

Noise Official Writeup

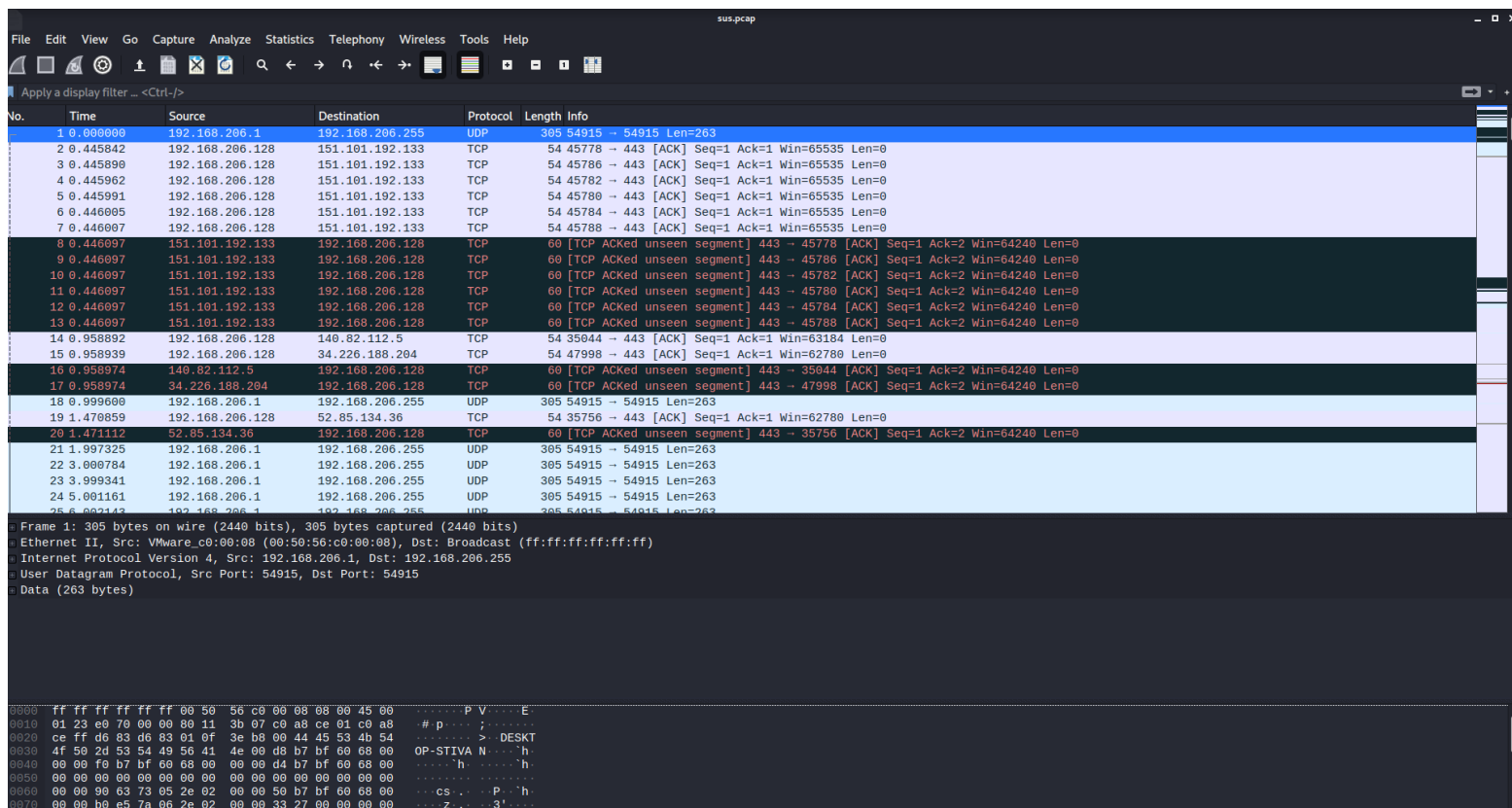
What is Noise room about?

