NMAP

Answer to the problem goes here.
Turn in for part 1: Follow the instructions and document the commands and results using screenshots in your report. Explain what is going on in each screenshot.

1. **Using nmap, find all the open ports and OS on each host in the 10.10.111.0/24 network. List the command that is used. [10 points]**
   Using command: nmap -v -sn to scan all hosts in 10.10.111.0/24, and find the hosts which are open. And using nmap -O with the addr can get the OS information. For 10.10.111.0/24 we have 6 open ports (can see figure three). And figures 1-3 are for port information, figures 4-6 are OS information.

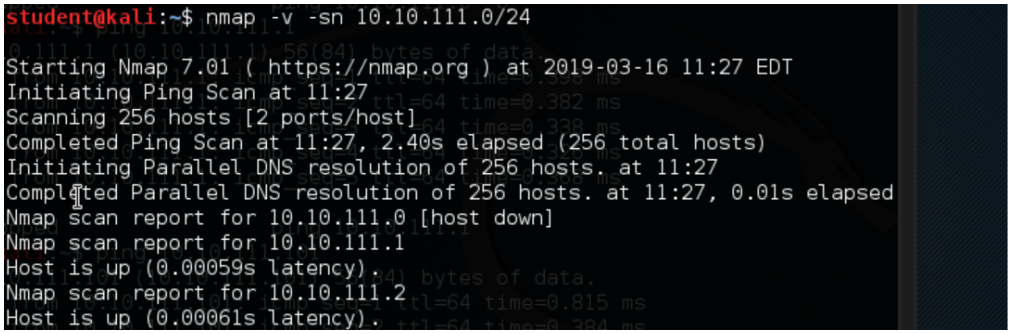

Figure 1: Using nmap -sn to find open host

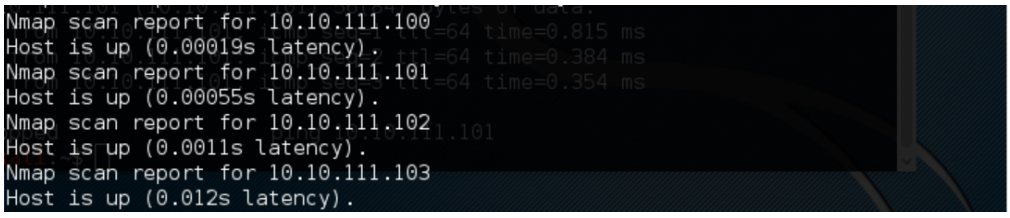

Figure 2: Continue using nmap -sn to find open host

Figure 3: Final result after scanning all hosts

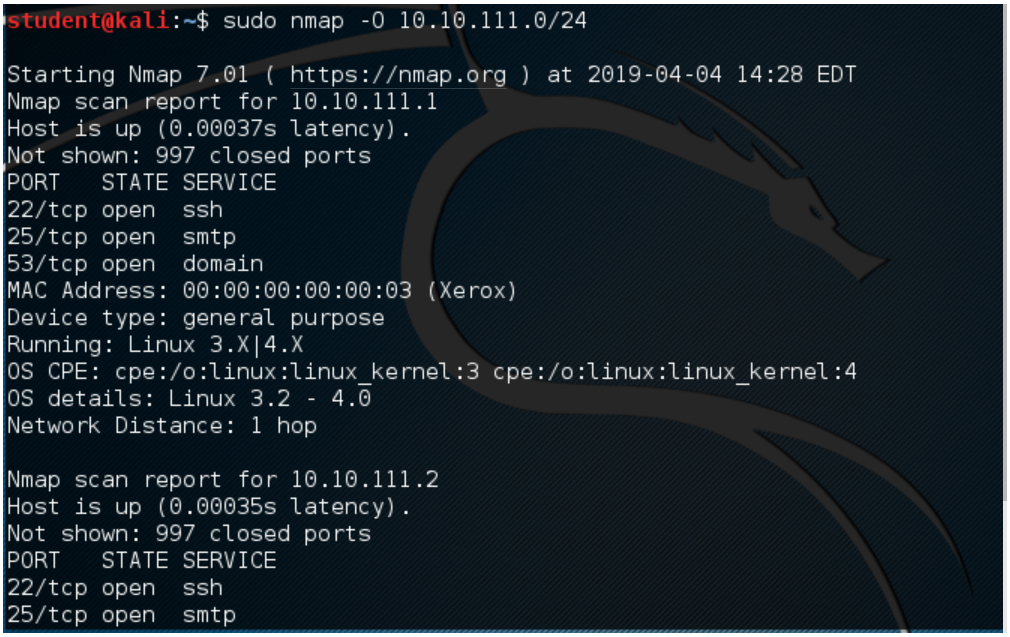

Figure 4: Using nmap -O 10.10.111.0/24 to find OS infor

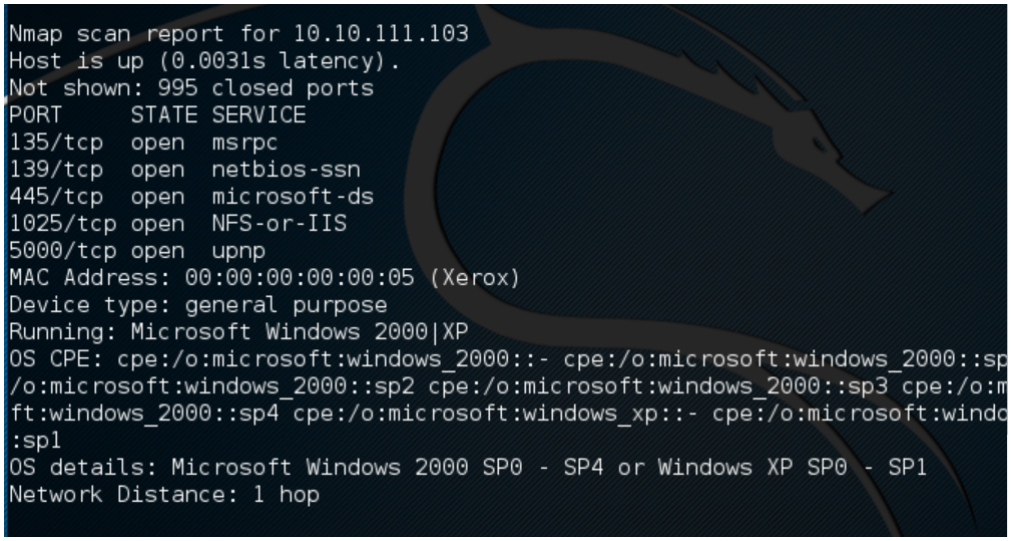

Figure 5: Using nmap -O 10.10.111.0/24 to find OS infor

