Network Security and Forensics

CSEC.462.01

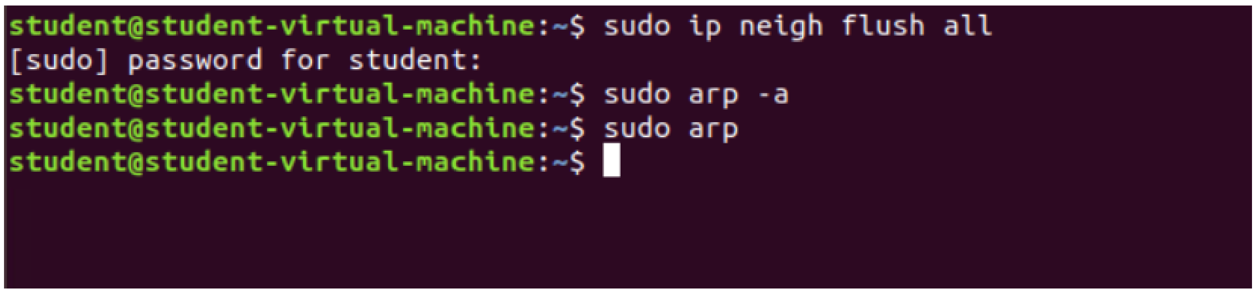
Lab 2

Student: Wissam El Labban

Table of contents

Step 1 **1**
What does the default firewall in Ubuntu look like? 2
Explanation of output 3
The NAT table 4
Step 2 **5**
Pinging the ubuntu machine from kali 5
Setting the iptables rules on Ubuntu 6
Testing that ubuntu does not get pings and explanation of observations 7
Step 3 **9**
SSH download 9
Adding and testing the two rules 10
Will this work? 10
Adding and testing the two rules in reverse order 11
Will this work now? 11
Step 4 **13**
What does this do? 13
Is this rule triggered before or after the other rules we see? 13
Step 5 **14**
Finding the kali mac address 14
Setting up the filter table and the complete command 15
Testing the ruleset 15
Step 6 **17**
Rebooting and seeing what happens to the firewall 17
Adding a rule and saving it 18
Rebooting and restoring rules 18
References **19**

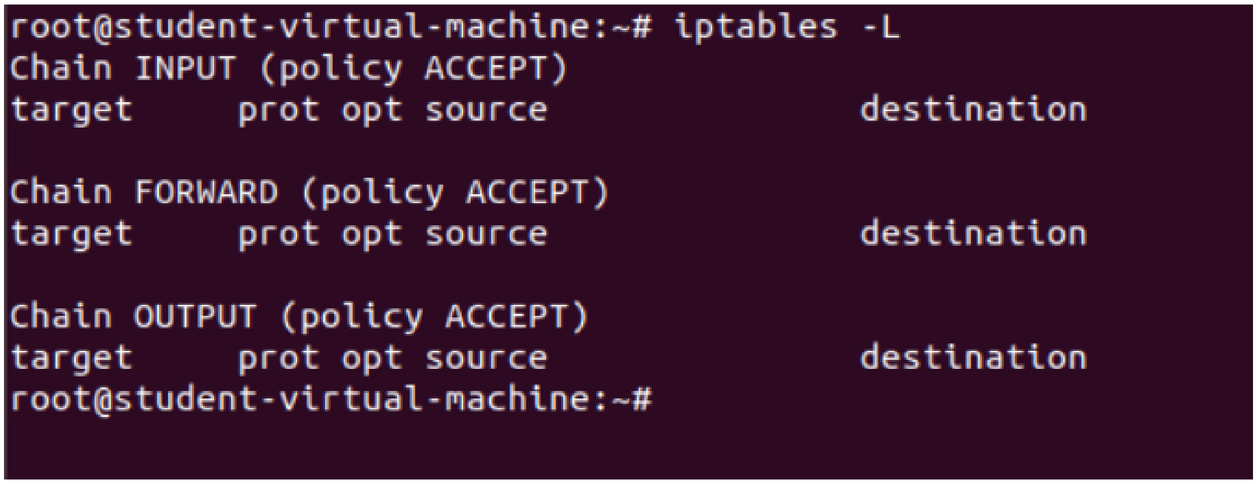
Step 1



```
student@student-virtual-machine:~$ sudo ip neigh flush all
[sudo] password for student:
student@student-virtual-machine:~$ sudo arp -a
student@student-virtual-machine:~$ sudo arp
student@student-virtual-machine:~$ 
```

Cleared the ARP table on the ubuntu machine.

What does the default firewall in Ubuntu look like?



```
root@student-virtual-machine:~# 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
root@student-virtual-machine:~#
```

Iptables -L output in above shows the default firewall in ubuntu.

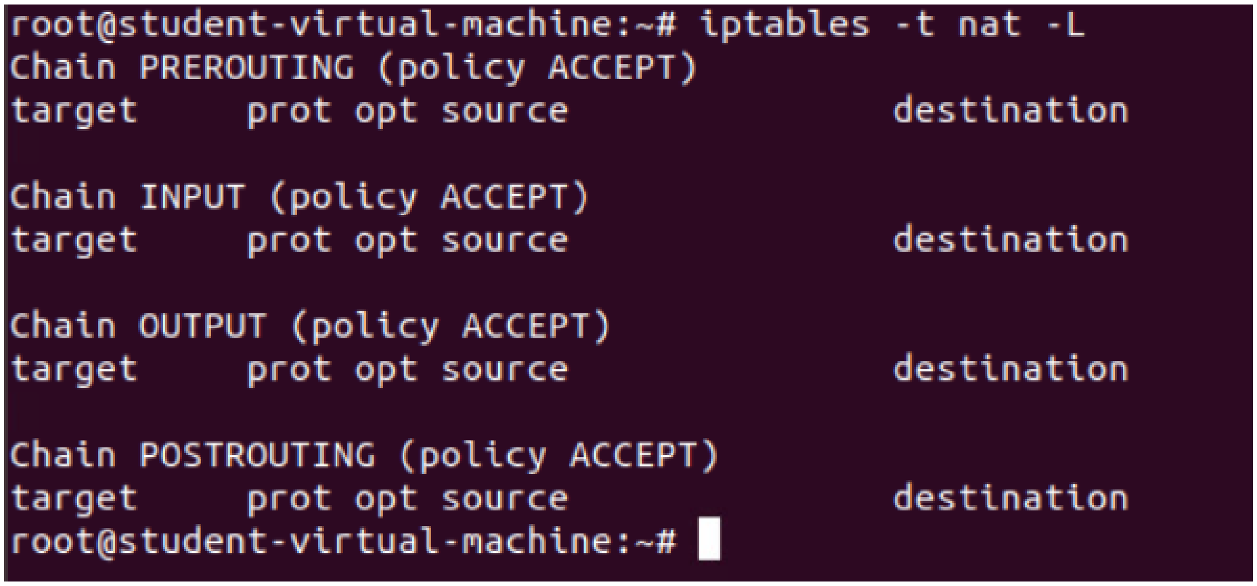
Explanation of output

There are Chains which are a sequence of rules that apply to packets that meet certain criteria. There are three built-in chains in this table: INPUT, FORWARD, and OUTPUT.

- Input filters packets that are destined for the local system.
- Output filters packets that are sent to remote systems from the local system
- Forward filter packets that are routed through the local system to another destination (this is the kind of chain in IPtables that routers would use).

Target under each chain shows what actions are taken if the Packets fall under those categories. Target has actions like ACCEPT, DROP, and REJECT. However, we are not seeing any of those here.
The reason is because this is the default configuration of the Ubuntu machine where there are no rules defined. All three chains have ACCEPT as their default policy which means that all packets that are either generated by the machine, have the machine as the destination, or go to get forwarded by the machine (even though it cannot forward them since IPTables should have NICs and ipv4 forwarding manually configured to forward packets) will be accepted and are allowed to proceed.

