Noise Official Writeup

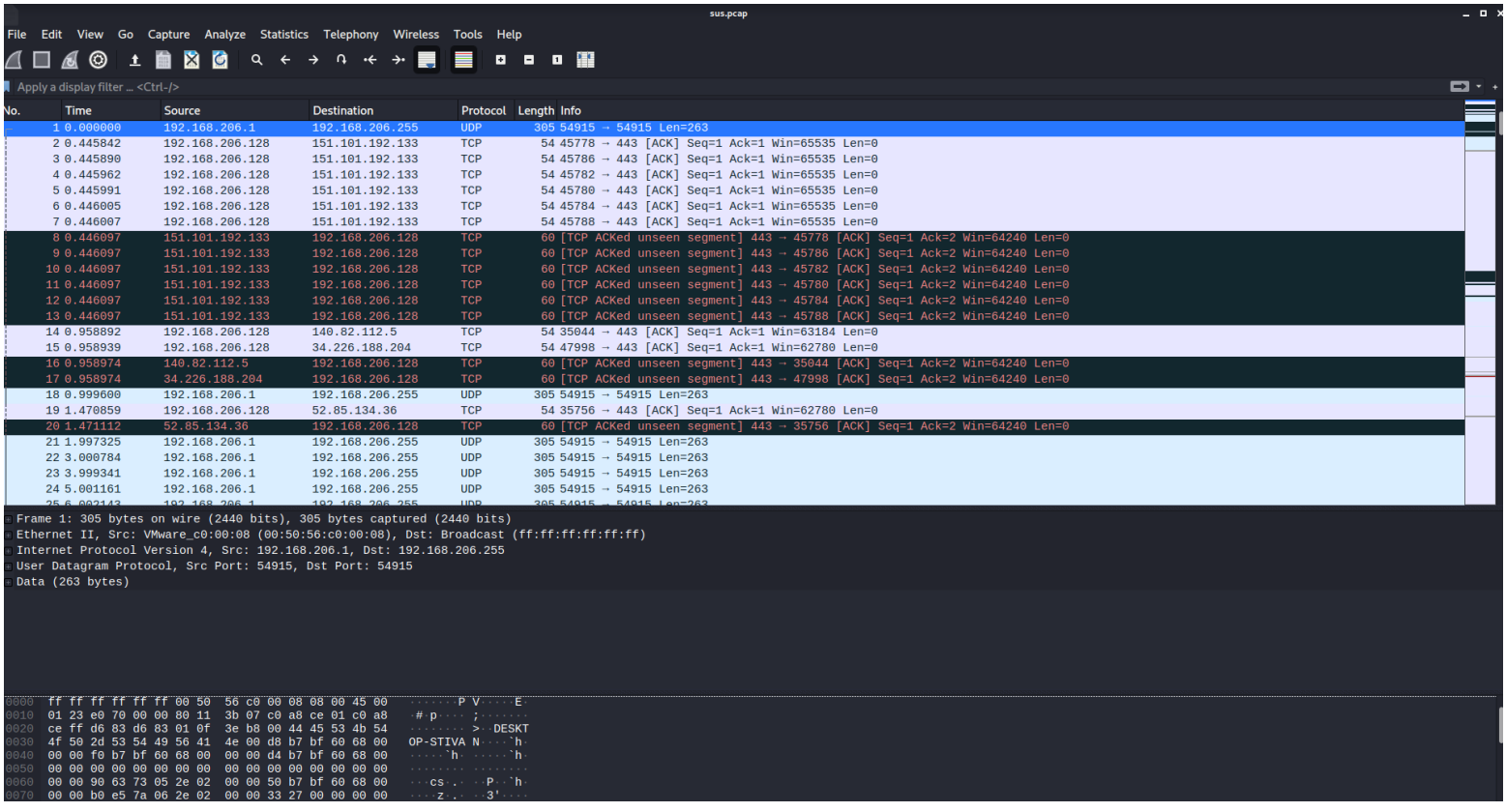
What is Noise room about?

- It's an example of a DNS Exfiltration Attack.