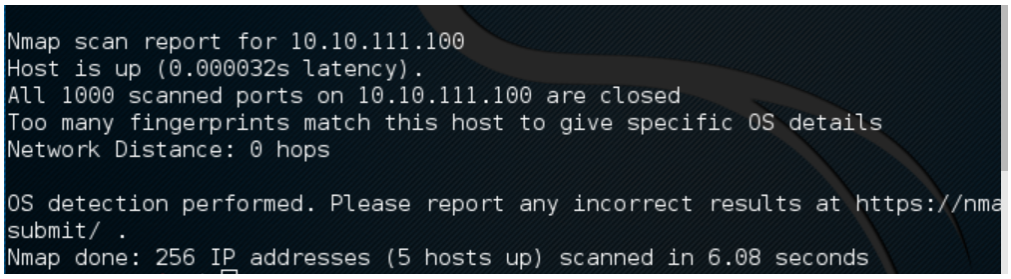

Figure 6: Using nmap -O 10.10.111.0/24 to find OS infor

2. **Using nmap, find all the open ports and OS on each host in the 10.20.111.0/24 network. List the command that is used. [10 points]**
   Using the same command line with last question, and find the open hosts and the OS information for each open port. For 10.20.111.0/24 only two ports are opened. Figures 7 and 8 are for the open ports information, and figures 9 and 10 are for each OS information.

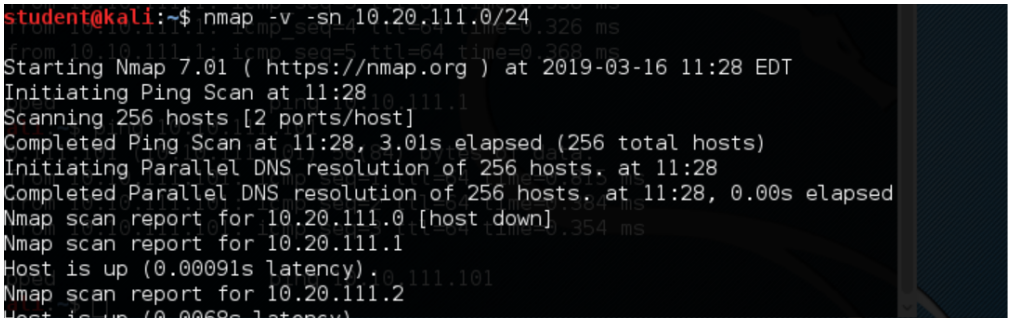

Figure 7: Using nmap -sn to find open host

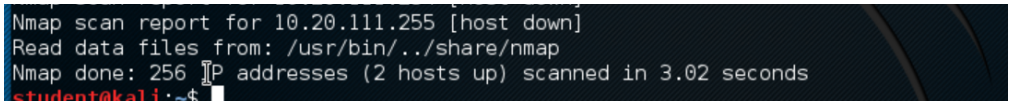

Figure 8: Final result after scanning all hosts

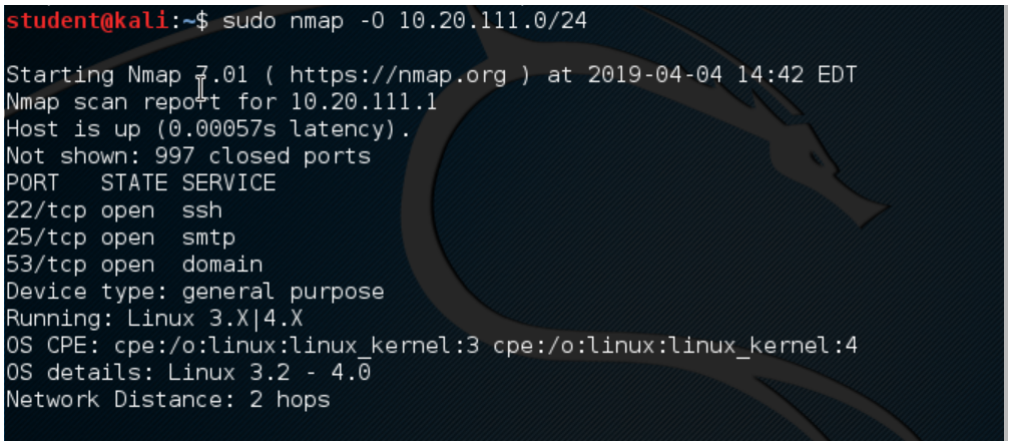

Figure 9: Using nmap -O 10.20.111.0/24 to find OS infor

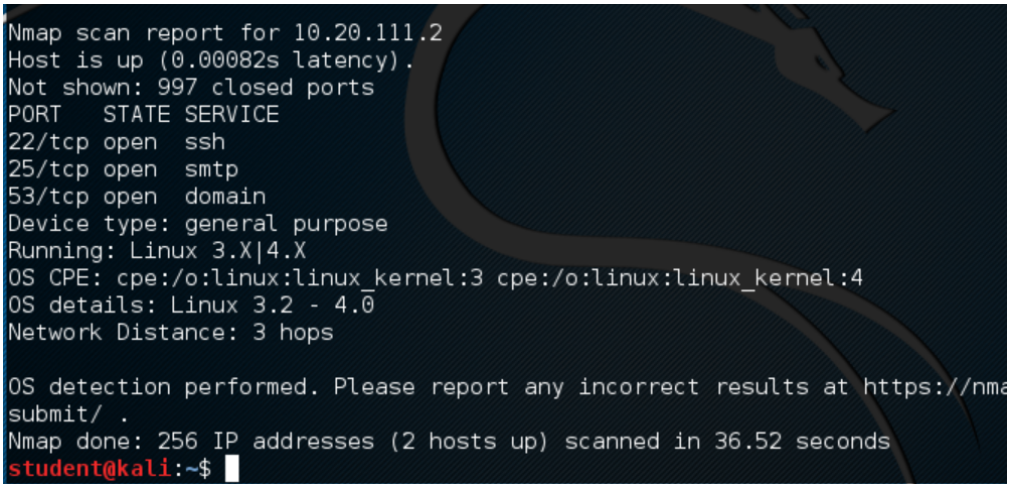

Figure 10: Using nmap -O 10.20.111.0/24 to find OS infor

IPTables

Answer to the problem goes here.
Turn in for part 2: Configure the iptables firewall on the internal network firewall machine to implement the following firewall policies and list the commands used.