The NAT table

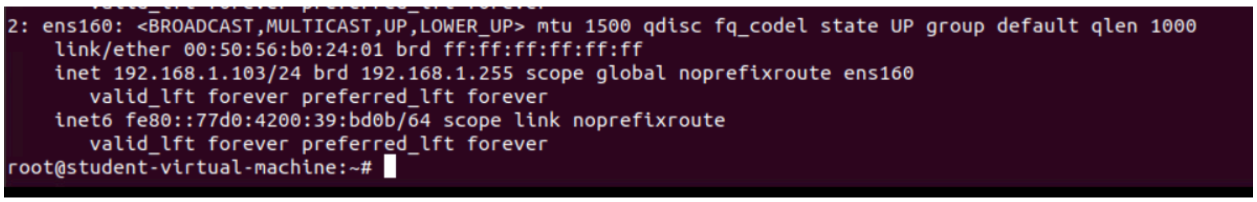


```
root@student-virtual-machine:~# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
root@student-virtual-machine:~#
```

The above screenshot shows the output of “iptables -t nat -L”. This is the nat table from the iptables and there is a prerouting chain that was not available on the filter table. The prerouting chain applies the given rules on a packet before it is routed to its final destination. Some tables have the prerouting chain in them while filter doesn’t. Like the filter table, our output has no rules and all policies are ACCEPT by default.

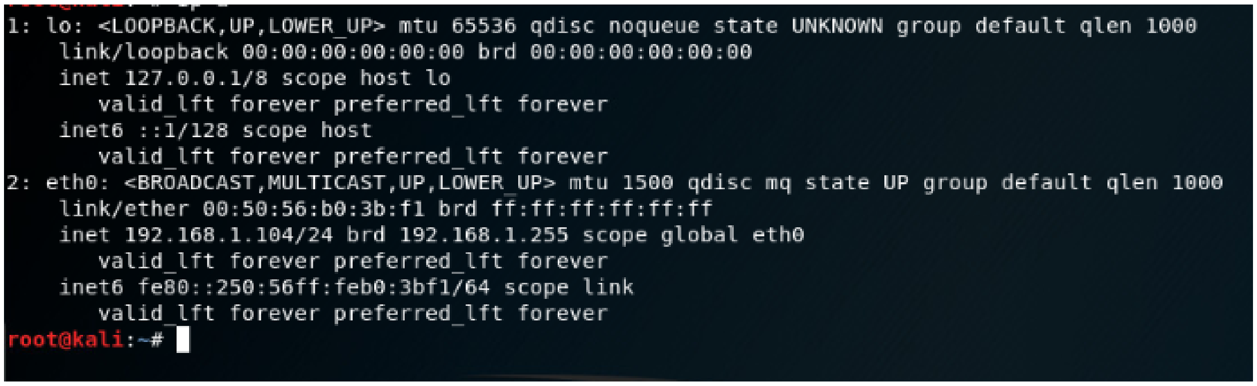
Step 2

Pinging the ubuntu machine from kali



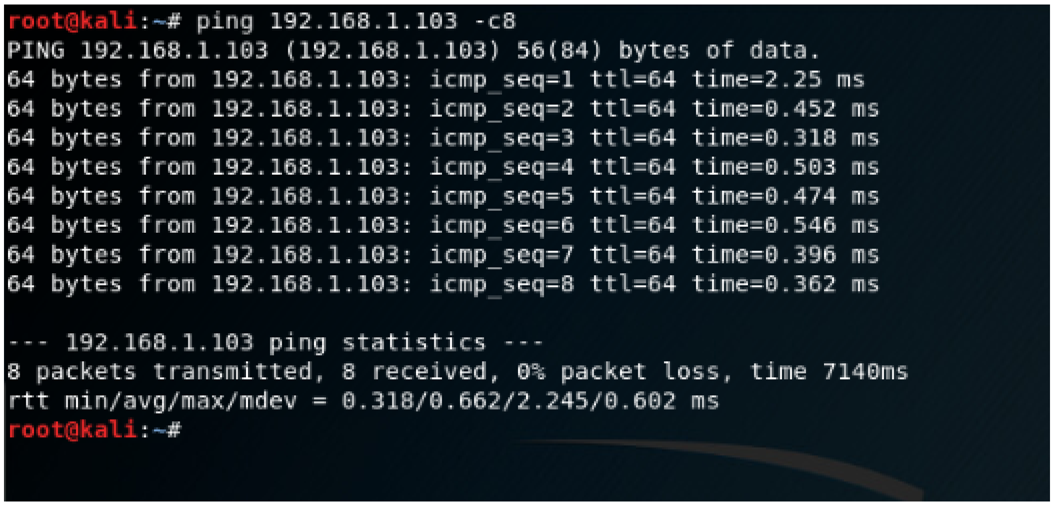
```
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:50:56:b0:24:01 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.103/24 brd 192.168.1.255 scope global noprefixroute ens160
       valid_lft forever preferred_lft forever
    inet6 fe80::77d0:4200:39:bd0b/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
root@student-virtual-machine:~#
```

Ip address of ubuntu machine is 192.168.1.103



```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:b0:3b:f1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.104/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:feb0:3bf1/64 scope link
       valid_lft forever preferred_lft forever
root@kali:~#
```

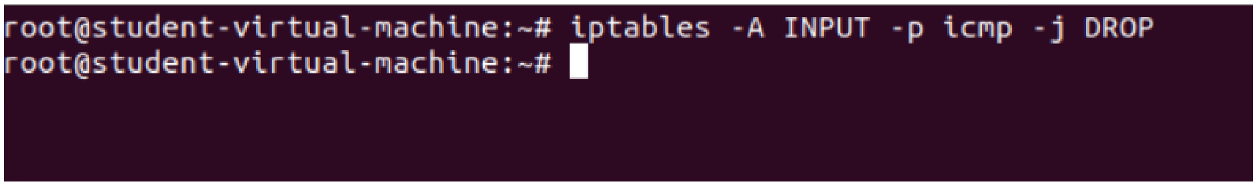
The ip address of the kali machine is 192.168.1.104



```
root@kali:~# ping 192.168.1.103 -c8
PING 192.168.1.103 (192.168.1.103) 56(84) bytes of data.
64 bytes from 192.168.1.103: icmp_seq=1 ttl=64 time=2.25 ms
64 bytes from 192.168.1.103: icmp_seq=2 ttl=64 time=0.452 ms
64 bytes from 192.168.1.103: icmp_seq=3 ttl=64 time=0.318 ms
64 bytes from 192.168.1.103: icmp_seq=4 ttl=64 time=0.503 ms
64 bytes from 192.168.1.103: icmp_seq=5 ttl=64 time=0.474 ms
64 bytes from 192.168.1.103: icmp_seq=6 ttl=64 time=0.546 ms
64 bytes from 192.168.1.103: icmp_seq=7 ttl=64 time=0.396 ms
64 bytes from 192.168.1.103: icmp_seq=8 ttl=64 time=0.362 ms

--- 192.168.1.103 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7140ms
rtt min/avg/max/mdev = 0.318/0.662/2.245/0.602 ms
root@kali:~#
```

