Farhad Ahmed
Network Security
Professor McCoy
04/02/19

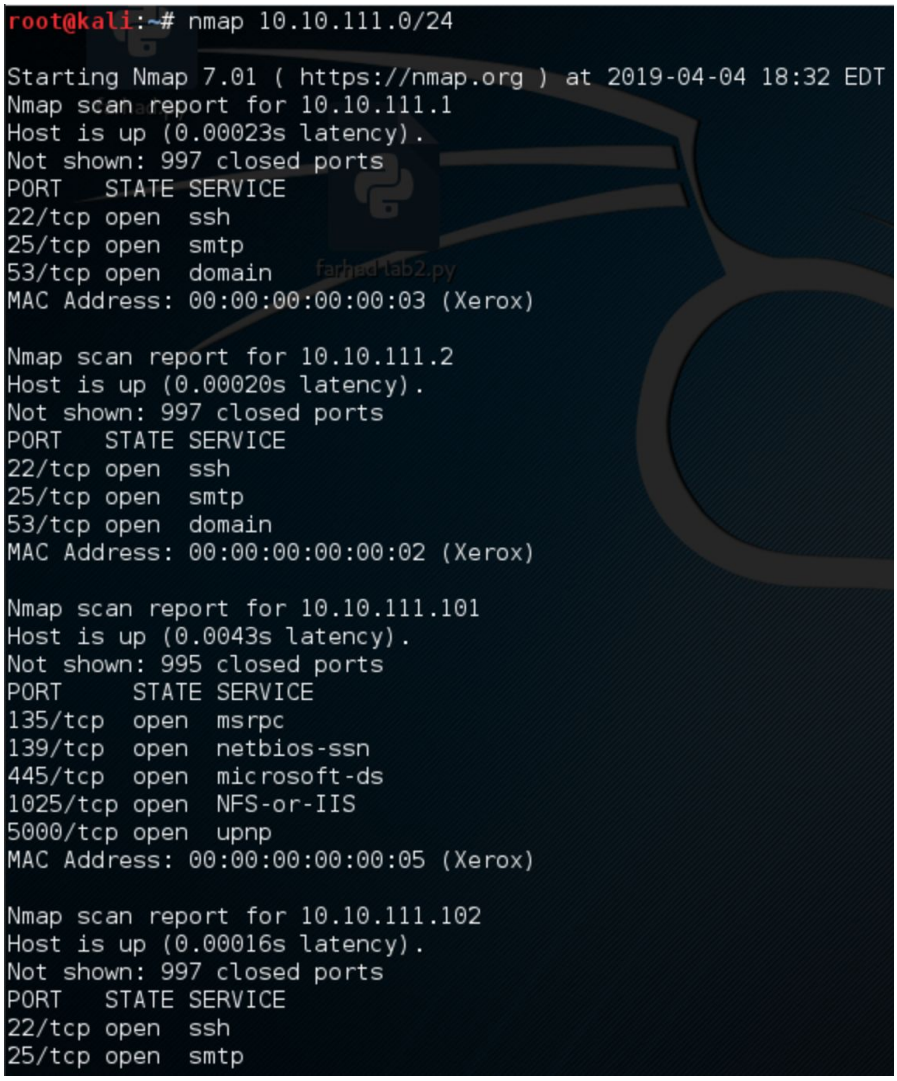
Lab 03 -- NMAP & IPTables

Part 1: NMAP

Nmap is a security scanner that can discover hosts and services on a network, thus creating a network map. This is done by sending specially crafted packets to target hosts and analyzing responses.

The first part of the assignment required that the nmap command be used in the terminal to scan for information about two networks (10.10.111.0/24, 10.20.111.0/24).

So the first command to run was: nmap 10.10.111.0/24
which produced the following output.