1. **For outgoing traffic (from 10.20.111.0/24 to 10.10.111.0/24) - your internal machine should be able to communicate with the external network and the external machines without restrictions. [10 pts]**
   Adding OUTPUT chain policy to meet the command for this question, and verify the right answer by pinging the ip address from 10.10.111.0/24.

```
student@int-rtr:~$ sudo iptables -F
student@int-rtr:~$ sudo iptables -A OUTPUT -d 10.20.111.0/24 -j ACCEPT
student@int-rtr:~$ sudo iptables -A OUTPUT -d 10.10.111.0/24 -j ACCEPT
student@int-rtr:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             10.20.111.0/24
ACCEPT     all  --  anywhere             10.10.111.0/24
student@int-rtr:~$
```

Figure 11: Command line for OUTPUT

```
student@int-rtr:~$ ping 10.10.111.2
PING 10.10.111.2 (10.10.111.2) 56(84) bytes of data.
64 bytes from 10.10.111.2: icmp_seq=1 ttl=64 time=0.063 ms
64 bytes from 10.10.111.2: icmp_seq=2 ttl=64 time=0.049 ms
64 bytes from 10.10.111.2: icmp_seq=3 ttl=64 time=0.049 ms
^Z
[4]+  Stopped                 ping 10.10.111.2
student@int-rtr:~$ 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=64 time=0.355 ms
64 bytes from 10.20.111.2: icmp_seq=2 ttl=64 time=0.339 ms
64 bytes from 10.20.111.2: icmp_seq=3 ttl=64 time=0.418 ms
^Z
[5]+  Stopped                 ping 10.20.111.2
student@int-rtr:~$
```

Figure 12: Verify the iptable's change

2. **For incoming traffic (from the 10.10.111.0/24 to the 10.20.111.0/24) - all incoming connection requests should be rejected with the following exceptions:**

   (a) **The internal machine (10.20.111.2) should respond to a ping from 10.10.111.0/24 [5 pts]**
   Unable all host from 10.10.111.0/24 to ping the host to 10.20.111.0/24, unless ping 10.20.111.2. The first step is to DROP all ping to 10.20.111.0/24 from 10.10.111.0/24. Second step is to ACCEPT the ping to destination ip address which is 10.20.111.2. Using ping request from Kali to verify whether command line in int-router works. And after implement, ping 10.20.111.2 works, but other

hosts doesn't, 10.20.111.1 is opened and 10.20.111.10 is opened which is checked before.

```
student@int-rtr:~$ sudo iptables -A INPUT -s 10.10.111.0/24 -j DROP
student@int-rtr:~$ sudo iptables -A INPUT -s 10.10.111.0/24 -d 10.20.111.2 -i et
h0 -j ACCEPT
student@int-rtr:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  10.10.111.0/24       anywhere
ACCEPT     all  --  10.10.111.0/24       10.20.111.2

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             10.20.111.0/24
ACCEPT     all  --  anywhere             10.10.111.0/24
student@int-rtr:~$
```

Figure 13: Command line for INPUT

```
student@kali:~$ 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=0.755 ms
64 bytes from 10.20.111.2: icmp_seq=2 ttl=63 time=0.796 ms
64 bytes from 10.20.111.2: icmp_seq=3 ttl=63 time=0.743 ms
64 bytes from 10.20.111.2: icmp_seq=4 ttl=63 time=0.674 ms
64 bytes from 10.20.111.2: icmp_seq=5 ttl=63 time=0.714 ms
^Z
[12]+  Stopped                 ping 10.20.111.2
student@kali:~$ ping 10.20.111.1
PING 10.20.111.1 (10.20.111.1) 56(84) bytes of data.
^Z
[13]+  Stopped                 ping 10.20.111.1
student@kali:~$ ping 10.20.111.10
PING 10.20.111.10 (10.20.111.10) 56(84) bytes of data.
From 10.10.111.2 icmp_seq=1 Destination Host Unreachable
From 10.10.111.2 icmp_seq=2 Destination Host Unreachable
From 10.10.111.2 icmp_seq=3 Destination Host Unreachable
^Z
[14]+  Stopped                 ping 10.20.111.10
student@kali:~$
```

Figure 14: Verify the iptable's change

(b) **The internal machine (10.20.111.2) should block all incoming SSH and http requests from 10.10.111.0/24 [5 pts] Verify that your rules are installed correctly by generating appropriate traffic.**
I test two ways to close SSH port. For this question, first we scan the open

ports for 10.20.111.2, and we can find that ssh can be used and http is opened at the first time, as Figure 10. Then, the first method is: *iptables -I INPUT -p tcp –dport 22 -s 10.10.111.0/24 -d 10.20.111.2 -j REJECT* which means to closed the 22 port for the connect from 10.10.111.0/24 (*source*) to the 10.20.111.2 (*destination*). The second method is: *iptables -A INPUT -p tcp –dport ssh -s 10.10.111.0/24 -d 10.20.111.2 -m state –state NEW,ESTABLISHED -j REJECT* and *iptables -A OUTPUT -p tcp –sport 22 -d 10.10.111.0/24 -s 10.20.111.2 -m state –state ESTABLISHED -j REJECT*, which is because the SSH is two way connection, we need to make sure two way are blocked. And for http, *iptables -I INPUT -p tcp –dport 80 -s 10.10.111.0/24 -d 10.20.111.2 -j REJECT* works. To verify the command line, we can use nmap in Kali to scan the 22 port and 80 port, also can ssh 10.20.111.2 to test whether can connect or not.