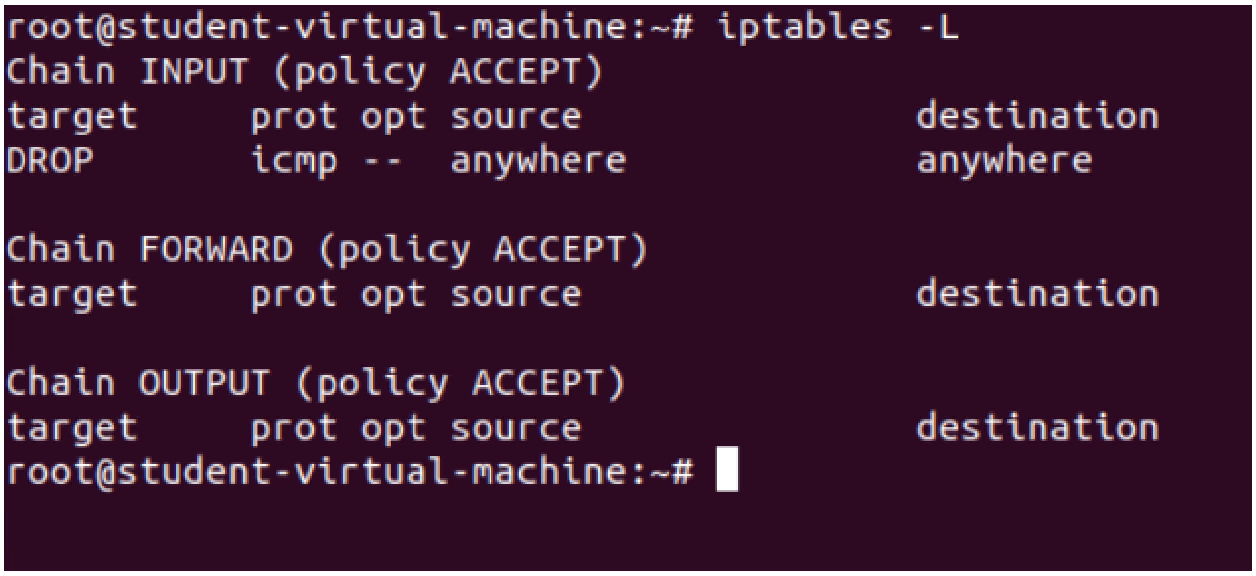
Pinging the ubuntu from the kali machine is no problem.

Setting the iptables rules on Ubuntu



```
root@student-virtual-machine:~# iptables -A INPUT -p icmp -j DROP
root@student-virtual-machine:~# 
```

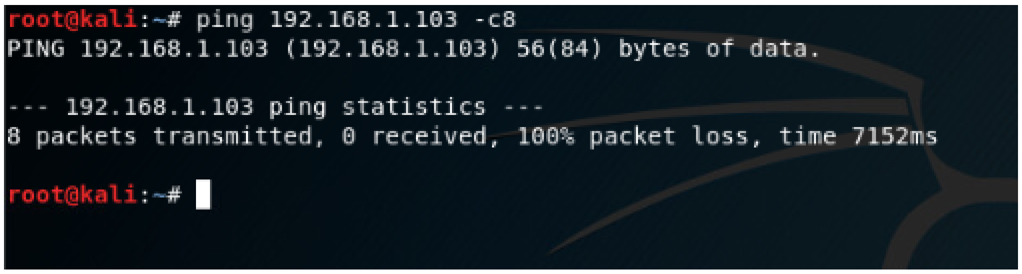
Dropping all incoming ICMP packets with the command above.



```
root@student-virtual-machine:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@student-virtual-machine:~# 
```

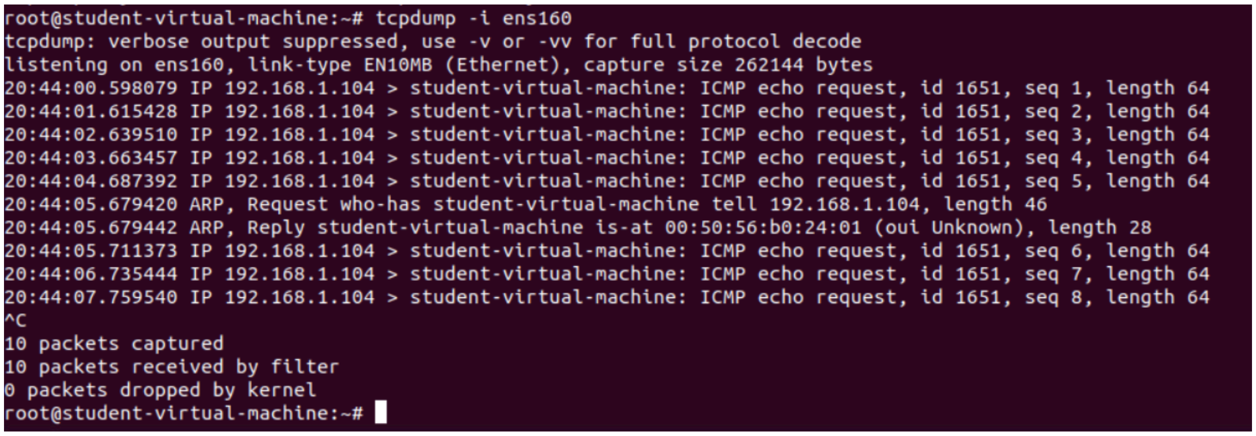
I have verified that it is on the filter table now.

Testing that ubuntu does not get pings and explanation of observations



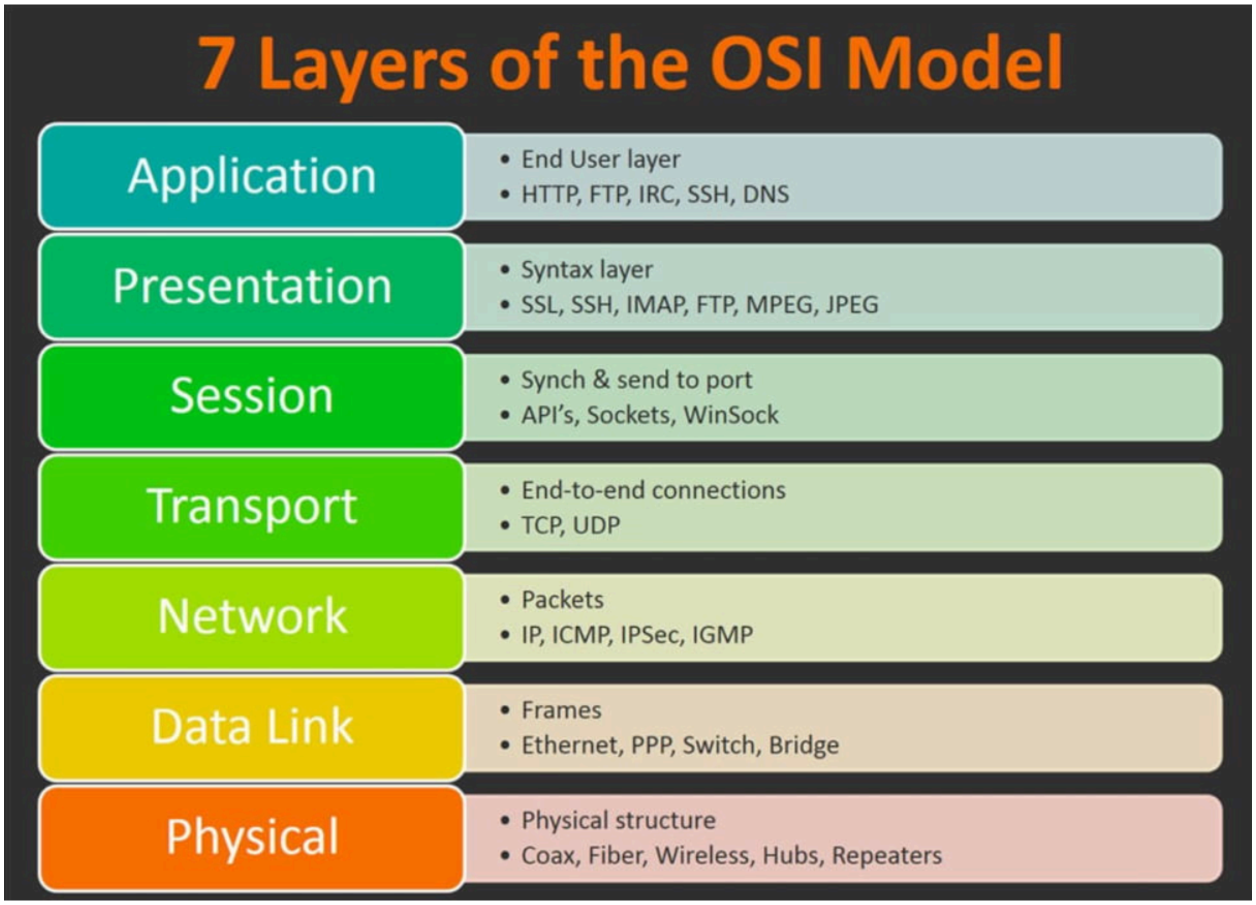
```
root@kali:~# ping 192.168.1.103 -c8
PING 192.168.1.103 (192.168.1.103) 56(84) bytes of data.

--- 192.168.1.103 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7152ms

root@kali:~# 
```