- 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:

The image shows a Wireshark packet capture of a DNS query. The packet list shows a query for a CloudFront domain. The packet details show the query name and the packet bytes.

No.	Time	Source	Destination	Protocol	Length	Info
50	12.592802	192.168.206.128	192.168.206.2	DNS	88	Standard query 0x524f A dh1hytrfnzed.cloudfront.net
51	12.648762	192.168.206.2	192.168.206.128	DNS	155	Standard query response 0x524f A dh1hytrfnzed.cloudfront.net SOA ns-418.awsdns-52.com
52	12.649850	192.168.206.128	192.168.206.2	DNS	88	Standard query 0x52bd AAAA dh1hytrfnzed.cloudfront.net
53	12.698846	192.168.206.2	192.168.206.128	DNS	155	Standard query response 0x52bd AAAA dh1hytrfnzed.cloudfront.net SOA ns-418.awsdns-52.com
59	17.715782	192.168.206.128	192.168.206.2	DNS	88	Standard query 0x94c9 A du7ofjwdiqn3u.cloudfront.net
60	17.767857	192.168.206.2	192.168.206.128	DNS	155	Standard query response 0x94c9 A du7ofjwdiqn3u.cloudfront.net SOA ns-418.awsdns-52.com
61	17.768176	192.168.206.128	192.168.206.2	DNS	88	Standard query 0x1050 AAAA du7ofjwdiqn3u.cloudfront.net
62	17.821627	192.168.206.2	192.168.206.128	DNS	155	Standard query response 0x1050 AAAA du7ofjwdiqn3u.cloudfront.net SOA ns-418.awsdns-52.com
254	22.837824	192.168.206.128	192.168.206.2	DNS	88	Standard query 0x6afc A di76n15ji8j4q.cloudfront.net
255	22.895957	192.168.206.2	192.168.206.128	DNS	155	Standard query response 0x6afc A di76n15ji8j4q.cloudfront.net SOA ns-418.awsdns-52.com
256	22.899995	192.168.206.128	192.168.206.2	DNS	88	Standard query 0xd8ec AAAA di76n15ji8j4q.cloudfront.net
257	22.941467	192.168.206.2	192.168.206.128	DNS	155	Standard query response 0xd8ec AAAA di76n15ji8j4q.cloudfront.net SOA ns-418.awsdns-52.com
644	26.396813	192.168.206.128	192.168.206.2	DNS	82	Standard query 0x4f3b A static.doubleclick.net
645	26.430891	192.168.206.2	192.168.206.128	DNS	147	Standard query response 0x4f3b A static.doubleclick.net CNAME static-doubleclick-net.l.google.com A 172.217.12.230
714	26.652160	192.168.206.128	192.168.206.2	DNS	72	Standard query 0x5e02 A i1.ytimg.com
715	26.652215	192.168.206.2	192.168.206.128	DNS	72	Standard query 0x8f04 AAAA i1.ytimg.com
814	26.680437	192.168.206.2	192.168.206.128	DNS	100	Standard query response 0x8f04 AAAA i1.ytimg.com AAAA 2607:f8b0:4004:809::200e
815	26.680437	192.168.206.2	192.168.206.128	DNS	88	Standard query response 0x5e02 A i1.ytimg.com A 172.217.13.238
918	26.981598	192.168.206.128	192.168.206.2	DNS	92	Standard query 0x4b7e A r2---sn-p5qlnsd.googlevideo.com
919	26.981639	192.168.206.128	192.168.206.2	DNS	92	Standard query 0x8c79 AAAA r2---sn-p5qlnsd.googlevideo.com
929	27.011181	192.168.206.2	192.168.206.128	DNS	149	Standard query response 0x8c79 AAAA r2---sn-p5qlnsd.googlevideo.com CNAME r2.sn-p5qlnsd.googlevideo.com AAAA 2607:f8b0:4016::
930	27.011181	192.168.206.2	192.168.206.128	DNS	137	Standard query response 0x4b7e A r2---sn-p5qlnsd.googlevideo.com CNAME r2.sn-p5qlnsd.googlevideo.com A 173.194.7.24
1340	27.958511	192.168.206.128	192.168.206.2	DNS	88	Standard query 0xf3fe A dxgkq24ldq.cloudfront.net
1347	28.011450	192.168.206.2	192.168.206.128	DNS	155	Standard query response 0xf3fe A dxgkq24ldq.cloudfront.net SOA ns-418.awsdns-52.com
1348	28.011712	192.168.206.128	192.168.206.2	DNS	88	Standard query 0xc876 AAAA dxgkq24ldq.cloudfront.net