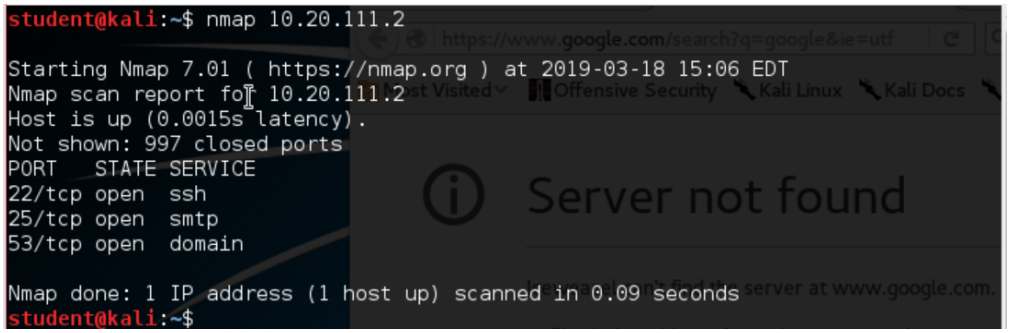

Figure 15: Test the SSH port and nmap 10.20.111.2

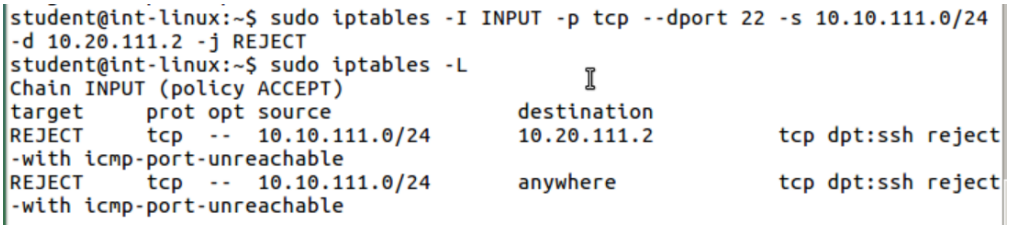

Figure 16: Command line for INPUT to REJECT ssh port

```
student@int-rtr:~$ sudo iptables -I INPUT -d 10.20.111.2 -p tcp --dport 80 -j DR
OP
student@int-rtr:~$ sudo iptables -I INPUT -d 10.20.111.2 -p tcp --dport 22 -j DR
OP
```

Figure 17: Command line for INPUT to DROP http 80 port

```
student@int-rtr:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             10.20.111.2          tcp dpt:ssh
DROP       tcp  --  anywhere             10.20.111.2          tcp dpt:http
DROP       tcp  --  anywhere             anywhere             tcp dpt:http
DROP       tcp  --  anywhere             anywhere             tcp dpt:ssh
DROP       all  --  10.10.111.0/24       anywhere
ACCEPT     all  --  10.10.111.0/24       10.20.111.2
```

Figure 18: iptables

```
student@kali:~$ ssh 10.20.111.2
ssh: connect to host 10.20.111.2 port 22: Connection refused
student@kali:~$
```

Figure 19: Test SSH 10.20.111.2 after implement

```
student@kali:~$ nmap 10.20.111.2

Starting Nmap 7.01 ( https://nmap.org ) at 2019-03-19 15:13 EDT
Nmap scan report for 10.20.111.2
Host is up (0.0012s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
25/tcp open  smtp
53/tcp open  domain

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
student@kali:~$ ssh 10.20.111.2
ssh: connect to host 10.20.111.2 port 22: Connection refused
student@kali:~$
```

Figure 20: Nmap 10.20.111.2 for open ports

Figure 21: Scan http port which is closed

Figure 22: Scan ssh port which is closed

PART 3

Answer to the problem goes here.

1. **Following are the options you will find yourself often needing when using nmap. Use each of these options to perform a scan on the Ubuntu VM using Kali as an attacker machine (see Lab 0 for the login information for the Ubuntu machine). Submit a quick one-liner beside each to explain what each does and screenshots of each scan that you performed. (10 points)**