As you can see, there are no ICMP messages being replied back from the ubuntu machine.



```
root@student-virtual-machine:~# tcpdump -i ens160
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens160, link-type EN10MB (Ethernet), capture size 262144 bytes
20:44:00.598079 IP 192.168.1.104 > student-virtual-machine: ICMP echo request, id 1651, seq 1, length 64
20:44:01.615428 IP 192.168.1.104 > student-virtual-machine: ICMP echo request, id 1651, seq 2, length 64
20:44:02.639510 IP 192.168.1.104 > student-virtual-machine: ICMP echo request, id 1651, seq 3, length 64
20:44:03.663457 IP 192.168.1.104 > student-virtual-machine: ICMP echo request, id 1651, seq 4, length 64
20:44:04.687392 IP 192.168.1.104 > student-virtual-machine: ICMP echo request, id 1651, seq 5, length 64
20:44:05.679420 ARP, Request who-has student-virtual-machine tell 192.168.1.104, length 46
20:44:05.679442 ARP, Reply student-virtual-machine is-at 00:50:56:b0:24:01 (oui Unknown), length 28
20:44:05.711373 IP 192.168.1.104 > student-virtual-machine: ICMP echo request, id 1651, seq 6, length 64
20:44:06.735444 IP 192.168.1.104 > student-virtual-machine: ICMP echo request, id 1651, seq 7, length 64
20:44:07.759540 IP 192.168.1.104 > student-virtual-machine: ICMP echo request, id 1651, seq 8, length 64
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
root@student-virtual-machine:~# 
```

However, when I use tcpdump on the ubuntu machine, I can see that there are ICMP packets that are being received despite my iptables rule. That's completely fine.



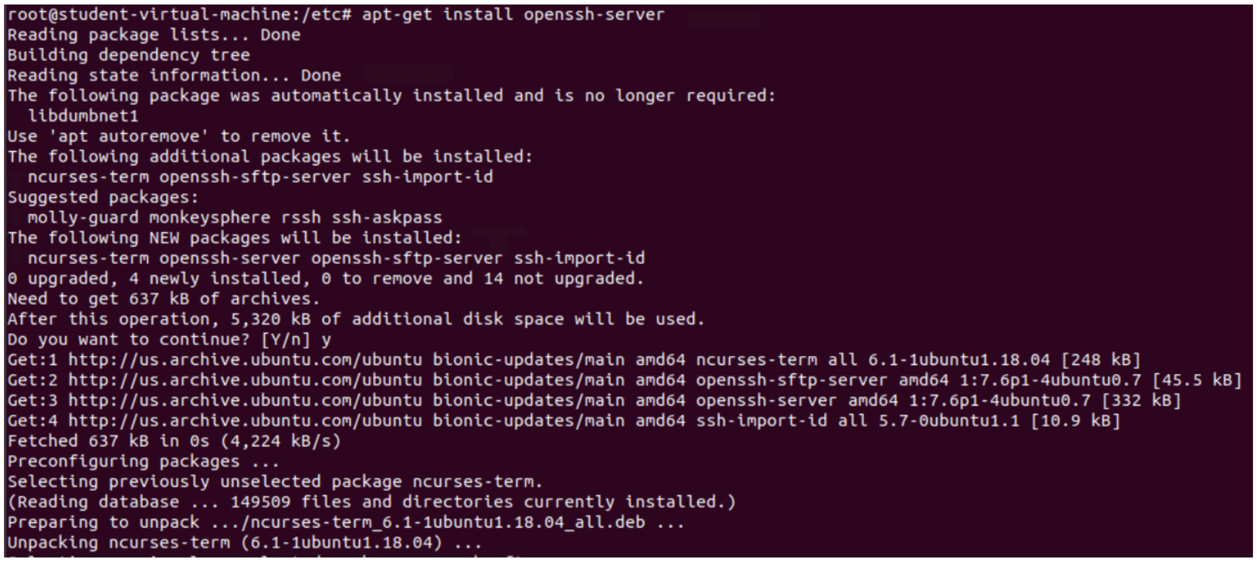
7 Layers of the OSI Model

Application	• End User layer • HTTP, FTP, IRC, SSH, DNS
Presentation	• Syntax layer • SSL, SSH, IMAP, FTP, MPEG, JPEG
Session	• Synch & send to port • API's, Sockets, WinSock
Transport	• End-to-end connections • TCP, UDP
Network	• Packets • IP, ICMP, IPSec, IGMP
Data Link	• Frames • Ethernet, PPP, Switch, Bridge
Physical	• Physical structure • Coax, Fiber, Wireless, Hubs, Repeaters

Tcpdump works on the data link layer while iptables works on the network layer. This means that even though the packets are being dropped by iptables, they are still going through the data link layer of the ubuntu machine and are still visible on tools like tcpdump before they are dropped by iptables on the network layer.

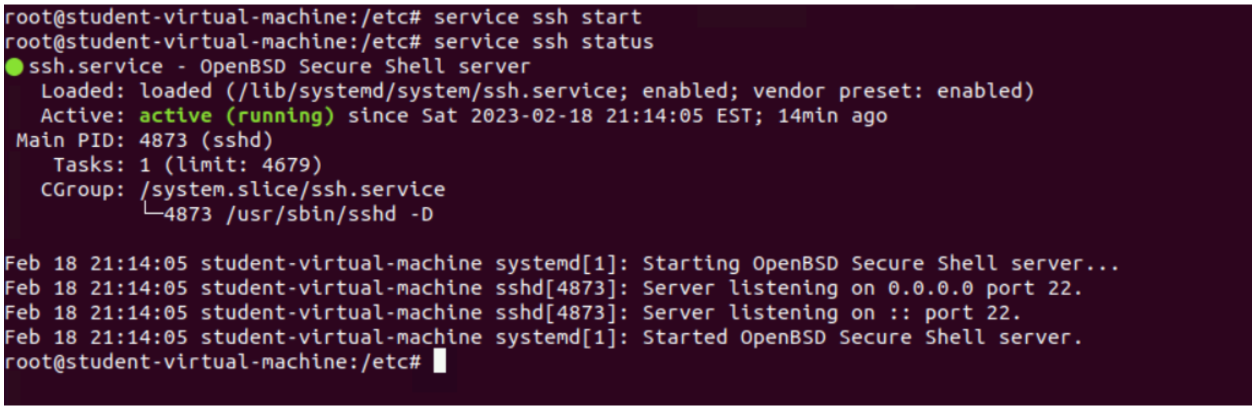
Step 3

SSH download



```
root@student-virtual-machine:/etc# apt-get install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libdumbnet1
Use 'apt autoremove' to remove it.
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere rssh ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 14 not upgraded.
Need to get 637 kB of archives.
After this operation, 5,320 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ncurses-term all 6.1-1ubuntu1.18.04 [248 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 openssh-sftp-server amd64 1:7.6p1-4ubuntu0.7 [45.5 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 openssh-server amd64 1:7.6p1-4ubuntu0.7 [332 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ssh-import-id all 5.7-0ubuntu1.1 [10.9 kB]
Fetched 637 kB in 0s (4,224 kB/s)
Preconfiguring packages ...
Selecting previously unselected package ncurses-term.
(Reading database ... 149509 files and directories currently installed.)
Preparing to unpack .../ncurses-term_6.1-1ubuntu1.18.04_all.deb ...
Unpacking ncurses-term (6.1-1ubuntu1.18.04) ...
```

