE04DD9D14F19C0D20
37733A7C1CA71471980F6C224CC286BE3BDC68C0AD76CDDB3A6CF8897619C932890ECF44334065A4C
7EF2984E3EED38F9F66E42D1B4011DE8CE8A036BCBAF9D97A88238B2FAFEB839E297F7A9D69C35975
E52A0F3FFEE50C3338C5856ED90CA24144CCB66D4DEE1A7CFDFEF7270F9FFEF2E944334000BA79765
```

Using an online tool, the hex is converted to a grayscale image of The Marauders' Map.
Now in the hidden files, there is another python file called spellEncoded.py
In spellEncoded.py it is seen that a QR Code is hidden behind the image that was found.
Also the comments - bitplanes and lsb - give clues about the encoding techniques.




```python
from PIL import Image
import numpy as np

# Convert cover image to gray-scale
cover = Image.open("coverPhoto.png").convert('L

data_c = np.array(cover)

# Convert image to 1-bit pixel, black and white
secret = Image.open("qrFrame.png").convert('1')
secret = secret.resize(cover.size)

data_s = np.array(secret, dtype=np.uint8)

# Rewrite LSB
res = data_c & ~1 | data_s

new_img = Image.fromarray(res).convert("L")
new_img.save("cover-secret.png")
new_img.show()


#bitpplanesbitplanesbitplanes
#lsblsblsblsb
```

A script is written to extract the actual Least Significant Bits and the QR code is obtained as a PNG image.

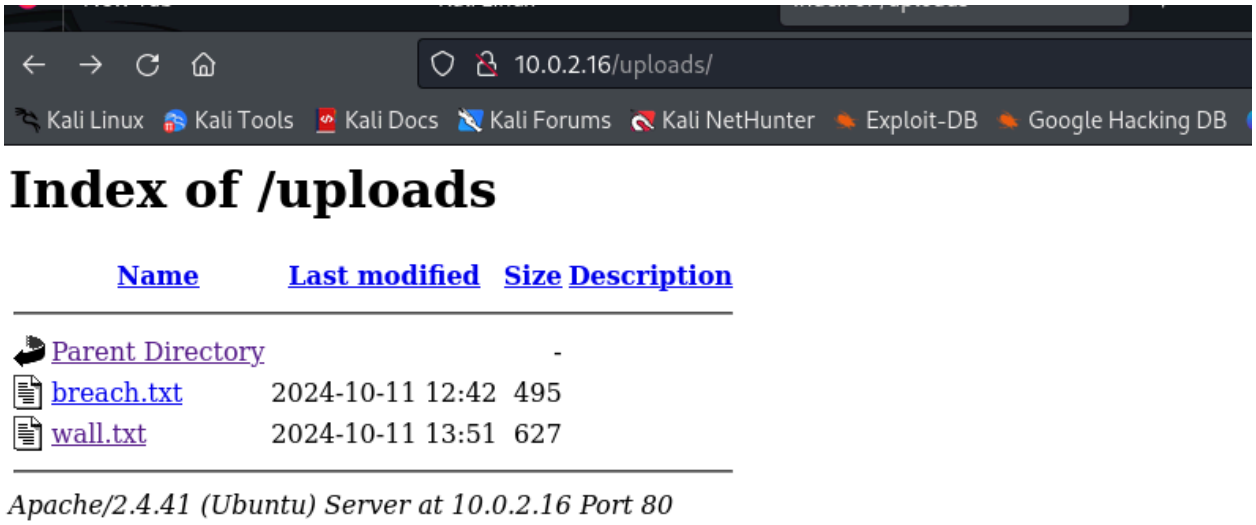
```python
from PIL import Image
import numpy as np