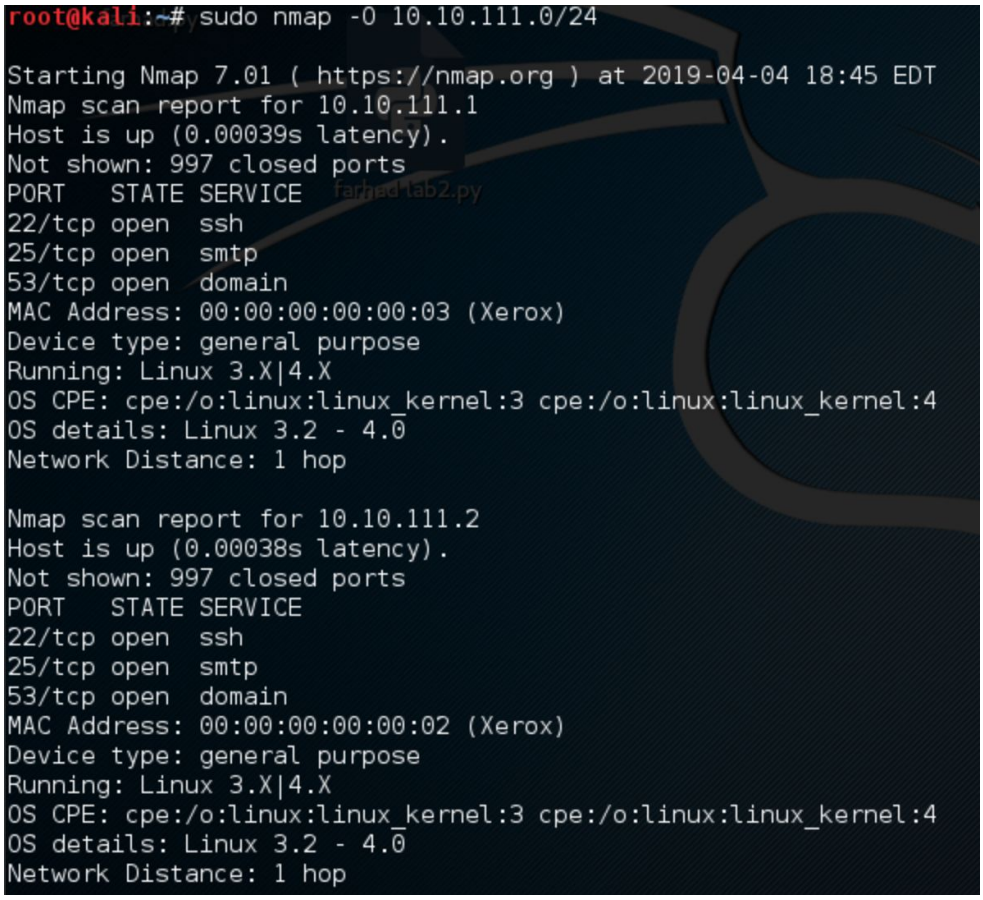
```
53/tcp open  domain
MAC Address: 00:00:00:00:00:06 (Xerox)

Nmap scan report for 10.10.111.103
Host is up (0.00045s latency).
Not shown: 992 filtered ports
PORT     STATE  SERVICE
21/tcp   closed ftp
22/tcp   open   ssh
80/tcp   open   http
445/tcp  open   microsoft-ds
631/tcp  open   ipp
3000/tcp closed ppp
3306/tcp open   mysql
8181/tcp open   unknown
MAC Address: 00:00:00:00:00:07 (Xerox)

Nmap scan report for 10.10.111.100
Host is up (0.0000020s latency).
All 1000 scanned ports on 10.10.111.100 are closed

Nmap done: 256 IP addresses (6 hosts up) scanned in 5.97 seconds
```