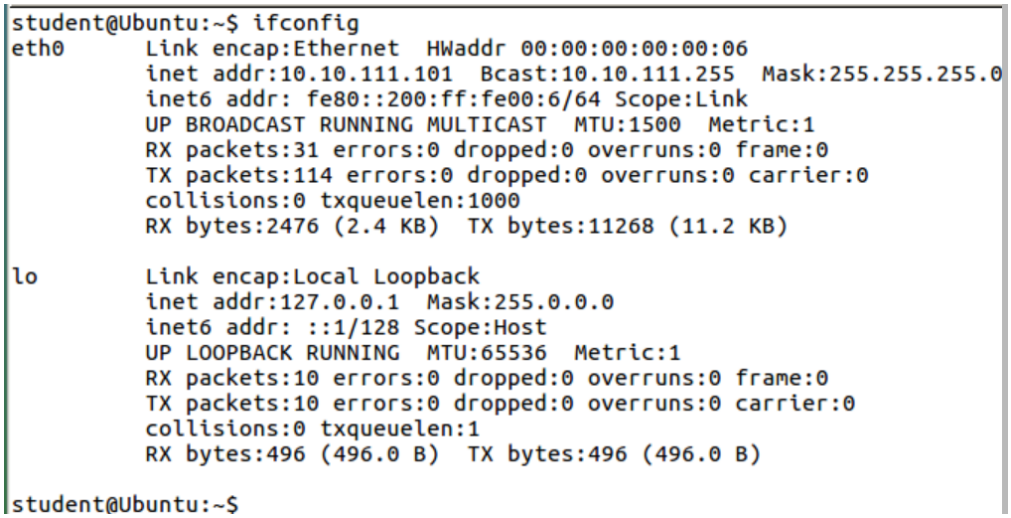

Figure 23: Information fro Ubuntu VM

(a) -n : don't do DNS resolution

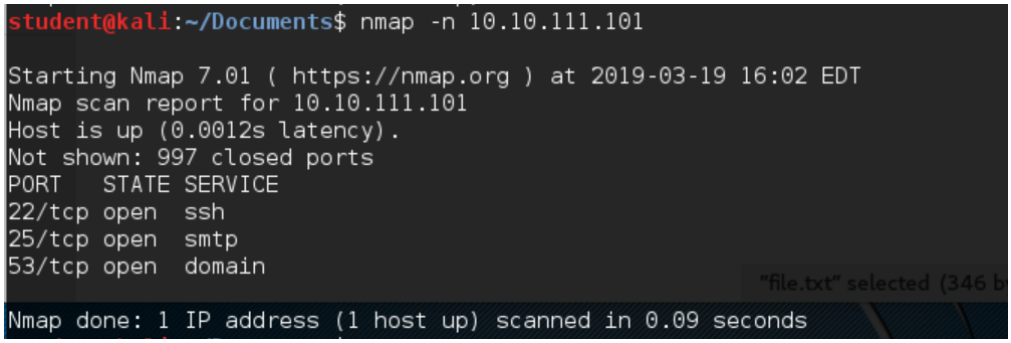

Figure 24: Nmap -n

(b) -P0 : continue with protocol list, for IP protocol Ping

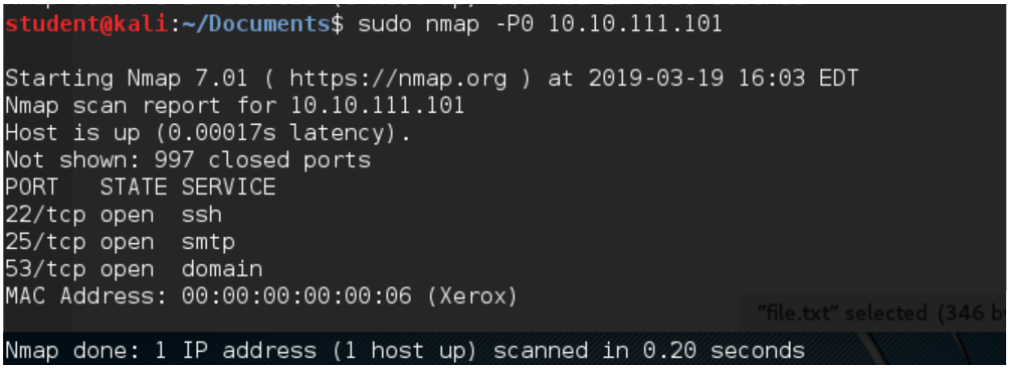

Figure 25: Nmap -P0

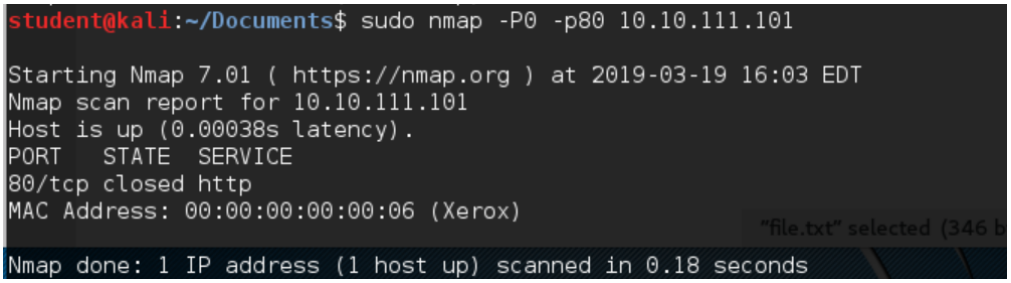

Figure 26: Nmap -P0 for port 80

(c) -O : enable OS detection

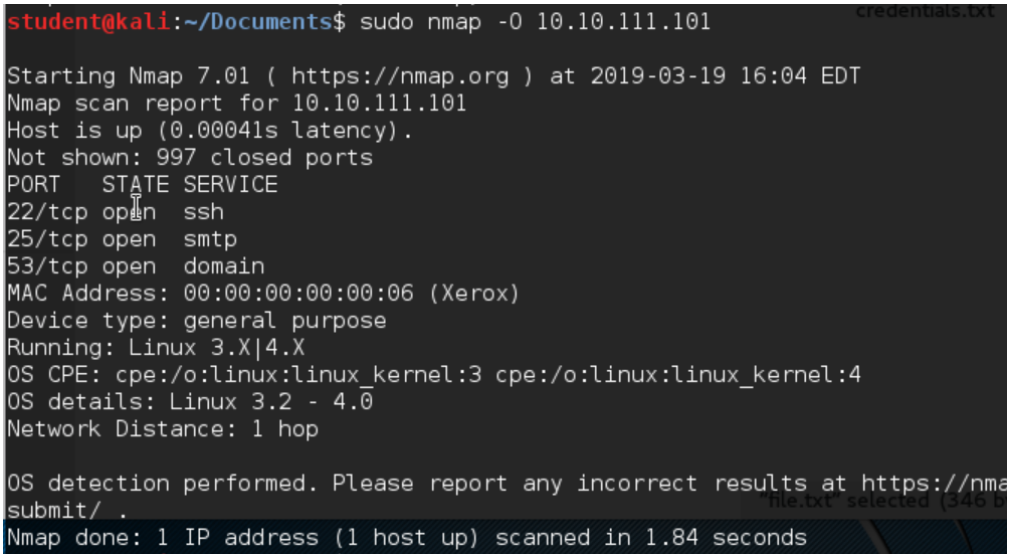

Figure 27: Nmap -O

(d) -v :increase verbosity level, and if use -vv will have more greater effect

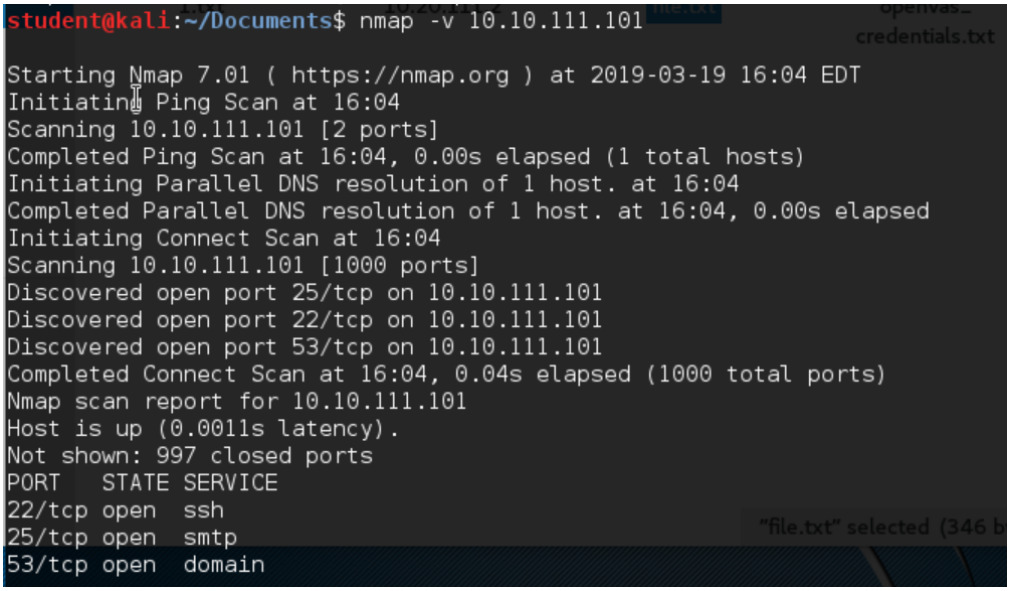

Figure 28: Nmap -v

(e) -oN : output scan in normal, write the output into a directly file.

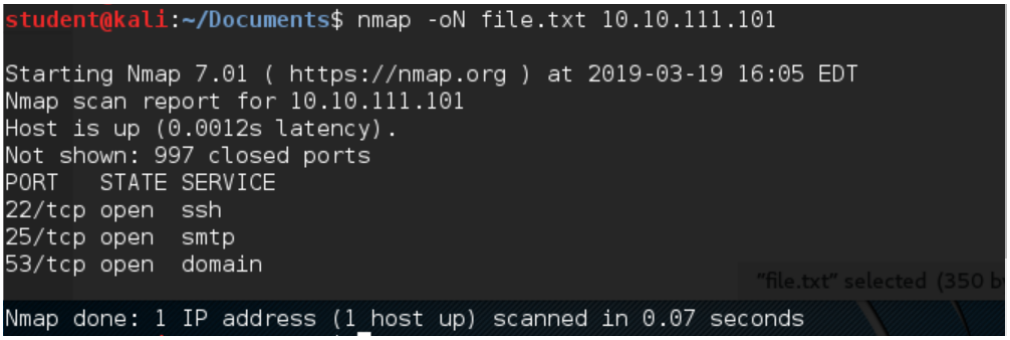

Figure 29: Nmap -oN