# Open the encoded image
encoded = Image.open("cover-secret.png").convert('L')
data_e = np.array(encoded)

# Extract the LSBs to reveal the hidden message
hidden_bits = data_e & 1

# Scale bits back to pixel values (0 or 255)
hidden_image = Image.fromarray(hidden_bits * 255).convert('1')

# Save and show the hidden image
hidden_image.save("decoded-secret.png")
hidden_image.show()
```

On scanning the QR, a flag is found.
PotterHead[I_S0lemnly%Sw3ar_I^am_Up!to#N0_G00d]

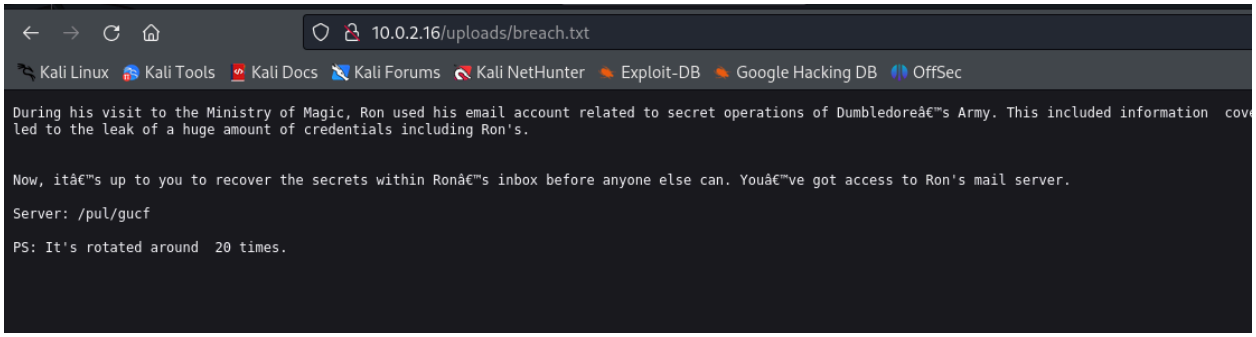
To look for possible hidden directories and other directories in the web server at the target machine,dirb is used. The subdirectory /uploads is found

```
┌──(groot101㉿kali)-[~]
└─$ dirb http://10.0.2.16

_____________
DIRB v2.22
By The Dark Raver
_____________

START_TIME: Fri Oct 11 09:56:54 2024
URL_BASE: http://10.0.2.16/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

_____________

GENERATED WORDS: 4612

---- Scanning URL: http://10.0.2.16/ ----
+ http://10.0.2.16/index.html (CODE:200|SIZE:10918)
+ http://10.0.2.16/server-status (CODE:403|SIZE:274)
==> DIRECTORY: http://10.0.2.16/uploads/

---- Entering directory: http://10.0.2.16/uploads/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