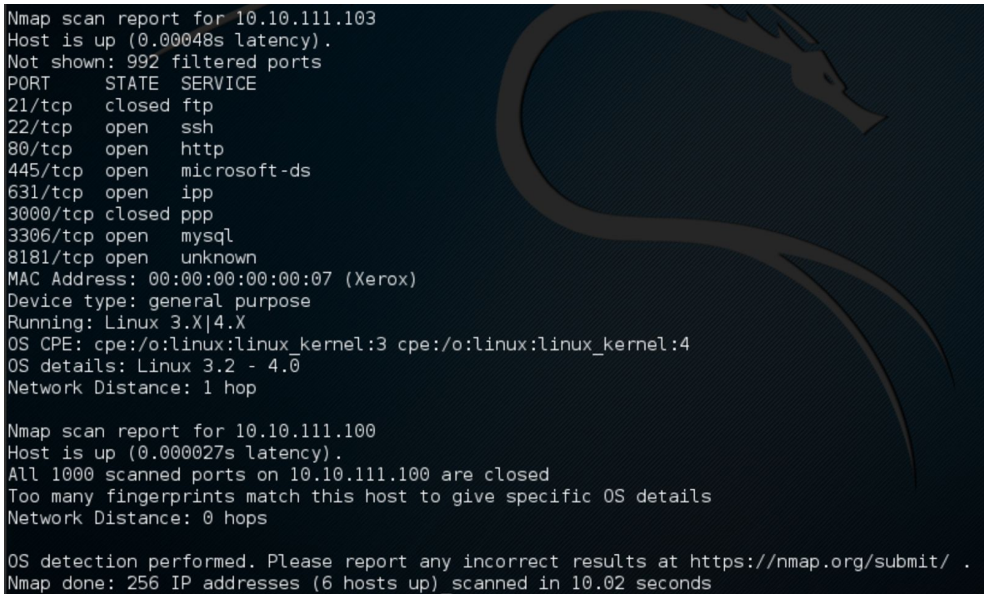
For the second network, the command to run was: nmap 10.20.111.0/24
which produced the following output.

```
root@kali:~# nmap 10.20.111.0/24

Starting Nmap 7.01 ( https://nmap.org ) at 2019-04-04 18:38 EDT
Nmap scan report for 10.20.111.1
Host is up (0.00080s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
25/tcp open  smtp
53/tcp open  domain

Nmap scan report for 10.20.111.2
Host is up (0.0011s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
25/tcp open  smtp
53/tcp open  domain

Nmap done: 256 IP addresses (2 hosts up) scanned in 36.95 seconds
```

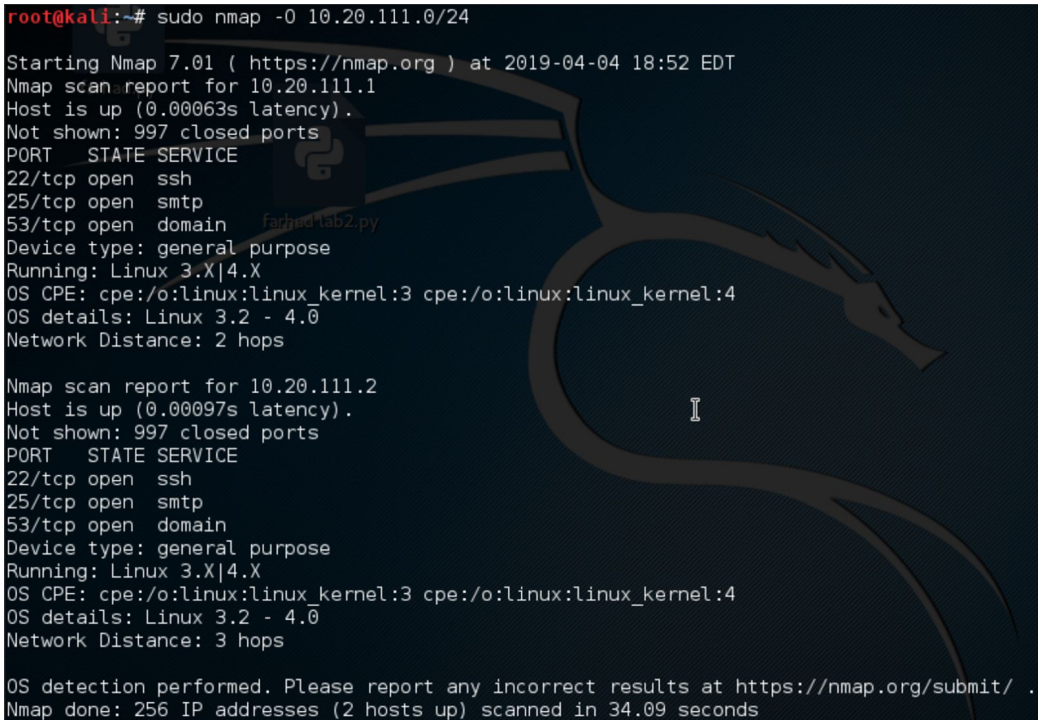
After gathering the information, it was necessary to find all the open ports and the OS on each host in each network.

For the first network the command run was: nmap -O 10.10.111.0/24
The -O flag enables OS detection during the scan.





The command for the next network is: nmap -O 10.20.111.0/24



Part 2: IPTables

A) The iptables firewall on the internal network firewall machine needed to be configured so that for outgoing traffic (from 10.20.111.0/24 to 10.10.111.0/24) the internal machine should be able to communicate with the external network and the external machines without restrictions.