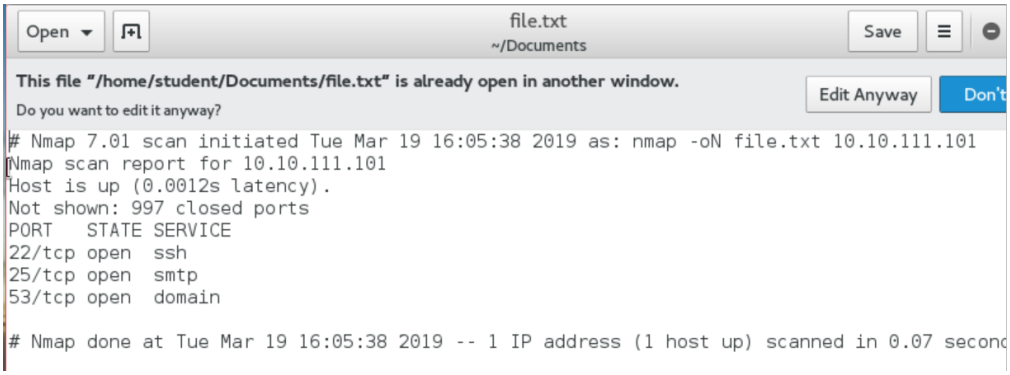

Figure 30: The file created by sys

2. **Using Kali as the scanning machine, perform an nmap scan on the Int-Linux VM (see Lab 0 for the login information for the Int-Linux machine). Include screenshots of the scan results in your report.**

   (a) **Did the Int-Linux VM respond to nmap's probes? If yes, write firewall rules to stop it. This involves blocking incoming ICMP packets, and ports 443 and 80. If you write firewall rules, make sure to verify that they are installed correctly. (5 points)**
   The first step is to block tcp ICMP port, and then block http and https ports as the question mentions. first two figures show to block the ICMP, and next two is for http and https. And the basic way to verify is to ping 10.20.111.2.