Frame 50: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface 0
Ethernet II, Src: VMware_21:fa:bd (00:0c:29:21:fa:bd), Dst: VMware_ff:2a:07 (00:50:56:ff:2a:07)
Internet Protocol Version 4, Src: 192.168.206.128, Dst: 192.168.206.2
User Datagram Protocol, Src Port: 47747, Dst Port: 53
Domain Name System (query)
Standard query query 0x524f A dh1hytrfnzed.cloudfront.net

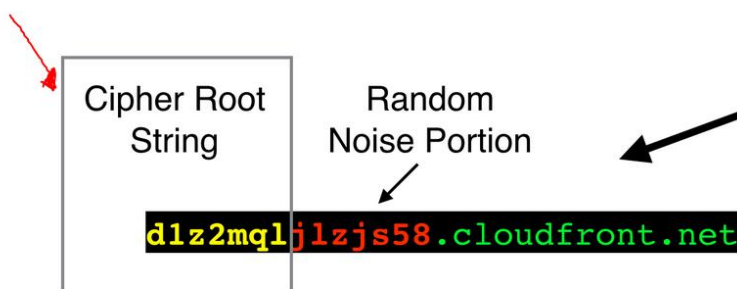
0000 00 50 56 ff 2a 07 00 0c 29 21 fa bd 08 00 45 00 PV *...)I...E
0010 00 4a cf 57 00 00 40 11 8d 77 c0 a8 ce 80 c0 a8 J.W.@..w....
0020 ce 02 ba 83 09 35 00 36 1e 1c 52 4f 01 00 00 015.6...RO...
0030 00 00 00 00 00 00 00 64 60 31 0c 60 70 74 72 66d h1hytrf
0040 6e 7a 65 64 8a 63 6c 6f 75 64 6e 72 6f 6e 74 63 nzed.clo udfront
0050 6e 65 74 00 00 01 00 01 net

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:

Cloudfront services are popular & conveniently use random-looking subdomains

So we can use that entropy to create unique non-repeated subdomains, without standing out like a sore thumb

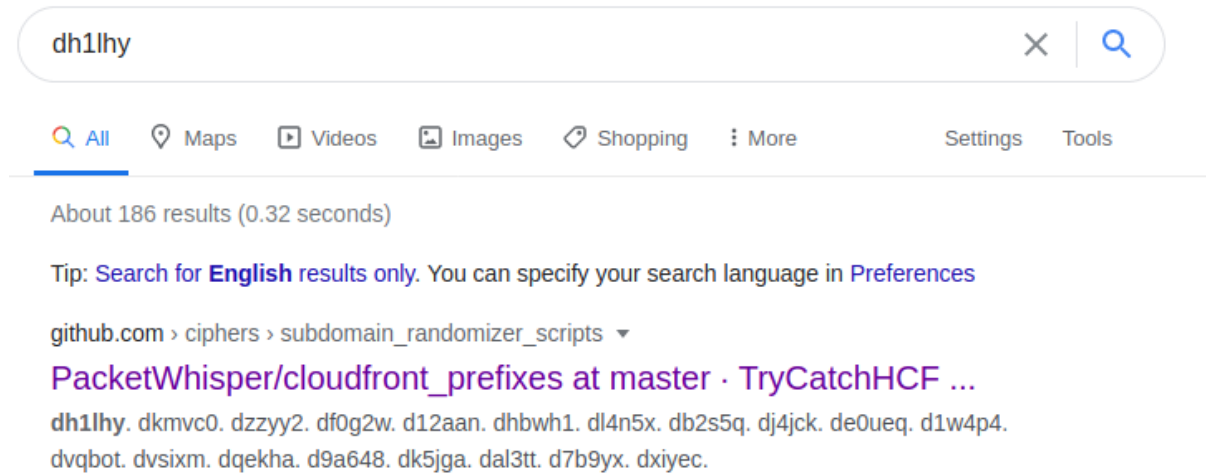
- https://contextual.media.net
- https://css-tricks.com
- https://d1z2fj1zjs58.cloudfront.net
- https://d2c8v52i15s99u.cloudfront.net
- https://d2p9i91d5g68ru.cloudfront.net
- https://dcdsl55x0411.cloudfront.net
- https://dcdhnxnoaycwm.cloudfront.net
- https://dp8hsntg6do36.cloudfront.net
- http://ecma-international.org
- https://education.github.com