For this step, the command run on the internal router was:
sudo iptables -A FORWARD -s 10.20.111.0/24 -d 10.10.111.0/24 -j ACCEPT

```
student@int-rtr:~$ ping 10.10.111.102
PING 10.10.111.102 (10.10.111.102) 56(84) bytes of data.
64 bytes from 10.10.111.102: icmp_seq=1 ttl=64 time=0.546 ms
64 bytes from 10.10.111.102: icmp_seq=2 ttl=64 time=0.340 ms
64 bytes from 10.10.111.102: icmp_seq=3 ttl=64 time=0.443 ms
64 bytes from 10.10.111.102: icmp_seq=4 ttl=64 time=0.313 ms
64 bytes from 10.10.111.102: icmp_seq=5 ttl=64 time=0.258 ms
64 bytes from 10.10.111.102: icmp_seq=6 ttl=64 time=0.280 ms
64 bytes from 10.10.111.102: icmp_seq=7 ttl=64 time=0.290 ms
64 bytes from 10.10.111.102: icmp_seq=8 ttl=64 time=0.307 ms
64 bytes from 10.10.111.102: icmp_seq=9 ttl=64 time=0.312 ms
64 bytes from 10.10.111.102: icmp_seq=10 ttl=64 time=0.245 ms
```

Running this command will ensure that packets sent to 10.10.111.0/24 by 10.20.111.0/24 are accepted which can verified by pinging an address.

```
64 bytes from 10.10.111.102: icmp_seq=21 ttl=64 time=0.330 ms
^C
--- 10.10.111.102 ping statistics ---
21 packets transmitted, 21 received, 0% packet loss, time 19997ms
rtt min/avg/max/mdev = 0.231/0.307/0.546/0.071 ms
```

B) The next configuration to set was to reject all incoming connection requests with a few exceptions.

1) The first exception was that the internal machine (10.20.111.2) should be allowed to respond to a ping from 10.10.111.0/24.
To set this configuration, the command to be run is:

sudo iptables -A FORWARD -p icmp -s 10.10.111.0/24 -d 10.20.111.2 -j ACCEPT

*The additional -p flag is included so that the type of the packet can be specified.

Running the command produces the following output:

```
student@int-rtr:~$ sudo iptables -A FORWARD -p icmp -s 10.10.111.0/24 -d 10.20.111.2 -
j ACCEPT
[sudo] password for student:
student@int-rtr:~$
```

This can then be verified, by pinging 10.20.111.2 from the Ubuntu machine.

```
student@Ubuntu:~$ ping 10.20.111.2
PING 10.20.111.2 (10.20.111.2) 56(84) bytes of data.
64 bytes from 10.20.111.2: icmp_seq=1 ttl=63 time=0.793 ms
From 10.10.111.1: icmp_seq=2 Redirect Host(New nexthop: 10.10.111.2)
64 bytes from 10.20.111.2: icmp_seq=2 ttl=63 time=0.644 ms
64 bytes from 10.20.111.2: icmp_seq=3 ttl=63 time=0.594 ms
64 bytes from 10.20.111.2: icmp_seq=4 ttl=63 time=0.581 ms
64 bytes from 10.20.111.2: icmp_seq=5 ttl=63 time=0.565 ms
64 bytes from 10.20.111.2: icmp_seq=6 ttl=63 time=0.611 ms
64 bytes from 10.20.111.2: icmp_seq=7 ttl=63 time=0.538 ms
64 bytes from 10.20.111.2: icmp_seq=8 ttl=63 time=0.440 ms
64 bytes from 10.20.111.2: icmp_seq=9 ttl=63 time=0.662 ms
64 bytes from 10.20.111.2: icmp_seq=10 ttl=63 time=0.503 ms
^C
--- 10.20.111.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 8998ms
rtt min/avg/max/mdev = 0.440/0.593/0.793/0.091 ms
student@Ubuntu:~$
```

2) The second exception was that the internal machine (10.20.111.2) should block SSH and http requests from 10.10.111.0/24.
To set the configuration to block SSH requests, the command run on the internal router was:
sudo iptables -A FORWARD -p tcp --dport 22 -s 10.10.111.0/24 -d 10.20.111.2 -j DROP