_____________

END_TIME: Fri Oct 11 09:57:01 2024
```

2 text files are found.

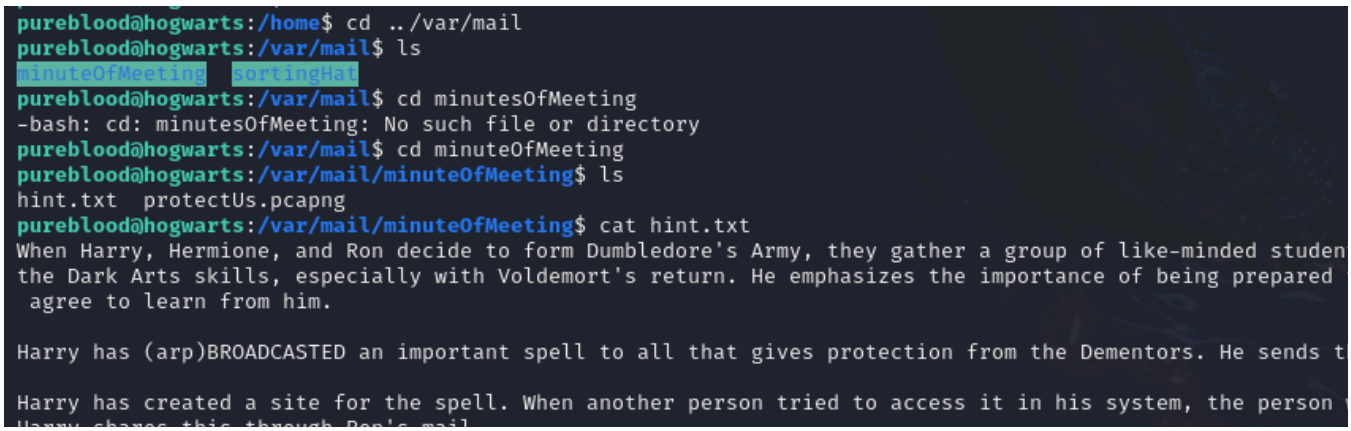


breach.txt turns out to be a clue for flags.



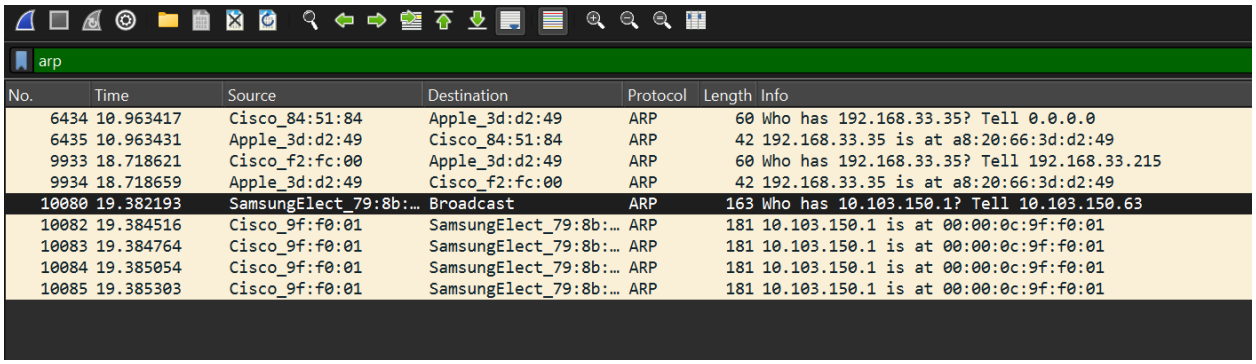
The server path is shifted 20 times. The actual path: /var/mail

On navigating to the folder, folders minuteOfMeeting and sortingHat are present.

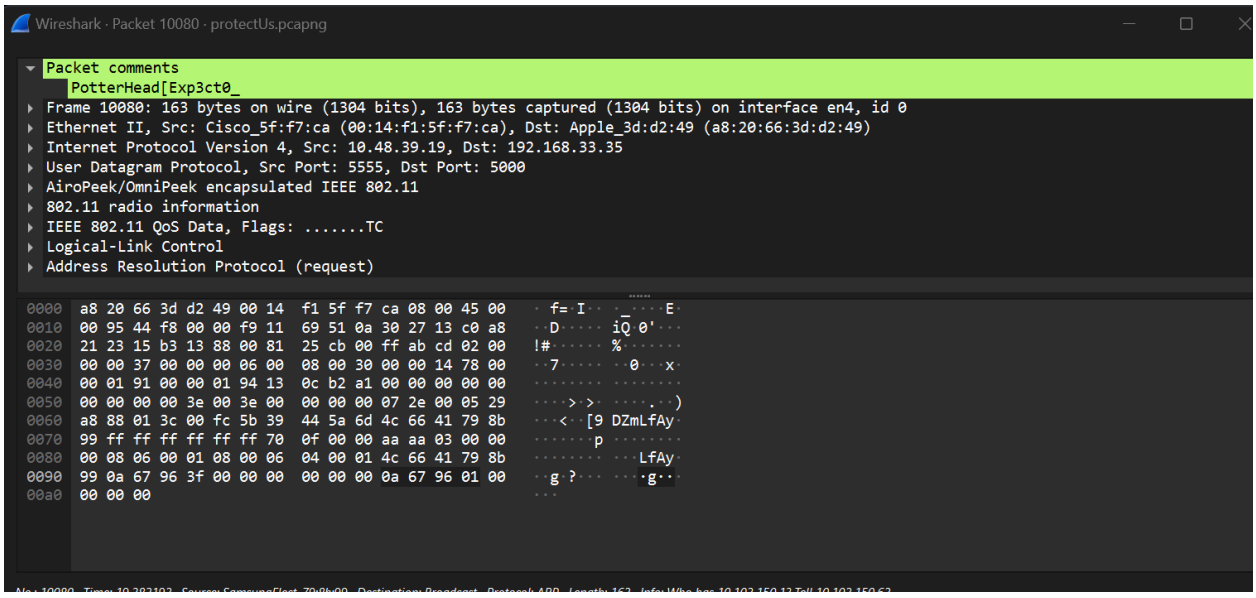


minuteOfMeeting contains a pcapng file.

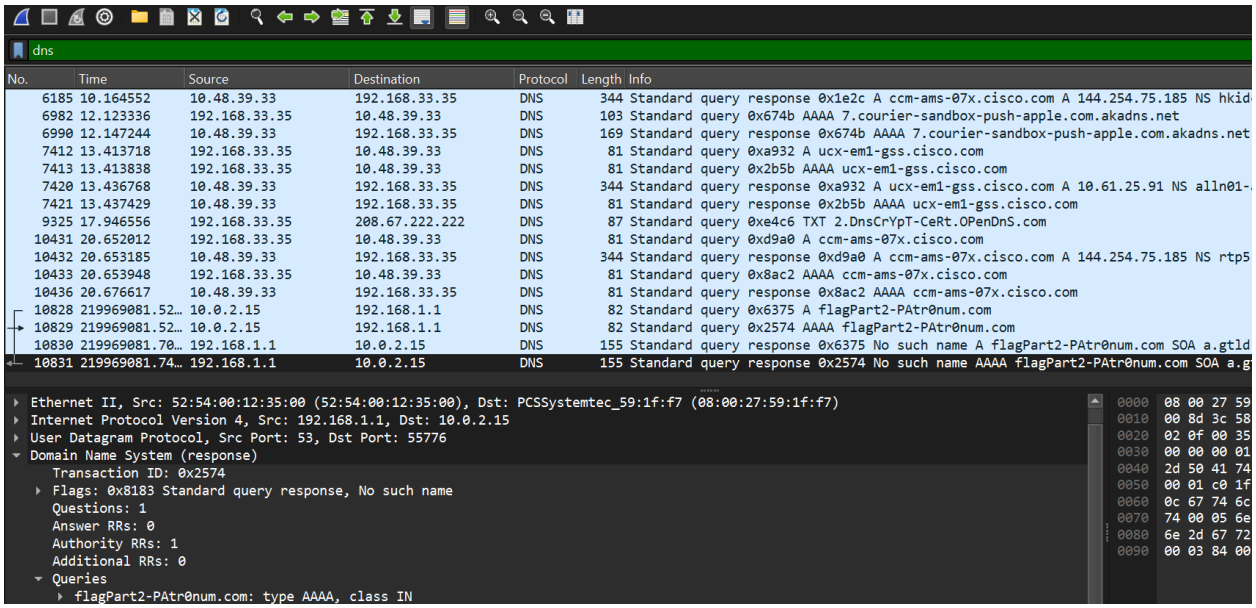
protectUs.pcapng is opened using WireShark. It has over 10000 packets.