Downloaded ssh

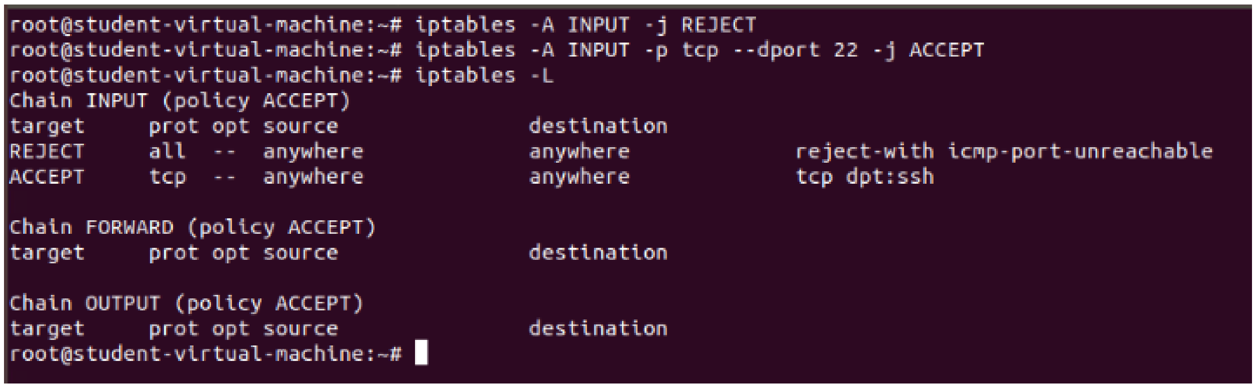


```
root@student-virtual-machine:/etc# service ssh start
root@student-virtual-machine:/etc# service ssh status
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2023-02-18 21:14:05 EST; 14min ago
 Main PID: 4873 (sshd)
    Tasks: 1 (limit: 4679)
   CGroup: /system.slice/ssh.service
           └─4873 /usr/sbin/sshd -D

Feb 18 21:14:05 student-virtual-machine systemd[1]: Starting OpenBSD Secure Shell server...
Feb 18 21:14:05 student-virtual-machine sshd[4873]: Server listening on 0.0.0.0 port 22.
Feb 18 21:14:05 student-virtual-machine sshd[4873]: Server listening on :: port 22.
Feb 18 21:14:05 student-virtual-machine systemd[1]: Started OpenBSD Secure Shell server.
root@student-virtual-machine:/etc#
```

Started ssh service

Adding and testing the two rules

After flushing the iptables with the iptables -F command. I did the following.



```
root@student-virtual-machine:~# iptables -A INPUT -j REJECT
root@student-virtual-machine:~# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
root@student-virtual-machine:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
REJECT     all  --  anywhere             anywhere             reject-with icmp-port-unreachable
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@student-virtual-machine:~#
```

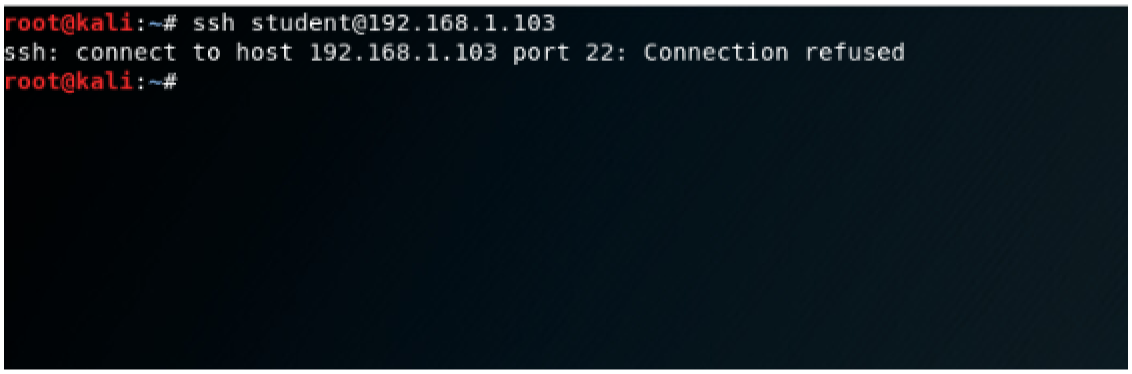
The first command in the image above will reject all incoming traffic.

The second command is adding an acceptance rule to tcp on port 22.

The filter table shows those changes in effect.

Will this work?

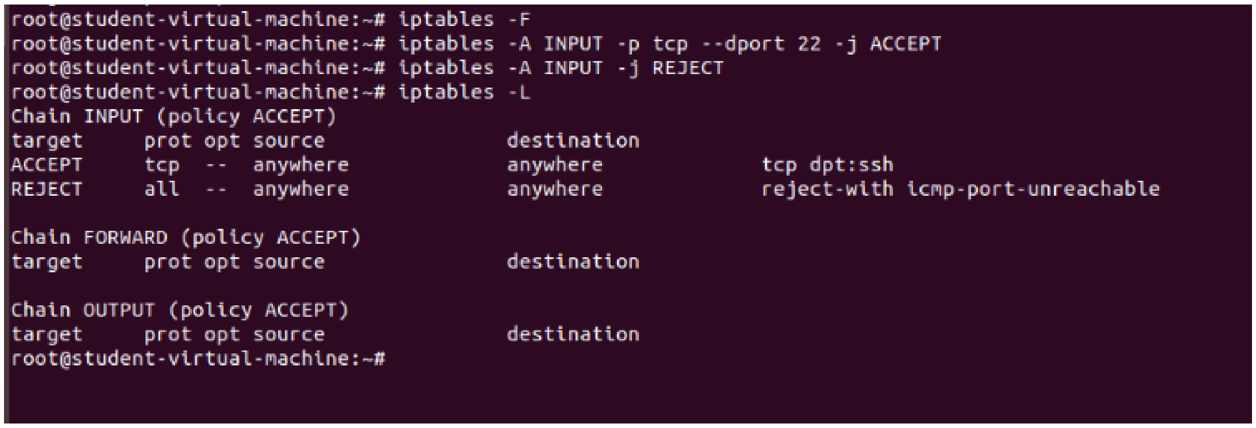
Since the reject rule is coming first, the accept rule will never come into play since it is processed after the reject rule.



```
root@kali:~# ssh student@192.168.1.103
ssh: connect to host 192.168.1.103 port 22: Connection refused
root@kali:~#
```

I cannot ssh into the ubuntu machine from kali.

Adding and testing the two rules in reverse order

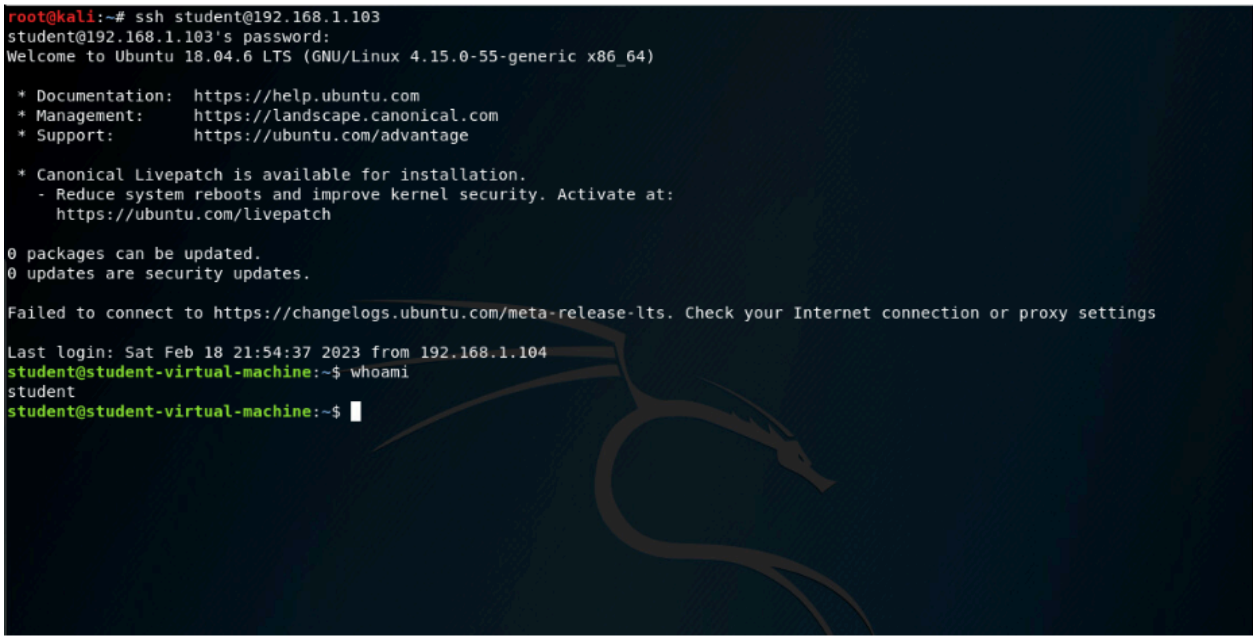


```
root@student-virtual-machine:~# iptables -F
root@student-virtual-machine:~# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
root@student-virtual-machine:~# iptables -A INPUT -j REJECT
root@student-virtual-machine:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
REJECT     all  --  anywhere             anywhere             reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@student-virtual-machine:~#
```