*The packet type specified is the tcp packet for SSH as well as port 22.
The command for http requests was similar to the above with the only change being the dport which was now 80.
sudo iptables -A FORWARD -p tcp --dport 80 -s 10.10.111.0/24 -d 10.20.111.2 -j DROP

Finally, in order to reject all the packets within the subnet, the following command had to be run:
sudo iptables -A FORWARD -s 10.10.111.0/24 -d 10.20.111.0/24 -j REJECT

All the commands were run in the internal router's terminal as seen below:

```
student@int-rtr:~$ sudo iptables -A FORWARD -p tcp --dport 22 -s 10.10.111.0/24 -d 10.20.111.2 -j DROP
student@int-rtr:~$ sudo iptables -A FORWARD -p tcp --dport 80 -s 10.10.111.0/24 -d 10.20.111.2 -j DROP
student@int-rtr:~$ sudo iptables -A FORWARD -s 10.10.111.0/24 -d 10.20.111.0/24 -j REJECT
```

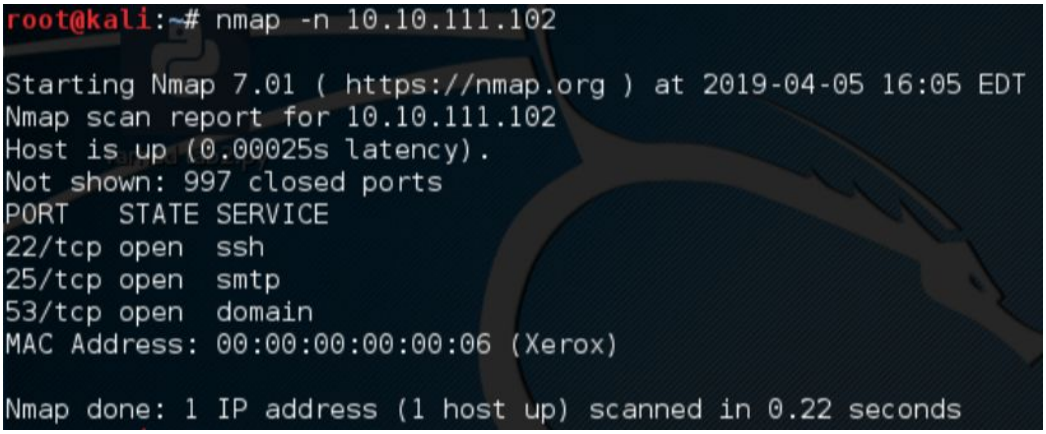
```
student@int-rtr:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  10.20.111.0/24       10.10.111.0/24
ACCEPT     icmp --  10.10.111.0/24       10.20.111.2
DROP       tcp  --  10.10.111.0/24       10.20.111.2          tcp dpt:ssh
DROP       tcp  --  10.10.111.0/24       10.20.111.2          tcp dpt:http
REJECT     all  --  10.10.111.0/24       10.20.111.0/24       reject-with icmp-port-unreachable

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

The configurations set during this part can be verified by running:
wget 10.20.111.2
ssh 10.20.111.2

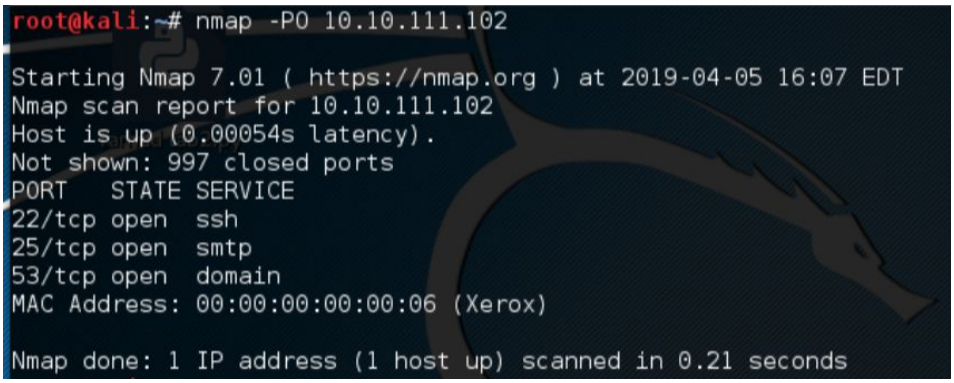
```
student@Ubuntu:~$ wget 10.20.111.2
--2019-04-04 20:44:37--  http://10.20.111.2/
Connecting to 10.20.111.2:80... failed: Connection timed out.
Retrying.