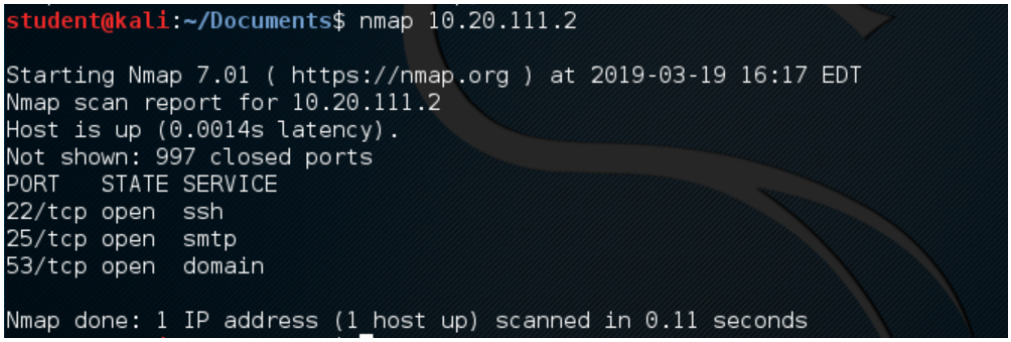

Figure 31: Basic nmap scan before implement

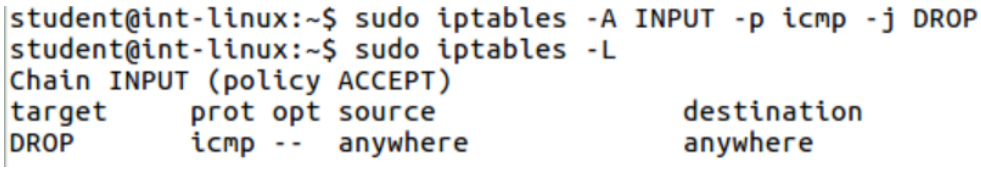

Figure 32: ICMP block implement

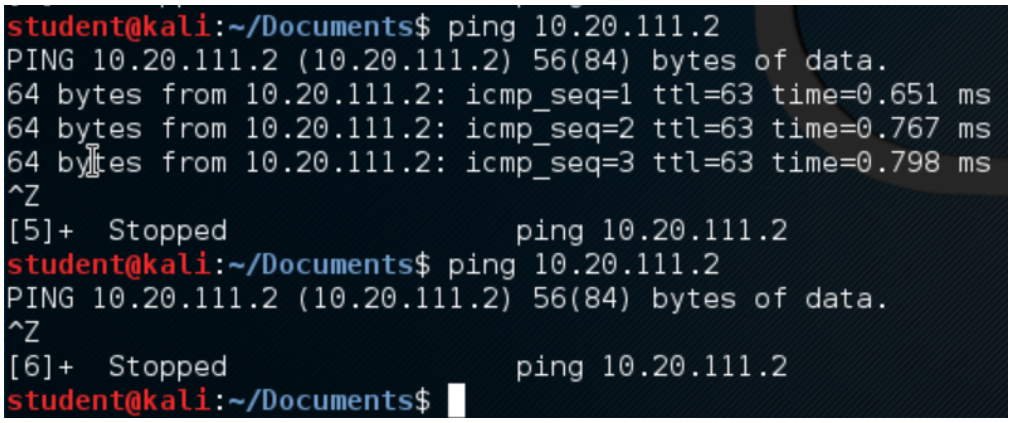

Figure 33: ping checking after implement(Compare before and after

```
student@int-linux:~$ sudo iptables -A INPUT -d 10.20.111.2 -p tcp --dport 443 -j DROP
student@int-linux:~$ sudo iptables -A INPUT -d 10.20.111.2 -p tcp --dport 80 -j DROP
student@int-linux:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere
DROP       tcp  --  anywhere             10.20.111.2          tcp dpt:https
DROP       tcp  --  anywhere             10.20.111.2          tcp dpt:http
```

Figure 34: Block the http and https ports

(b) **Now that you have implemented the appropriate rules on the IntLinux VM, execute nmap from Kali. Submit screenshots of your nmap command and results of your scan. (5 points)**
This time, after implement before question, when using nmap tp scan, it shows me cannot nmap 10.20.111.2.

```
student@kali:~/Documents$ nmap 10.20.111.2

Starting Nmap 7.01 ( https://nmap.org ) at 2019-03-19 17:04 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 3.04 seconds
```

Figure 35: Verification

(c) **There is a method of forcing nmap to scan hosts even if the initial nmap probes are blocked. Leaving the iptables in place that block nmap's initial probe requests, run nmap with a set of options that scans Int-Linux even when it doesn't reply to nmap's initial probe requests. Include the nmap options you used and a screenshot of the scan. (5 points)**
Using nmap -Pn to force scan.

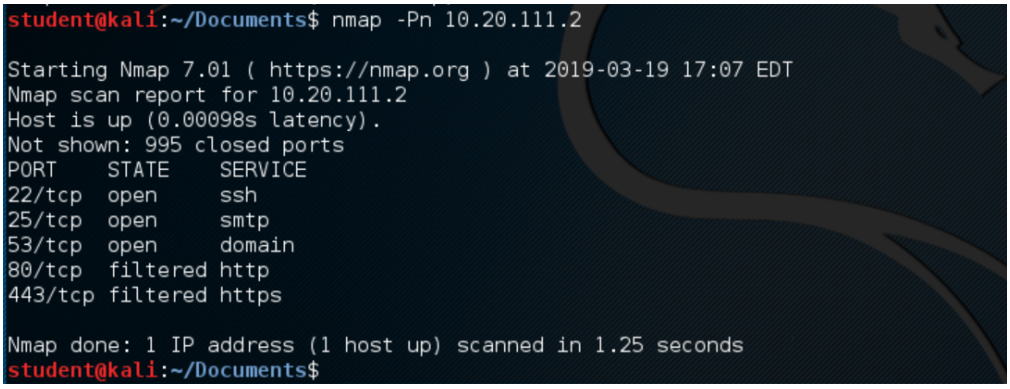

Figure 36: Forcing nmap to scan

3. **Using the Kali machine as an attacker machine, perform a nmap TCP SYN scan on the Metasploitable VM (see Lab 0 for the login information for the Metasploitable machine). Then construct an iptable rule to block all incoming TCP SYN packets only from the Kali scanning server's IP address. Explain the trade offs of blocking all TCP SYN packets from an IP address. Submit screenshots of your TCP SYN scans before and after applying the iptable rule. (5 points)**
   The first step is to know the ip address from Metasploitable, as figure 32, we can know the IP address is 10.10.111.102. second step is to figure out the connection between Metasploitable and Kali, and we can know we need to block **SYN, ACK, FIN, RST** in order to block all realation packets. And we need to block INPUT and OUTPUT. The next is verification, nmap -sS to verify.