hint.txt has the information that the spell is in 2 parts. So ,the flag is in 2 parts.
One of the clues in hint.txt is that the spell has been (arp)BROADCASTED.



Looking at the pcap file,there is only one ARP Broadcast packet.
On analyzing the packet,one half of a flag is found in the Packet Comments - PotterHead[Exp3ct0_



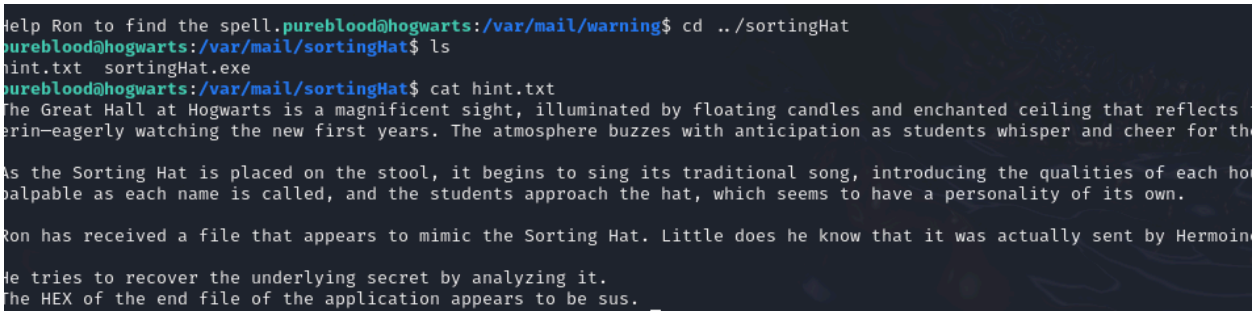
The next clue in hint.txt- The query to Harry's site was DSNied.
So DNS packets are checked which are 39 in number.Out of the 39 packets , only response packets containing deny messages are checked.



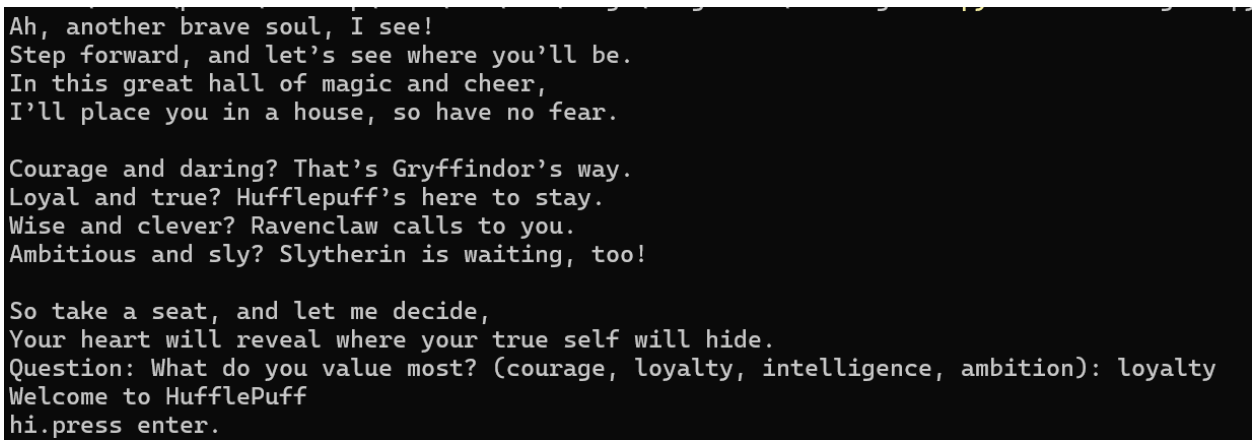