--2019-04-04 20:46:45--  (try: 2)  http://10.20.111.2/
Connecting to 10.20.111.2:80... failed: Connection timed out.
Retrying.

--2019-04-04 20:48:55--  (try: 3)  http://10.20.111.2/
Connecting to 10.20.111.2:80... ^C
student@Ubuntu:~$ ssh 10.20.111.2
ssh: connect to host 10.20.111.2 port 22: Connection timed out
```

Part 3: NMAP & IPTABLES

1. Defining nmap flags
    a. The -n flag specifies that the scan should never do DNS resolution. This can save time during the scan especially if the subnet is large.



    b. The -PO flag can be included to run an IP Protocol ping which is useful for pinging hosts within the subnet that can be discovered.



    c. The -O flag turns on OS detection which can be used to identify the OS of the host that is connected to the IP subnet.

```
root@kali:~# nmap -O 10.10.111.102

Starting Nmap 7.01 ( https://nmap.org ) at 2019-04-05 16:08 EDT
Nmap scan report for 10.10.111.102
Host is up (0.00029s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
25/tcp open  smtp
53/tcp open  domain
MAC Address: 00:00:00:00:00:06 (Xerox)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.0
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/
submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.05 seconds
```

d. The -v flag enables verbose logging, which prints the output of the scan in verbose English allowing for an easier read of the scan results.

```
root@kali:~# nmap -v 10.10.111.102

Starting Nmap 7.01 ( https://nmap.org ) at 2019-04-05 16:10 EDT
Initiating ARP Ping Scan at 16:10
Scanning 10.10.111.102 [1 port]
Completed ARP Ping Scan at 16:10, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 16:10
Completed Parallel DNS resolution of 1 host. at 16:10, 0.00s elapsed
Initiating SYN Stealth Scan at 16:10
Scanning 10.10.111.102 [1000 ports]
Discovered open port 25/tcp on 10.10.111.102
Discovered open port 53/tcp on 10.10.111.102
Discovered open port 22/tcp on 10.10.111.102
Completed SYN Stealth Scan at 16:10, 0.06s elapsed (1000 total ports)
Nmap scan report for 10.10.111.102
Host is up (0.00039s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
25/tcp open  smtp
53/tcp open  domain
MAC Address: 00:00:00:00:00:06 (Xerox)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.27 seconds
           Raw packets sent: 1001 (44.028KB) | Rcvd: 1001 (40.040KB)
```

e. The -oN flag is used along with a path to an output file where the results of the scan will be stored and written to.

```
root@kali:~# nmap -oN scanResults.txt 10.10.111.102

Starting Nmap 7.01 ( https://nmap.org ) at 2019-04-05 16:11 EDT
Nmap scan report for 10.10.111.102
Host is up (0.00043s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
25/tcp open  smtp
53/tcp open  domain
MAC Address: 00:00:00:00:00:06 (Xerox)

Nmap done: 1 IP address (1 host up) scanned in 0.26 seconds
```

Open ▾ scanResults.txt ~/ Sa

```
# Nmap 7.01 scan initiated Fri Apr  5 16:11:50 2019 as: nmap -oN scanResults.txt 10.10.111.102
Nmap scan report for 10.10.111.102
Host is up (0.00043s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
25/tcp open  smtp
53/tcp open  domain
MAC Address: 00:00:00:00:00:06 (Xerox)

# Nmap done at Fri Apr  5 16:11:50 2019 -- 1 IP address (1 host up) scanned in 0.26 seconds
```

2.

a. Running a nmap scan on the Int-Linux machine revealed that it was responding to probes.

```
student@kali:~$ sudo nmap 10.20.111.2

Starting Nmap 7.01 ( https://nmap.org ) at 2019-04-05 13:36 EDT
Nmap scan report for 10.20.111.2
Host is up (0.00060s latency).
All 1000 scanned ports on 10.20.111.2 are filtered

Nmap done: 1 IP address (1 host up) scanned in 5.19 seconds
```

In order to stop this behavior, iptable rules had to be implemented on the Int-Linux to block packets on ports 80 and 445.