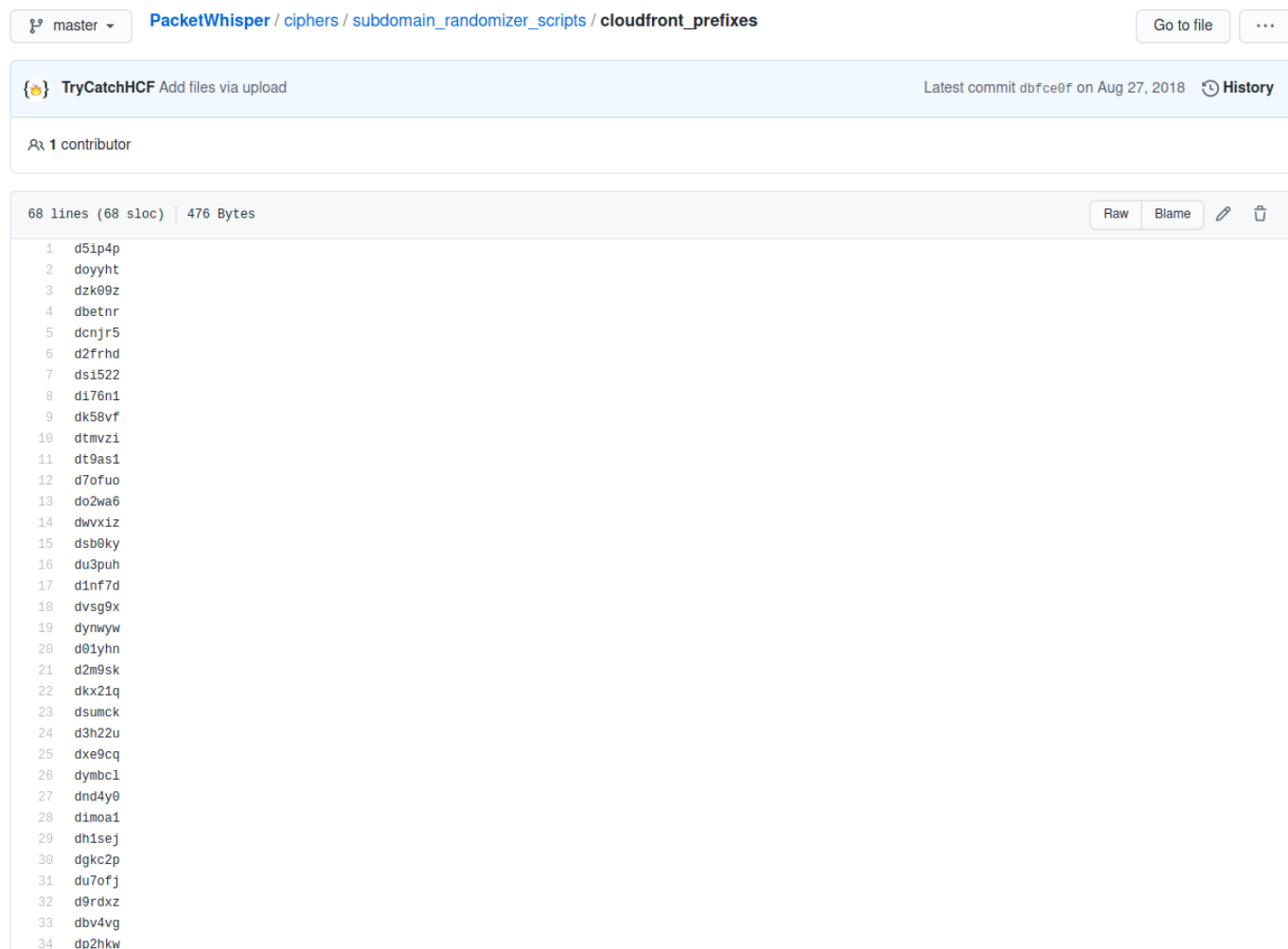
d1z2mqlj1zjs58.cloudfront.net
d1z3z6zwbvyp2r.cloudfront.net
d1zTphi9ijwroo.cloudfront.net
d1z2z3f1jws9kg.cloudfront.net
d1znhoes0si6480.cloudfront.net
d1ziuhxahasnwm.cloudfront.net
d1zvs4mkuaq2wp.cloudfront.net
d1zbcbkow6qasf.cloudfront.net
d1zdnqze7s2qo8.cloudfront.net
d1zbrsfpa5dsim.cloudfront.net
d1zf8l7uf2o3hf.cloudfront.net
d1zdrdf1v2g9co.cloudfront.net