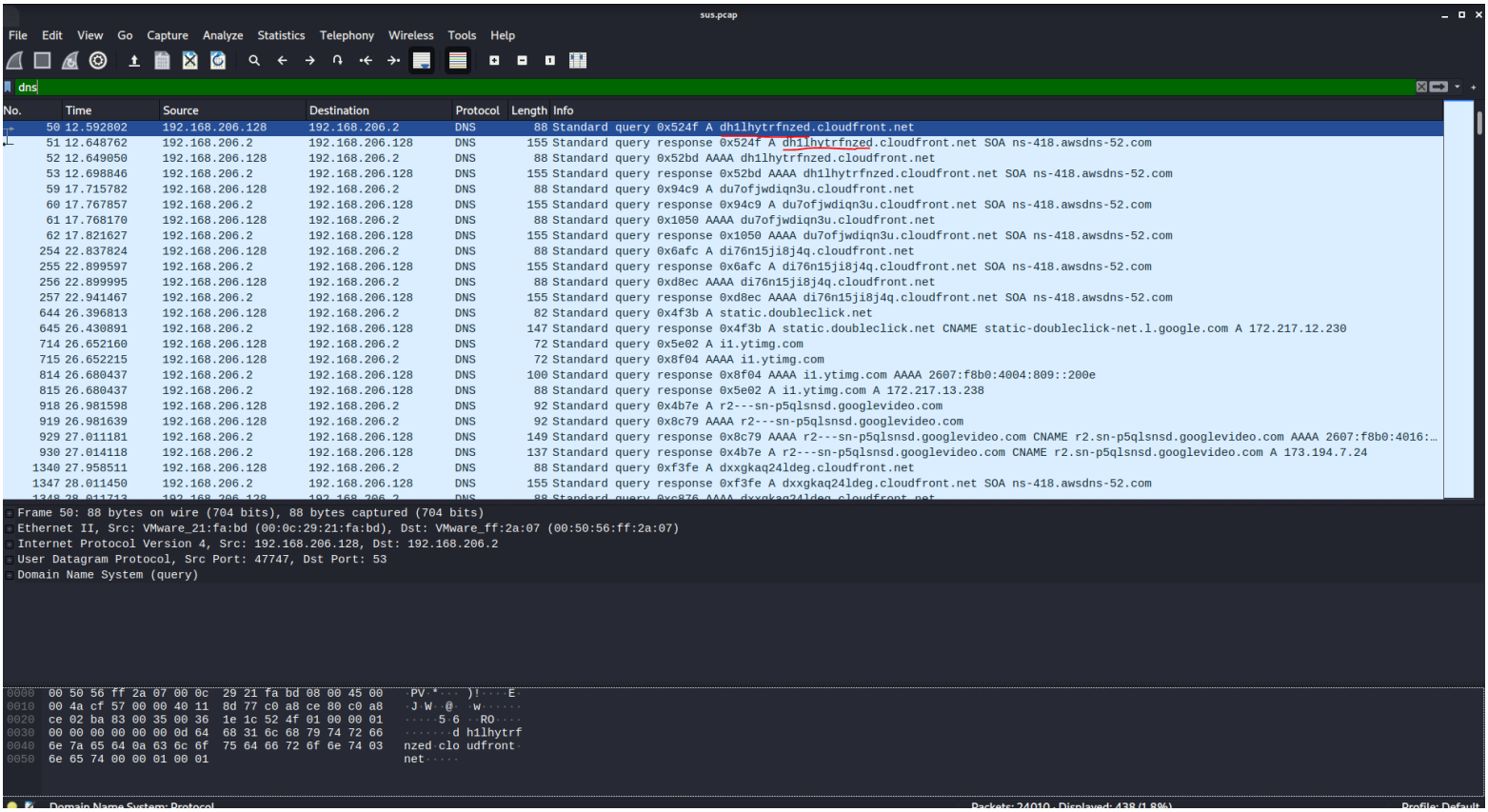
Walkthrough

At first, when we open the pcap file with wireshark we see that there is a ton of traffic on the network. A user must filter out the traffic in order to narrow down the DNS Exfiltration Attack. Once filtered the user must further narrow the traffic by detecting that there is a DNS Exfiltration Attack happening.