```
student@int-linux:~$ sudo iptables -p icmp -A INPUT -s 10.10.111.100 -d 10.20.11
1.2 -j REJECT
student@int-linux:~$ sudo iptables -p tcp --dport 80 -A FORWARD -s 10.10.111.100
 -d 10.20.111.2 -j REJECT
student@int-linux:~$ sudo iptables -p tcp --dport 443 -A FORWARD -s 10.10.111.10
0 -d 10.20.111.2 -j REJECT
student@int-linux:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
REJECT     icmp --  10.10.111.100       10.20.111.2          reject-with icmp-port-unreach
able

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
REJECT     tcp  --  10.10.111.100       10.20.111.2          tcp dpt:http reject-with icmp-port-unreachable
REJECT     tcp  --  10.10.111.100       10.20.111.2          tcp dpt:https reject-with icmp-port-unreachable

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
student@int-linux:~$
```

The rules implemented made sure to reject icmp packets from input as well as any packets that arrived at the Int-Linux machine on ports 80 and 443. This implementation can be verified by running the wget command.

```
student@kali:~$ wget 10.20.111.2
--2019-04-05 13:49:12--  http://10.20.111.2/
Connecting to 10.20.111.2:80... failed: Connection timed out.
Retrying.

--2019-04-05 13:51:20--  (try: 2)  http://10.20.111.2/
Connecting to 10.20.111.2:80... failed: Connection timed out.
Retrying.

--2019-04-05 13:53:29--  (try: 3)  http://10.20.111.2/
Connecting to 10.20.111.2:80... failed: Connection timed out.
Retrying.

--2019-04-05 13:55:39--  (try: 4)  http://10.20.111.2/
Connecting to 10.20.111.2:80... failed: Connection timed out.
Retrying.

--2019-04-05 13:57:51--  (try: 5)  http://10.20.111.2/
Connecting to 10.20.111.2:80... ping 10.20.111.2
^C
```

Finally, pinging the ip shows that the destination ports were unreachable.

```
student@kali:~$ ping 10.20.111.2
PING 10.20.111.2 (10.20.111.2) 56(84) bytes of data.
From 10.20.111.2 icmp_seq=1 Destination Port Unreachable
From 10.20.111.2 icmp_seq=2 Destination Port Unreachable
From 10.20.111.2 icmp_seq=3 Destination Port Unreachable
From 10.20.111.2 icmp_seq=4 Destination Port Unreachable
From 10.20.111.2 icmp_seq=5 Destination Port Unreachable
From 10.20.111.2 icmp_seq=6 Destination Port Unreachable
From 10.20.111.2 icmp_seq=7 Destination Port Unreachable
From 10.20.111.2 icmp_seq=8 Destination Port Unreachable
^V^C
--- 10.20.111.2 ping statistics ---
8 packets transmitted, 0 received, +8 errors, 100% packet loss, time 6999ms
```

B. After implementing the firewall rules the nmap command produces the following output:

```
student@kali:~$ sudo nmap 10.20.111.2

Starting Nmap 7.01 ( https://nmap.org ) at 2019-04-05 15:23 EDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 1.33 seconds
```

C. However, a flag can be added to the nmap scan that allows the a scan over blocked nmaps. That flag is the -Pn flag which marks all devices as online, thus assuming that the host provided is online.

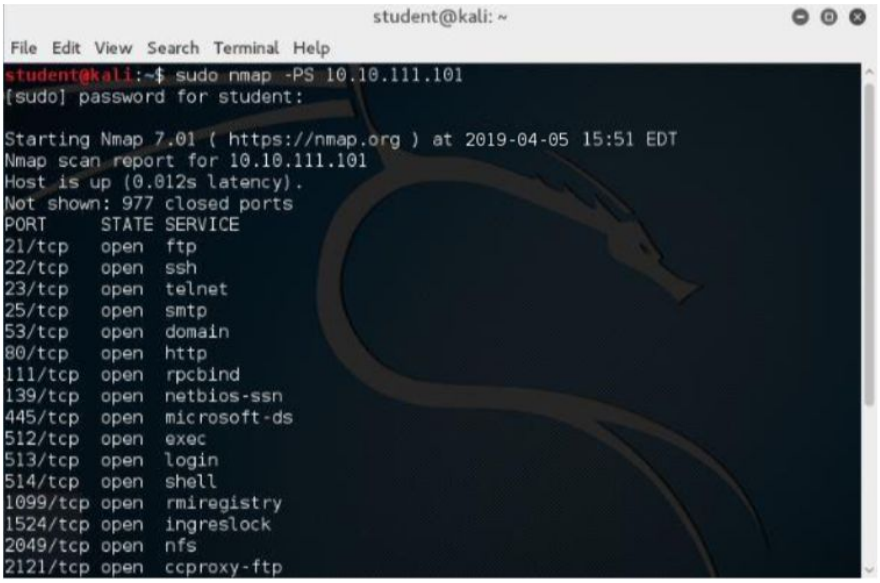
```
student@kali:~$ sudo nmap -Pn 10.20.111.2