There is a DNS packet - query containing domain name as flagPart2-PAtr0num.com.
So,the second part of the spell is obtained.

Flag found: **PotterHead[Exp3ct0_P@tr0num]**

sortingHat contains an executable file.



On running the executable,



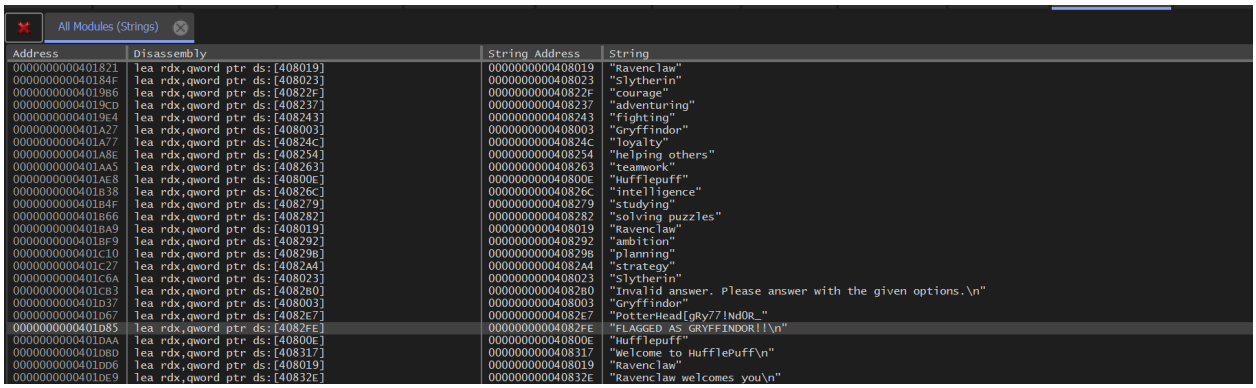
```
Question: What do you value most? (courage, loyalty, intelligence, ambition): ambition
Slytherin is proud to have you.
hi.press enter.
```

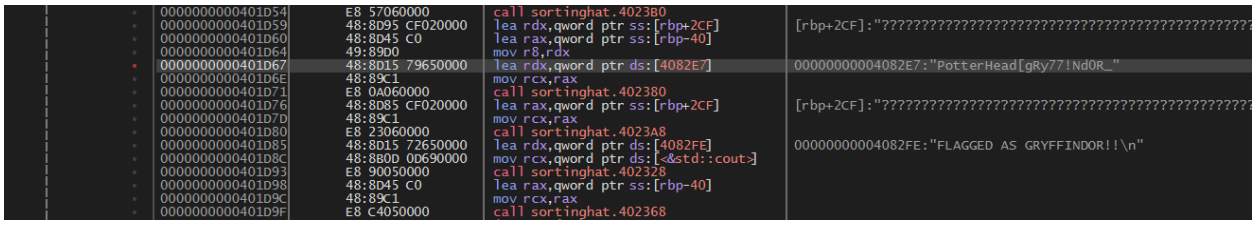
```
Question: What do you value most? (courage, loyalty, intelligence, ambition): intelligence
Ravenclaw welcomes you
hi.press enter.
```