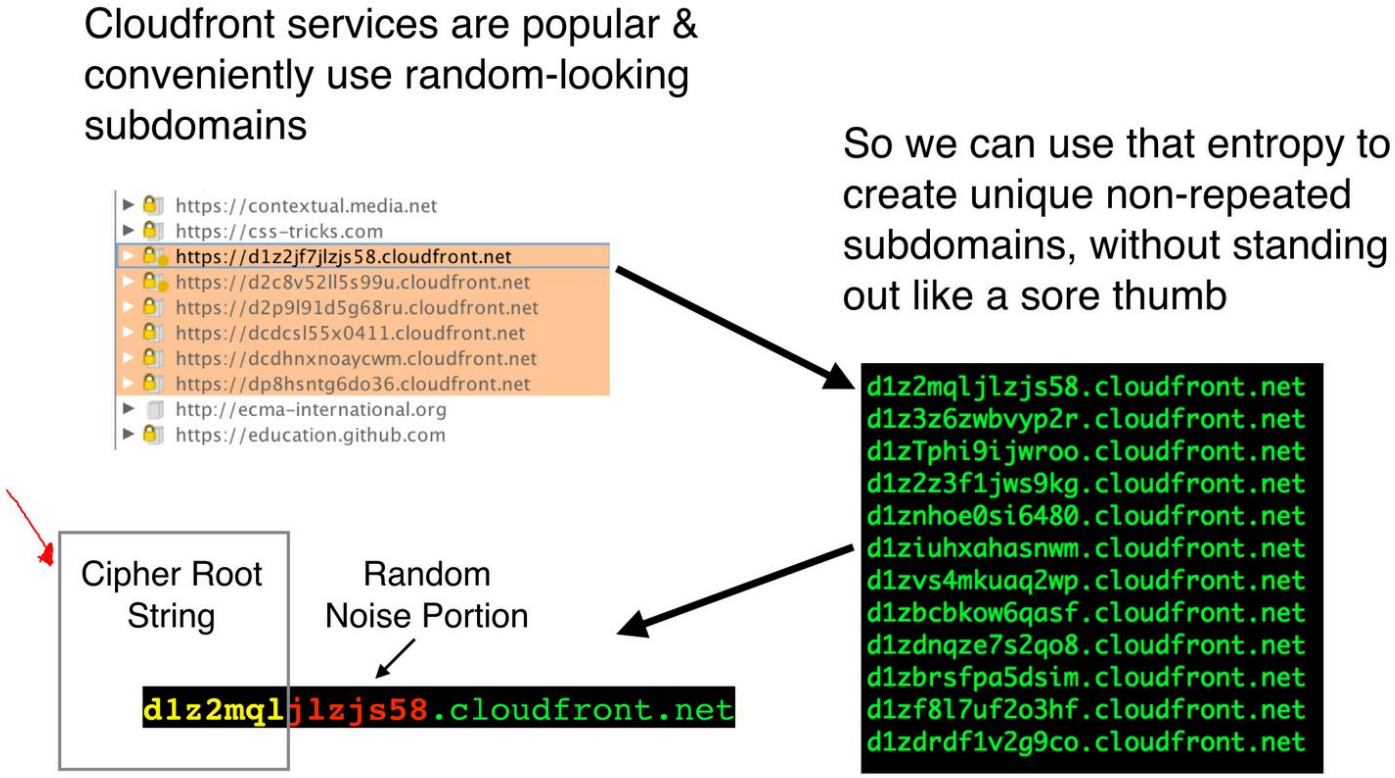
==========



Once filtered, we can clearly see the DNS Exfiltration Attack:

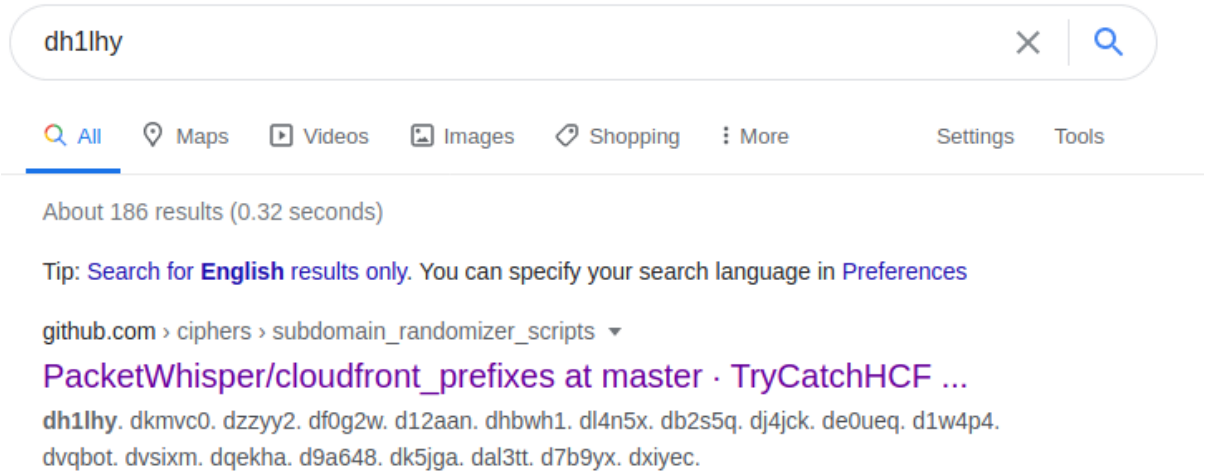


When we take a look at the DNS Query names, they just look random and make no sense. That is because that's how DNS Exfiltration works. It first encodes the data that needs to be exfiltrated and then is sent over DNS to the Attacker. Take a look at the following Image:



It appears that the example image's FQDN are kind of similar to the once in our pcap file.

As the image shows, the first 6 characters are the cipher root string and the rest are just random noise portion. So if the user google's any of the first 6 characters they will get this result:



Once opening the link there are redirected to a repo with cloudfront_prefixes

master · PacketWhisper / ciphers / subdomain_randomizer_scripts / **cloudfront_prefixes** · Go to file · ···

TryCatchHCF Add files via upload · Latest commit dbfce0f on Aug 27, 2018 · History