I flushed the iptables and put the rules in the reverse order this time.

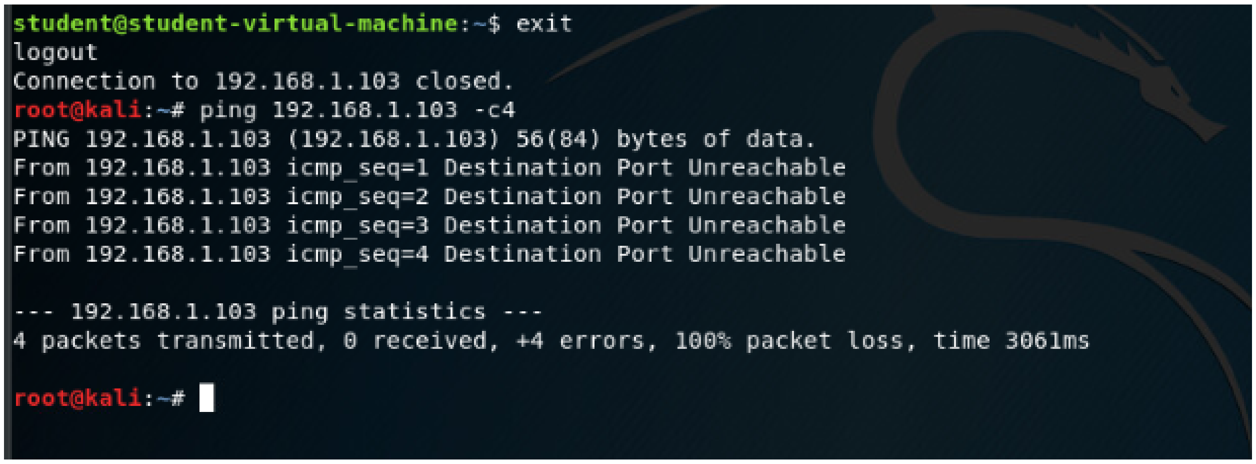
Will this work now?

Since the accept rule is being processed before the reject rule, I should be able to ssh into the ubuntu machine now.



```
root@kali:~# ssh student@192.168.1.103
student@192.168.1.103's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-55-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Sat Feb 18 21:54:37 2023 from 192.168.1.104
student@student-virtual-machine:~$ whoami
student
student@student-virtual-machine:~$
```

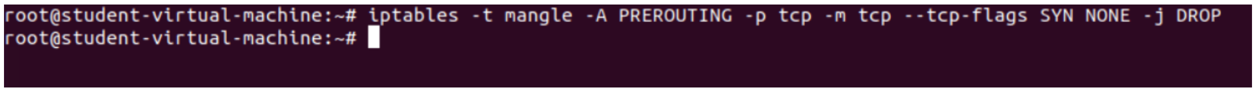
Surely, I was able to SSH into the ubuntu machine this time.



```
student@student-virtual-machine:~$ exit
logout
Connection to 192.168.1.103 closed.
root@kali:~# ping 192.168.1.103 -c4
PING 192.168.1.103 (192.168.1.103) 56(84) bytes of data.
From 192.168.1.103 icmp_seq=1 Destination Port Unreachable
From 192.168.1.103 icmp_seq=2 Destination Port Unreachable
From 192.168.1.103 icmp_seq=3 Destination Port Unreachable
From 192.168.1.103 icmp_seq=4 Destination Port Unreachable

--- 192.168.1.103 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3061ms

root@kali:~# 
```

I tried pinging after that to test that all other packets are rejected except for tcp on port 22. Surely it was the case.

Step 4



```
root@student-virtual-machine:~# iptables -t mangle -A PREROUTING -p tcp -m tcp --tcp-flags SYN NONE -j DROP
root@student-virtual-machine:~# █
```

What does this do?

The -t mangle options shows that this command will work with the mangle table which is used to alter packets.

-A PREROUTING means that we are adding a new rule to the prerouting chain which is a chain used to modify incoming packets before they route to their final destination.

The command is only going to apply to TCP packets since -p tcp is being used. The conditions for the packets are if the tcp packets have a SYN flag set to NONE meaning that the packet is not part of an established connection.

-j DROP in this command means that the machine will drop a TCP packet with a SYN flag set to NONE.

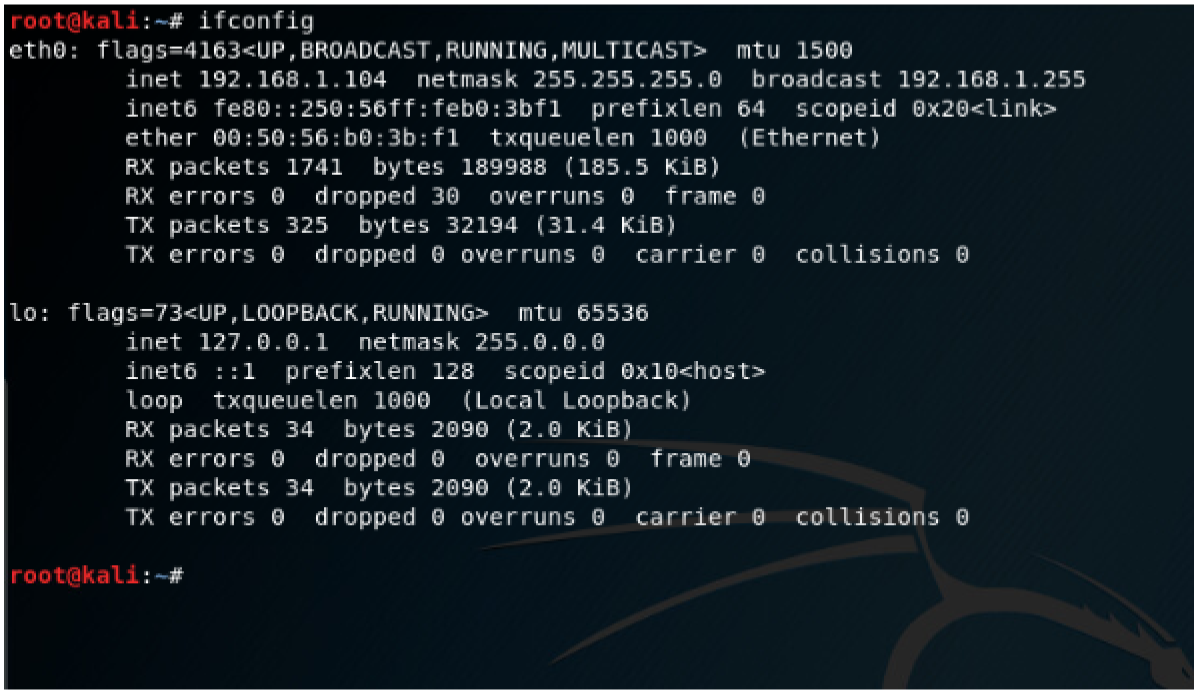
It is important to note that the -t mangle cooperation is not actually doing anything in the context of this command since the mangle table is associated with manipulation of packet headers. The command could do the same job without that operation included since there is no actual packet manipulation going on in this command. The command is simply dropping TCP packets that have a SYN flag set to NONE. It is not manipulating those packets.