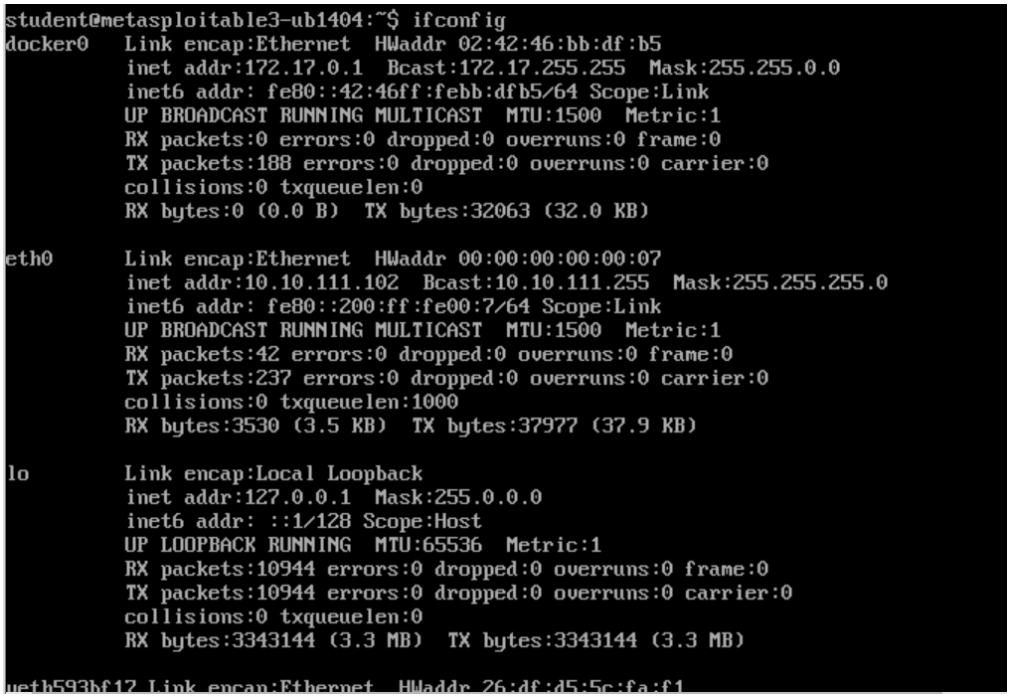

Figure 37: Basic information for Metasploitable

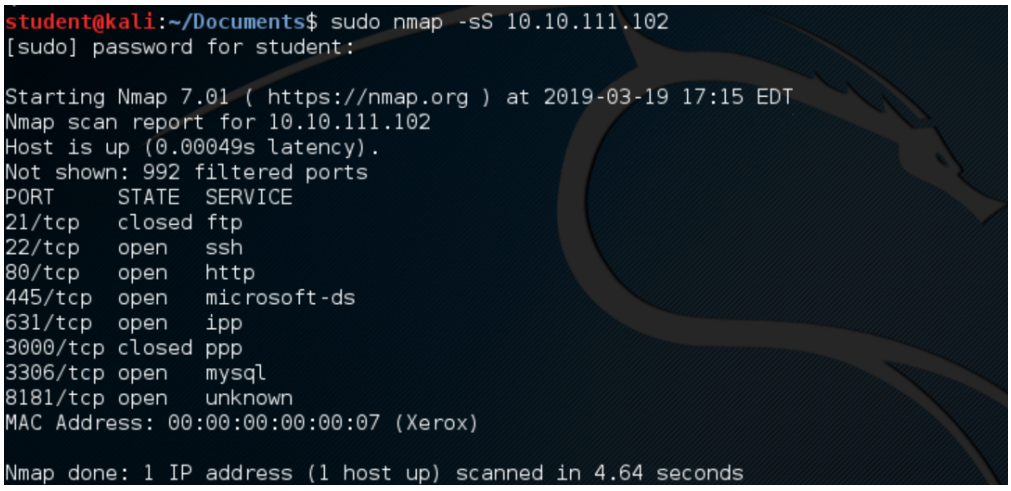

Figure 38: nmap scan in Kali

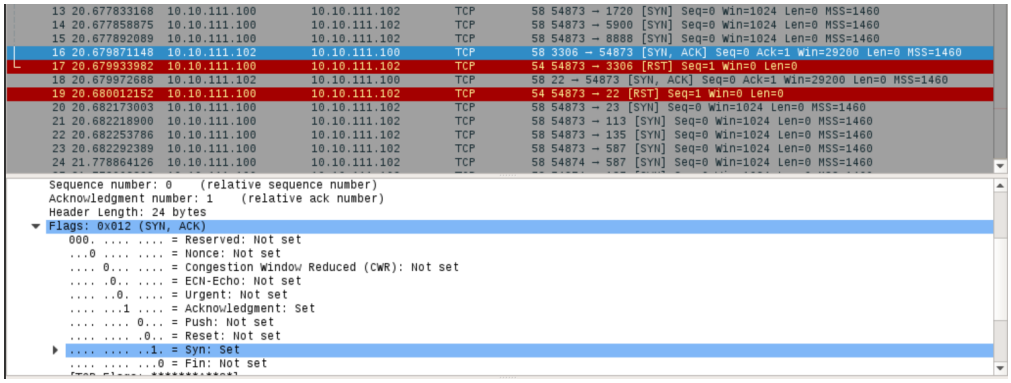

Figure 39: Wireshark scan the connect

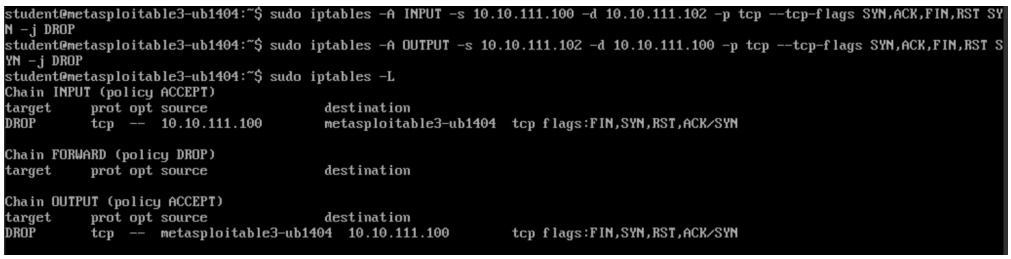

Figure 40: Block the TCP SYN packet

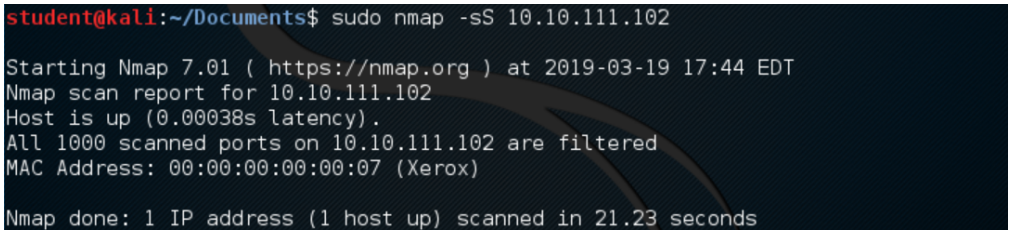

Figure 41: Verification