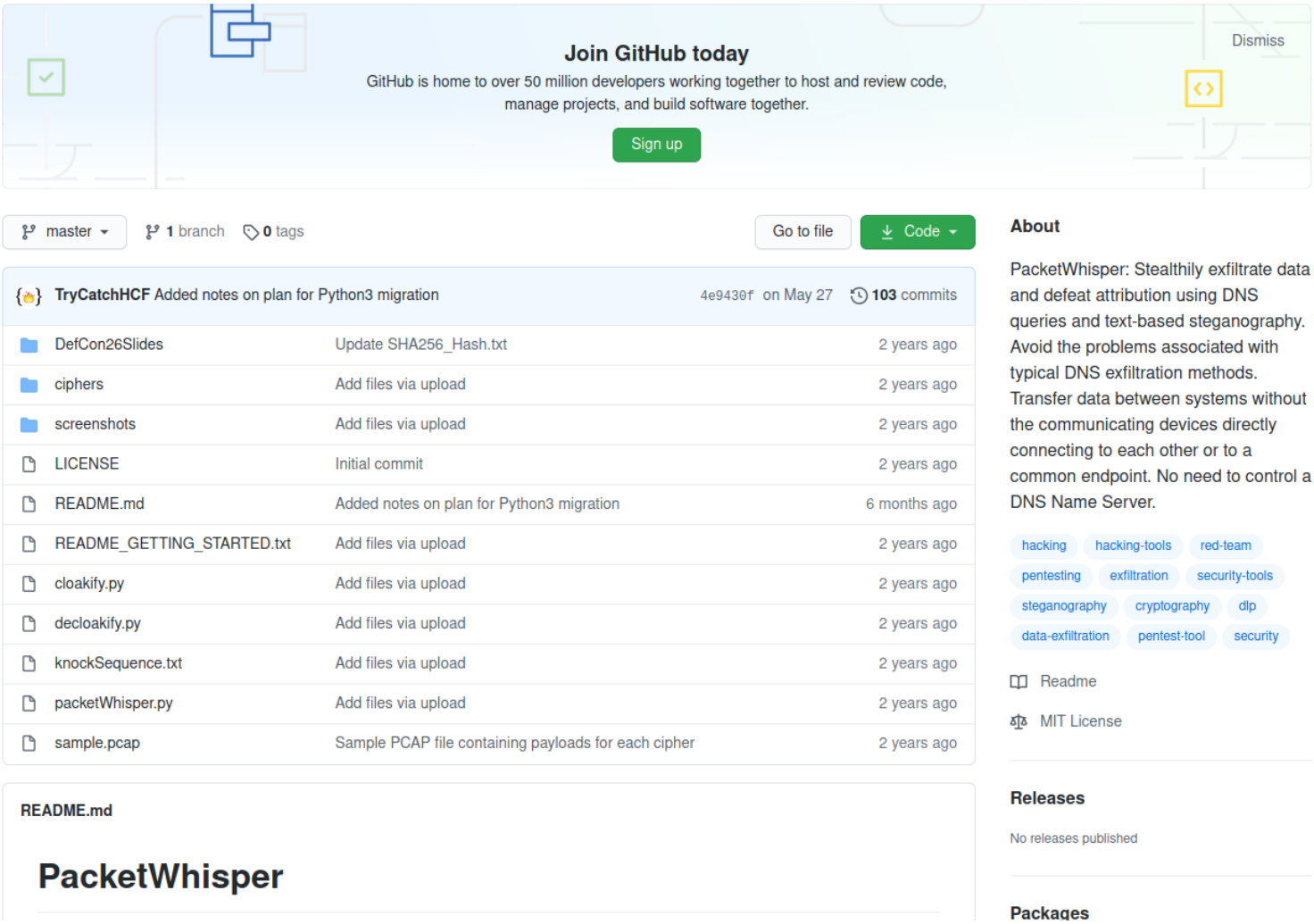
1 contributor

68 lines (68 sloc) | 476 Bytes · Raw · Blame

```
d5ip4p
doyyht
dzk09z
dbetnr
dcnjr5
d2frhd
dsi522
di76n1
dk58vf
dtmvzi
dt9as1
d7ofuo
do2wa6
dwvxiz
dsb0ky
du3puh
d1nf7d
dvsg9x
dynwyw
d01yhn
d2m9sk
dkx21q
dsumck
d3h22u
dxe9cq
dymbcl
dnd4y0
dimoa1
dh1sej
dgkc2p
du7ofj
d9rdxz
dbv4vg
dp2hkw
```

The list is the cipher root string. So each DNS query name in our pcap file with the cloudfront.net name, the first 6 characters will be matched to the list of cipher root string.

Now we navigate to the repo page:

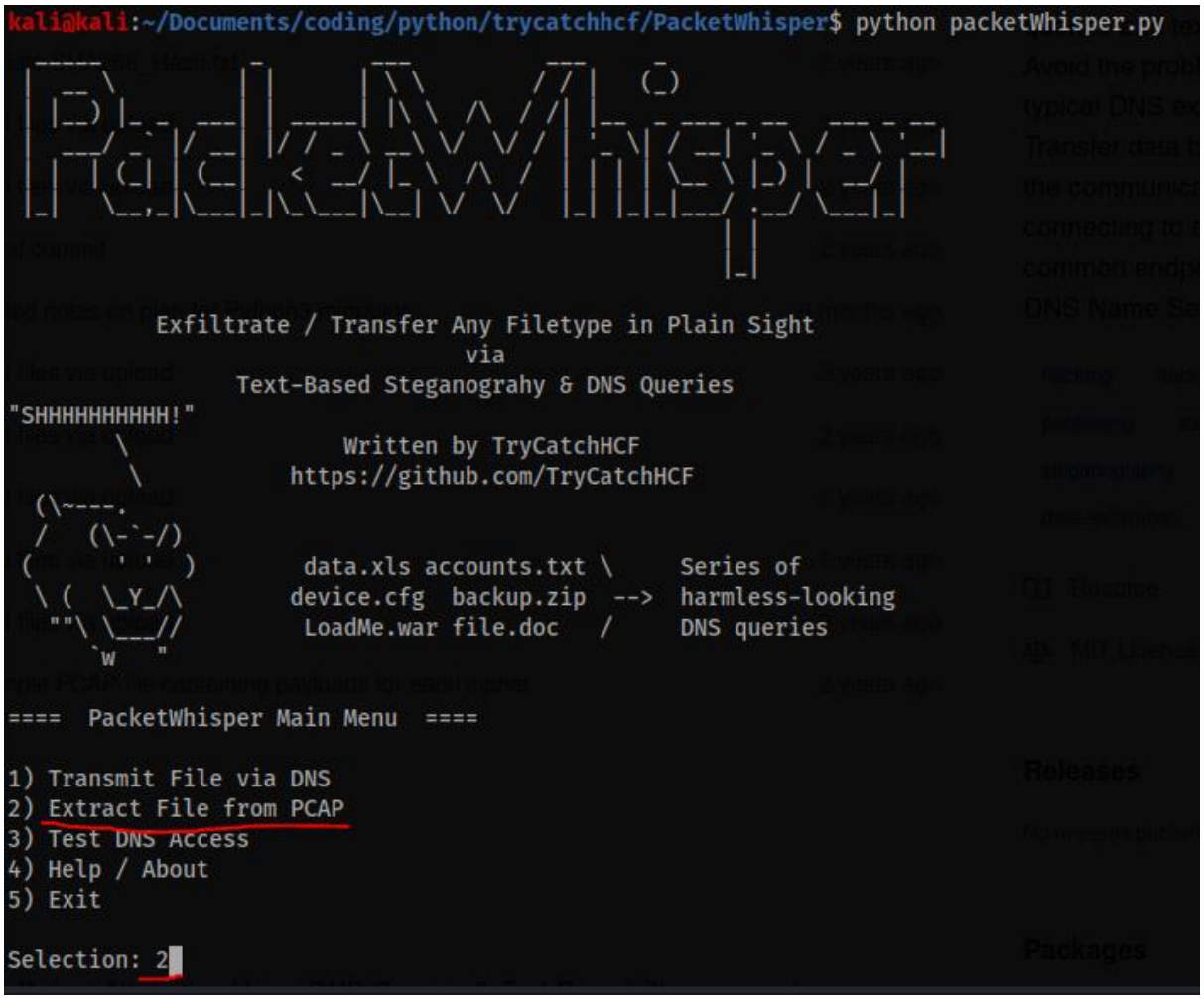


Once there, we use git clone on own local machine to get the files needed to decode the cipher.

We make sure that our pcap file is in the same folder as the git clone folder.

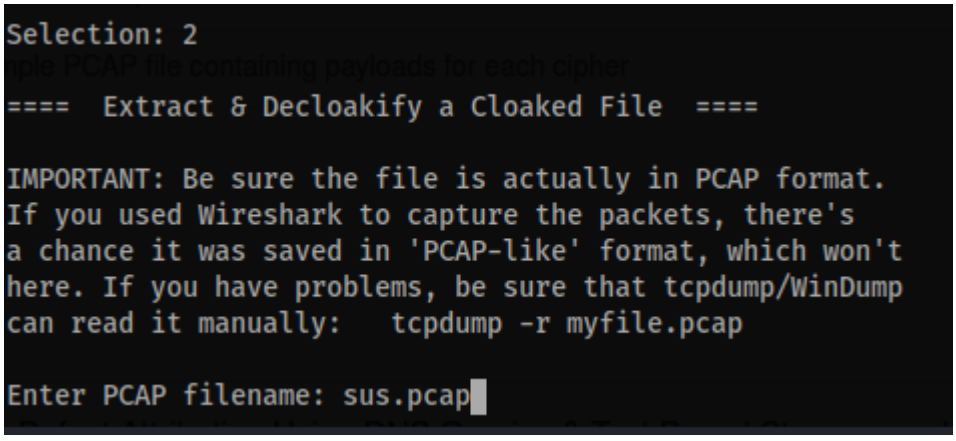
```
kali@kali:~/Documents/coding/python/trycatchhcf/PacketWhisper$ ls
ciphers          cloakify.pyc   decloakify.pyc  dnsQueries.txt  knockSequence.txt  README_GETTING_STARTED.txt  screenshots  sus.txt
cloaked.payload  decloaked      DefCon26Slides  dump6.pcap      LICENSE            README.md                   susor.pcap   tempFQDNList.txt
cloakify.py      decloakify.py  demothm.txt     flag.txt        packetWhisper.py   sample.pcap                 sus.pcap
```

Now we run it by typing, python packetWhisper.py



We select option 2, since we want to extract file from PCAP.