Is this rule triggered before or after the other rules we see?

This command is in the prerouting chain which gets triggered before other chains in the mangle table. It is also in the mangles table which is processed before the filter table in the processing flow of iptables since advanced packet manipulation is done before filtering. The rule in this command will be triggered before all the other rules we issued before.

Step 5

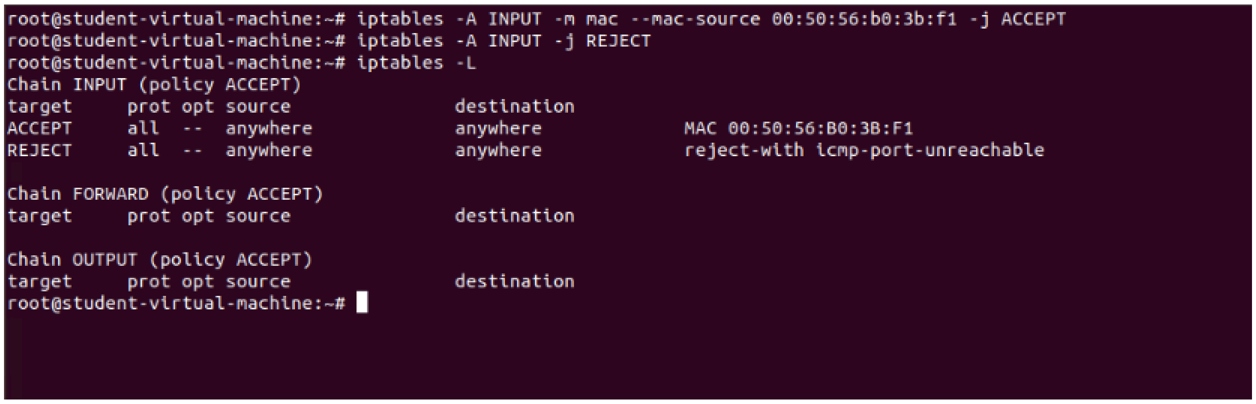
Finding the kali mac address



```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.104  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::250:56ff:feb0:3bf1  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:b0:3b:f1  txqueuelen 1000  (Ethernet)
        RX packets 1741  bytes 189988 (185.5 KiB)
        RX errors 0  dropped 30  overruns 0  frame 0
        TX packets 325  bytes 32194 (31.4 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 34  bytes 2090 (2.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 34  bytes 2090 (2.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@kali:~#
```

The mac address of the kali machine is 00:50:56:b0:3b:f1

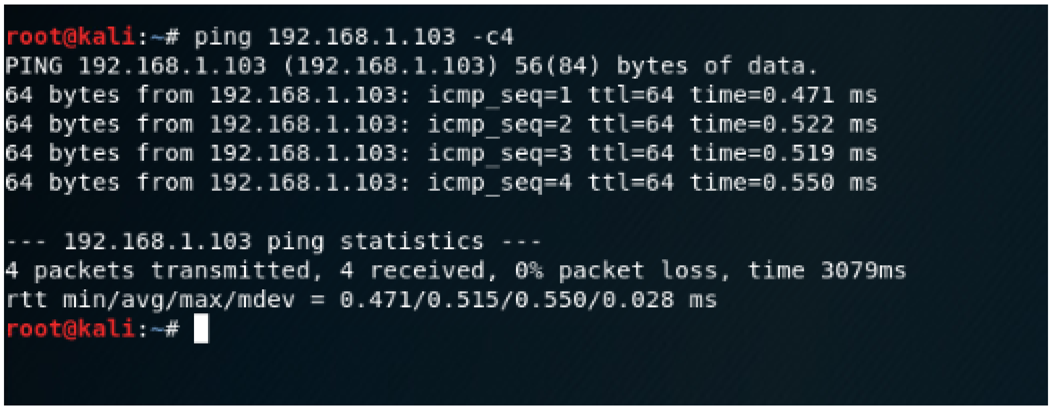
Setting up the filter table and the complete command



```
root@student-virtual-machine:~# iptables -A INPUT -m mac --mac-source 00:50:56:b0:3b:f1 -j ACCEPT
root@student-virtual-machine:~# iptables -A INPUT -j REJECT
root@student-virtual-machine:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             MAC 00:50:56:B0:3B:F1
REJECT     all  --  anywhere             anywhere             reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@student-virtual-machine:~# 
```

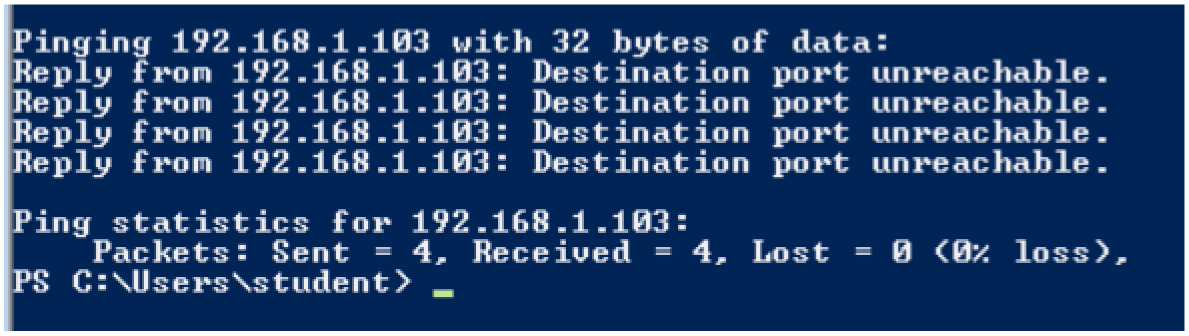
I used the complete command is the first command in this screenshot. I have the reject all packets rule set after that commands rule. This combination will make it so that only the kali machine can ping the ubuntu machine. All other machines will just have their ICMP packets rejected.

Testing the ruleset



```
root@kali:~# ping 192.168.1.103 -c4
PING 192.168.1.103 (192.168.1.103) 56(84) bytes of data.
64 bytes from 192.168.1.103: icmp_seq=1 ttl=64 time=0.471 ms
64 bytes from 192.168.1.103: icmp_seq=2 ttl=64 time=0.522 ms
64 bytes from 192.168.1.103: icmp_seq=3 ttl=64 time=0.519 ms
64 bytes from 192.168.1.103: icmp_seq=4 ttl=64 time=0.550 ms

--- 192.168.1.103 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3079ms
rtt min/avg/max/mdev = 0.471/0.515/0.550/0.028 ms
root@kali:~# 
```

The kali machine can definitely ping the ubuntu machine now unlike before.