Starting Nmap 7.01 ( https://nmap.org ) at 2019-04-05 15:30 EDT
Nmap scan report for 10.20.111.2
Host is up (0.00037s latency).
All 1000 scanned ports on 10.20.111.2 are filtered

Nmap done: 1 IP address (1 host up) scanned in 5.14 seconds
```

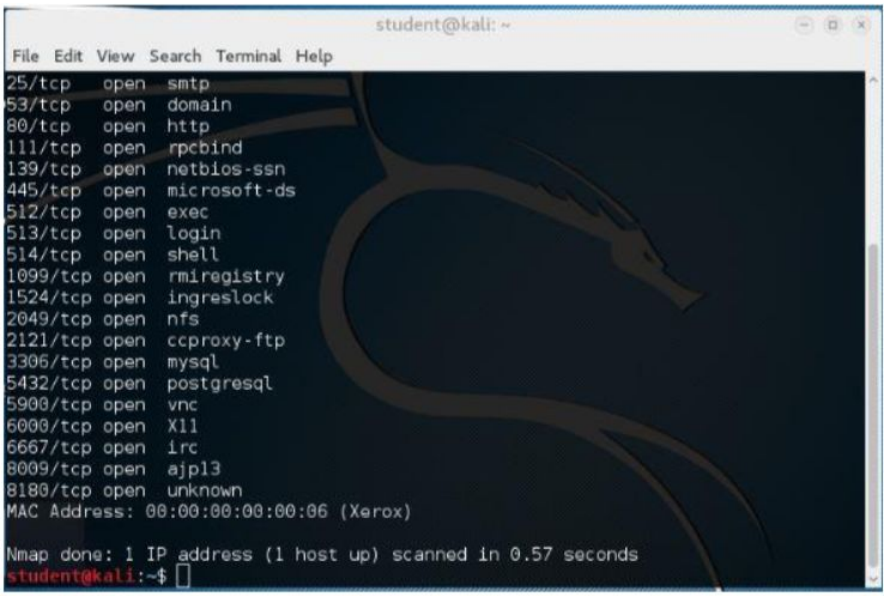
3.

The first step was to run a nmap TCP SYN scan on the Metasploitabe VM using the f following command:

Sudo nmap -PS 10.10.111.101

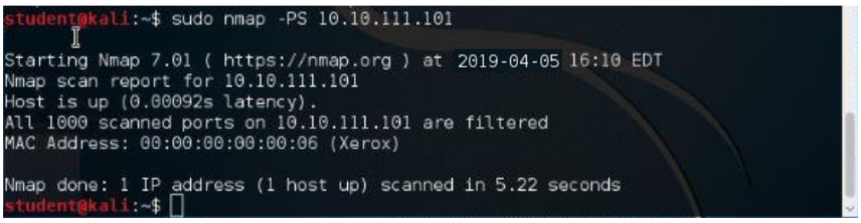
The additional -PS flag indicates that the scan method is TCP SYN.





The scan results show which ports are open on Metasploitable. The next step was to configure iptable rules such that all incoming TCP SYN packets from the Kali Linux machine were blocked. After flushing the old rules, the new rule added was:
sudo iptables -p tcp -A INPUT -s 10.10.111.100 -d 10.10.111.101 -j REJECT

Testing to make sure that the new rule was working properly, the nmap command from before was run again on the kali linux.



Although the nmap still believes the host to be up, the scanned ports are now filtered. This can be used to protect from SYN flood attacks by disallowing attackers from knowing which ports are open. Nonetheless, there are tradeoffs to blocking all TCP SYN packets from a certain IP address. For example, once a certain IP address is blocked it can't establish any future connections with whatever interface blocked the connection. This can be detrimental as the IP that has been blocked might have belonged to a legitimate user. But the benefits might be greater as an attacker trying to carry out a SYN flood attack can have their IP address blocked.