Next options asks us for the name of our pcap file:



Next it asks us about the OS that we are using.

```
Enter PCAP filename: sus.pcap

What OS are you currently running on?

1) Linux/Unix/MacOS
2) Windows

Select OS [1 or 2]: 1
```

Next, it asks about the cipher that has been used to transfer the data. In this case it is option 1, you can see the example given, it follows the syntax in our pcap file:

```
Select OS [1 or 2]: 1
reading from file sus.pcap, link-type EN10MB (Ethernet)

=======  Select PacketWhisper Cipher Used For Transfer  =======

1) Random Subdomain FQDNs  (example: d1z2mqljlzjs58.cloudfront.net)
2) Unique Repeating FQDNs  (example: John.Whorfin.yoyodyne.com)
3) [DISABLED] Common Website FQDNs    (example: www.youtube.com)

Selection: 1
```

*Next, which ciphers are being used, Option 3. It is the list with cipher root string*s:

```
Selection: 1

Ciphers:

1 - akstat_io_prefixes
2 - cdn_optimizely_prefixes
3 - cloudfront_prefixes
4 - log_optimizely_prefixes

Enter cipher #: 3
```

Finally, it asks about what kind of name should the output be.

```
Enter cipher #: 3

Extracting payload from PCAP using cipher: ciphers/subdomain_randomizer_scripts/cloudfront_prefixes

Save decloaked data to filename (default: 'decloaked.file'): sample.txt
```

Now we can exit the application

```
Extracting payload from PCAP using cipher: ciphers/subdomain_randomizer_scripts/cloudfront_prefixes

Save decloaked data to filename (default: 'decloaked.file'): sample.txt

File 'cloaked.payload' decloaked and saved to 'sample.txt'

Press return to continue...

====  PacketWhisper Main Menu  ====

1) Transmit File via DNS
2) Extract File from PCAP
3) Test DNS Access
4) Help / About
5) Exit

Selection: 5
```

We should have our output now, in my case, I named my output to be sample.txt, if I navigate to it and use cat to see the content of the file I should see the flag needed to pass the room.

```
kali@kali:~/Documents/coding/python/trycatchhcf/PacketWhisper$ ls
ciphers          cloakify.pyc   decloakify.pyc  dnsQueries.txt  knockSequence.txt  README_GETTING_STARTED.txt  sample.txt   sus.pcap
cloaked.payload  decloaked      DefCon26Slides  dump6.pcap      LICENSE            README.md                   screenshots  sus.txt
cloakify.py      decloakify.py  demothm.txt     flag.txt        packetWhisper.py   sample.pcap                 susor.pcap   tempFQDNList.txt
kali@kali:~/Documents/coding/python/trycatchhcf/PacketWhisper$ cat sample.txt
THM{DnsE$saltxFilTRA$salttion}
kali@kali:~/Documents/coding/python/trycatchhcf/PacketWhisper$
```

```
kali@kali:~/Documents/coding/python/trycatchhcf/PacketWhisper$ ls
ciphers          cloakify.pyc   decloakify.pyc  dnsQueries.txt  knockSequence.txt  README_GETTING_STARTED.txt  sample.txt   sus.pcap
cloaked.payload  decloaked      DefCon26Slides  dump6.pcap      LICENSE            README.md                   screenshots  sus.txt
cloakify.py      decloakify.py  demothm.txt     flag.txt        packetWhisper.py   sample.pcap                 susor.pcap   tempFQDNList.txt
kali@kali:~/Documents/coding/python/trycatchhcf/PacketWhisper$ cat sample.txt
THM{DnsE[redacted]xFilTRA[redacted]tion}
kali@kali:~/Documents/coding/python/trycatchhcf/PacketWhisper$
```