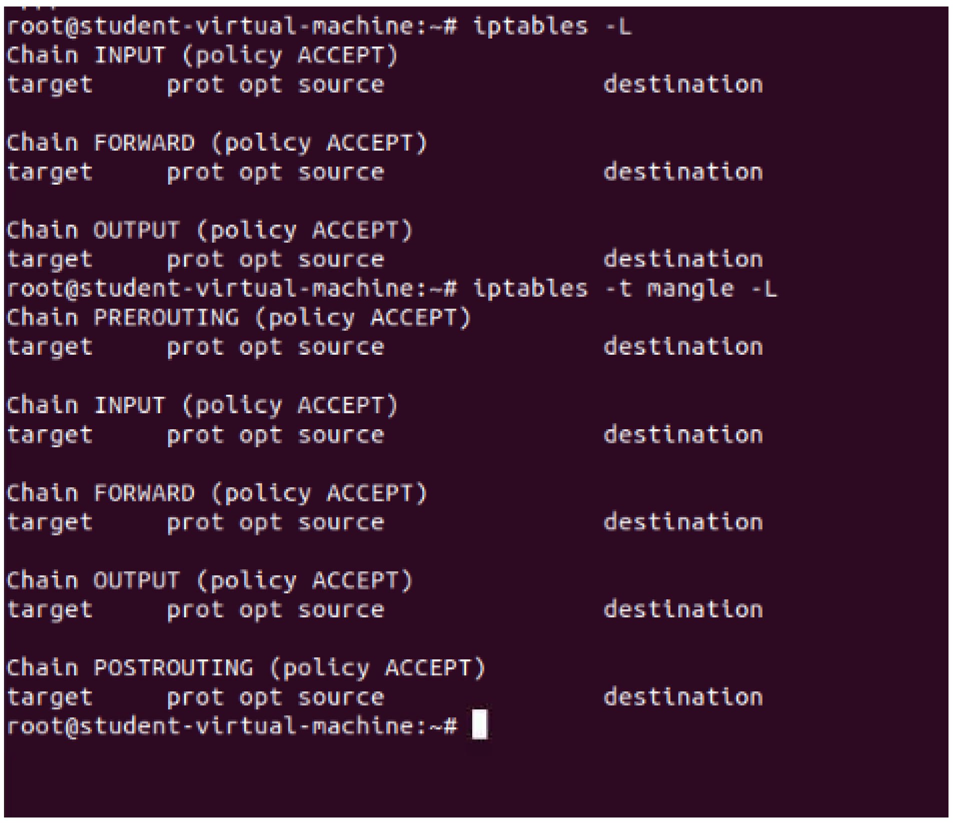


```
Pinging 192.168.1.103 with 32 bytes of data:
Reply from 192.168.1.103: Destination port unreachable.
Reply from 192.168.1.103: Destination port unreachable.
Reply from 192.168.1.103: Destination port unreachable.
Reply from 192.168.1.103: Destination port unreachable.

Ping statistics for 192.168.1.103:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
PS C:\Users\student> _
```

When I try to ping from the windows machine however, I get the port unreachable message.

Step 6

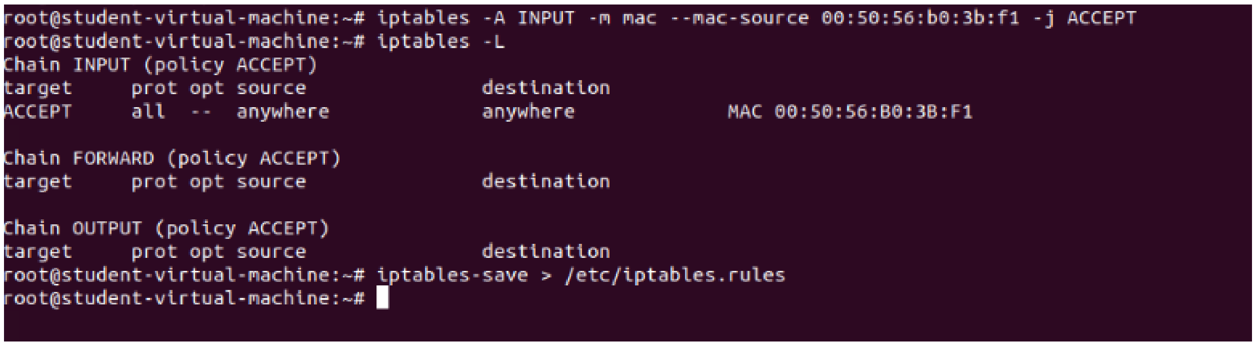
Rebooting and seeing what happens to the firewall



```
root@student-virtual-machine:~# 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
root@student-virtual-machine:~# iptables -t mangle -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

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

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
root@student-virtual-machine:~#
```

After rebooting, all the rules that I set in both filter and mangle tables are gone.

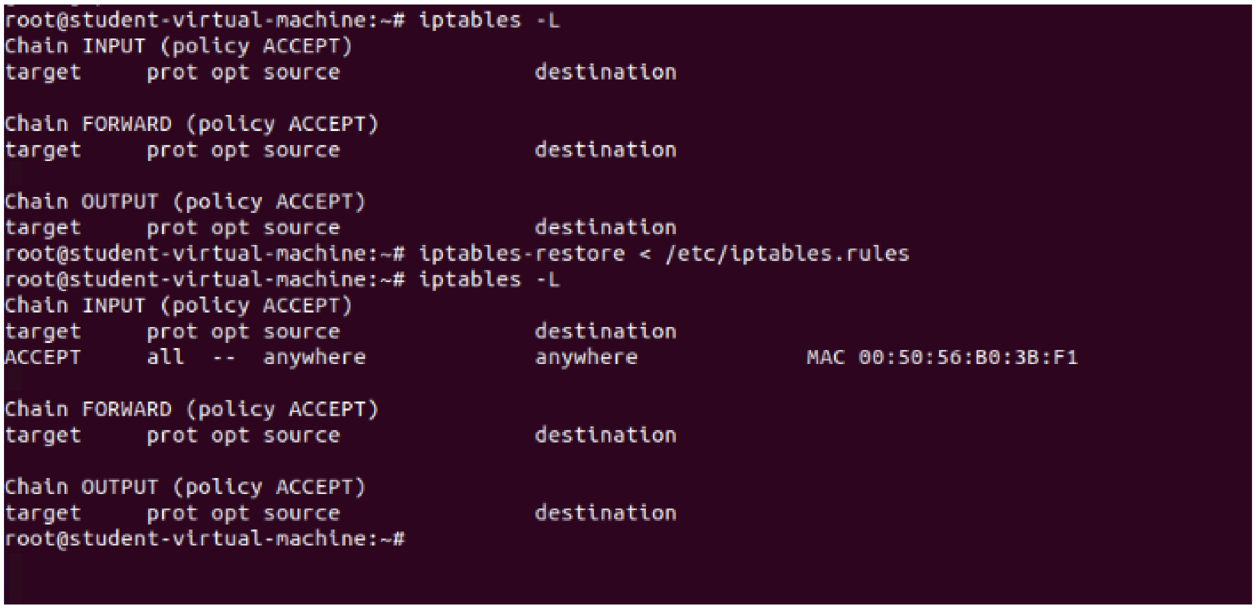
Adding a rule and saving it



```
root@student-virtual-machine:~# iptables -A INPUT -m mac --mac-source 00:50:56:b0:3b:f1 -j ACCEPT
root@student-virtual-machine:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             MAC 00:50:56:B0:3B:F1

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@student-virtual-machine:~# iptables-save > /etc/iptables.rules
root@student-virtual-machine:~# 
```

I re-added the rule where the kali machine can ping me and used the iptables save command. I then rebooted the machine again.

Rebooting and restoring rules



```
root@student-virtual-machine:~# 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
root@student-virtual-machine:~# iptables-restore < /etc/iptables.rules
root@student-virtual-machine:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             MAC 00:50:56:B0:3B:F1

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@student-virtual-machine:~#
```

After rebooting the machine, the rule was gone, but after I used the restore command the rule returned.

It is worth noting however that there is a package called iptables-persistent that allows saved rules to stay after reboot. That way iptable rules can always stay without the hassle of having to use crontab to run a script that restores the rules on reboot or manually doing it yourself.

References

https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-en-4/ch-iptables.html

https://www.digitalocean.com/community/tutorials/how-the-iptables-firewall-works

https://www.csie.ntu.edu.tw/~b93070/CNL/v4.0/CNLv4.0.files/Page697.htm

https://unix.stackexchange.com/questions/23060/what-level-of-the-network-stack-does-tcpdump-get-its-info-from

https://www.digitalocean.com/community/tutorials/how-to-list-and-delete-iptables-firewall-rules