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



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.

Join GitHub today

GitHub is home to over 50 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

master 1 branch 0 tags

Go to file

Code

TryCatchHCF Added notes on plan for Python3 migration 4e9430f on May 27 103 commits

DefCon26Slides	Update SHA256_Hash.txt	2 years ago
ciphers	Add files via upload	2 years ago
screenshots	Add files via upload	2 years ago
LICENSE	Initial commit	2 years ago
README.md	Added notes on plan for Python3 migration	6 months ago
README_GETTING_STARTED.txt	Add files via upload	2 years ago
cloakify.py	Add files via upload	2 years ago
decloakify.py	Add files via upload	2 years ago
knockSequence.txt	Add files via upload	2 years ago
packetWhisper.py	Add files via upload	2 years ago
sample.pcap	Sample PCAP file containing payloads for each cipher	2 years ago

README.md

PacketWhisper

About

PacketWhisper: Stealthily exfiltrate data and defeat attribution using DNS queries and text-based steganography. Avoid the problems associated with typical DNS exfiltration methods. Transfer data between systems without the communicating devices directly connecting to each other or to a common endpoint. No need to control a DNS Name Server.

hacking

hacking-tools

red-team

pentesting

exfiltration

security-tools

steganography

cryptography

dip

data-exfiltration

pentest-tool

security

Readme

MIT License

Releases

No releases published

Packages

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`

```
kali@kali:~/Documents/coding/python/trycatchhcf/PacketWhisper$ python packetWhisper.py

PacketWhisper (C)

Exfiltrate / Transfer Any Filetype in Plain Sight
via
Text-Based Steganography & DNS Queries

Written by TryCatchHCF
https://github.com/TryCatchHCF

data.xls accounts.txt \
device.cfg backup.zip --> harmless-looking
LoadMe.war file.doc /   DNS queries

==== PacketWhisper Main Menu ====

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

Selection: 2
```

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

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

```
Selection: 2
Please PCAP file containing payloads for each cipher

==== 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
work here. If you have problems, be sure that tcpdump/WinDump
can read it manually: tcpdump -r myfile.pcap

Enter PCAP filename: sus.pcap
```

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 strings:

```
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$saltxFilTRA$salttion}
kali@kali:~/Documents/coding/python/trycatchhcf/PacketWhisper$
```