```
Question: What do you value most? (courage, loyalty, intelligence, ambition): courage
FLAGGED AS GRYFFINDOR!!
hi.press enter.
```

As seen from the above images, each input gives a different output and the output for the input “courage” seems interesting.

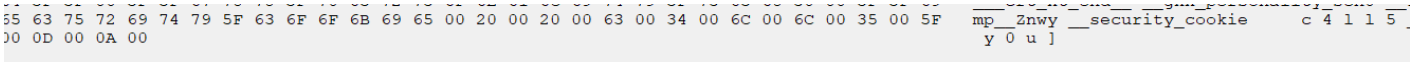
x64dbg is used to analyze the executable.
The references to the string “FLAGGED AS GRYFFINDOR” are searched for.





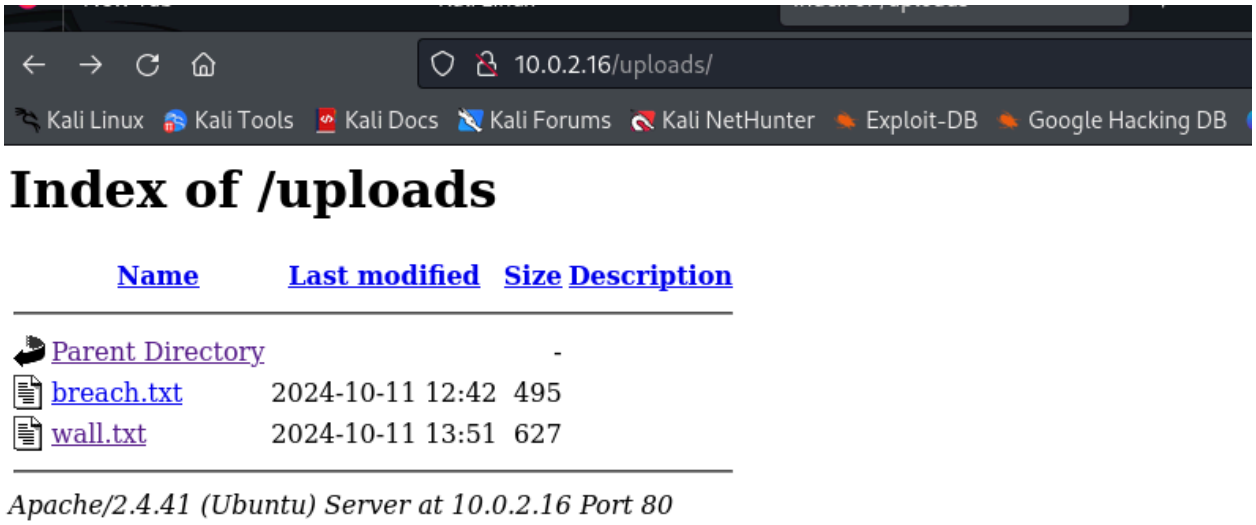
As seen in the above images, the flag’s first half - "PotterHead[gRy77!Nd0R_" is found in the string references.

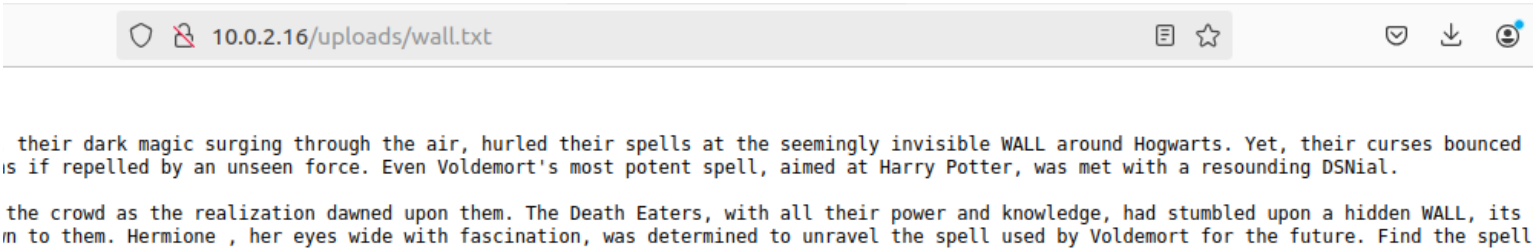
There is a clue in the hint.txt provided.
“The HEX of the end file of the application appears to be sus.”
On opening the binary file with a hex editor, the flag’s second part - c4ll5_y0u] of the flag is found.



Flag: **PotterHead[gRy77!Nd0R_c4ll5_y0u]**

In wall.txt, another clue is found





The hint suggests checking the firewall rules.
The flag: **PotterHead[4vad4_k3d4vra]** is found.