University of Windsor

Course Name

Networking and Data Security (COMP-8677)

Document Type

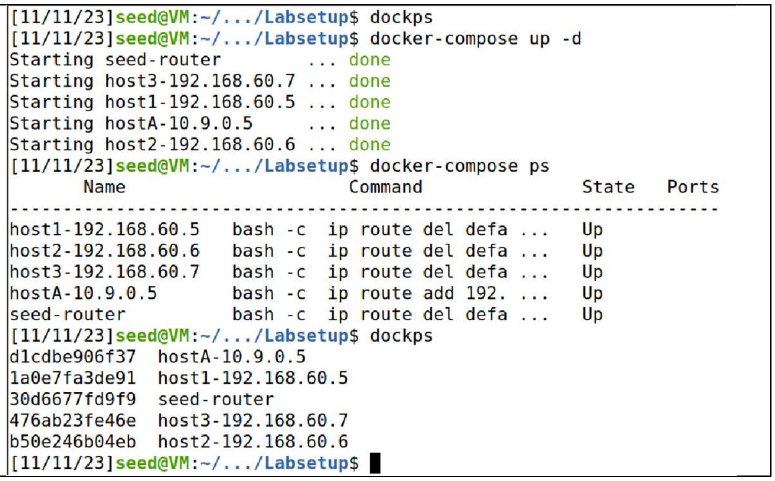
Lab Assignment 7

Professor

Dr. Shaoquan Jiang

Team - Members	Student ID
Manjinder Singh	110097177

Docker containers are up and running fine with the commands executed (shown in RHS).



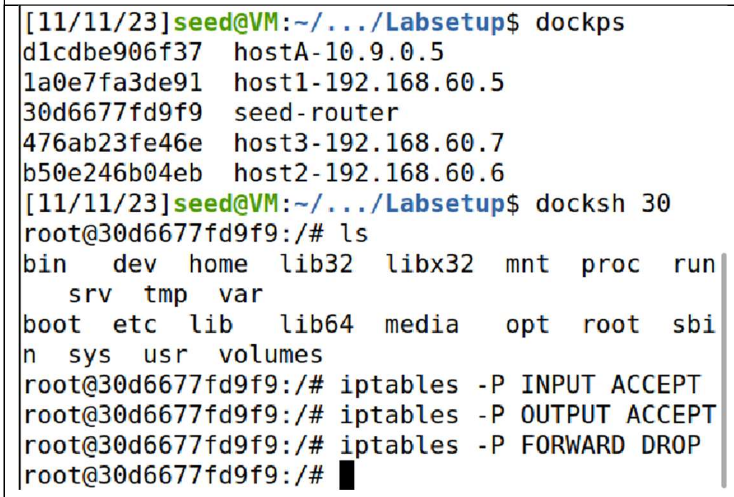
```
[11/11/23]seed@VM:~/.../Labsetup$ dockps
[11/11/23]seed@VM:~/.../Labsetup$ docker-compose up -d
Starting seed-router         ... done
Starting host3-192.168.60.7 ... done
Starting host1-192.168.60.5 ... done
Starting hostA-10.9.0.5      ... done
Starting host2-192.168.60.6 ... done
[11/11/23]seed@VM:~/.../Labsetup$ docker-compose ps
       Name                      Command               State   Ports
-------------------------------------------------------------------
host1-192.168.60.5   bash -c  ip route del defa ...   Up
host2-192.168.60.6   bash -c  ip route del defa ...   Up
host3-192.168.60.7   bash -c  ip route del defa ...   Up
hostA-10.9.0.5       bash -c  ip route add 192. ...   Up
seed-router          bash -c  ip route del defa ...   Up
[11/11/23]seed@VM:~/.../Labsetup$ dockps
d1cdbe906f37  hostA-10.9.0.5
1a0e7fa3de91  host1-192.168.60.5
30d6677fd9f9  seed-router
476ab23fe46e  host3-192.168.60.7
b50e246b04eb  host2-192.168.60.6
[11/11/23]seed@VM:~/.../Labsetup$ 
```

1. Use the following commands on **router** to set the default policies for a table**.**
sudo iptables –P INPUT ACCEPT
sudo iptables –P OUTPUT ACCEPT
sudo iptables –P FORWARD DROP
Recall, INPUT is to check incoming packet; OUTPUT is to check outgoing packet; FORWARDING is to check the passing packet (at router). Further, the commands assume the default table **filter (-t filter)**.

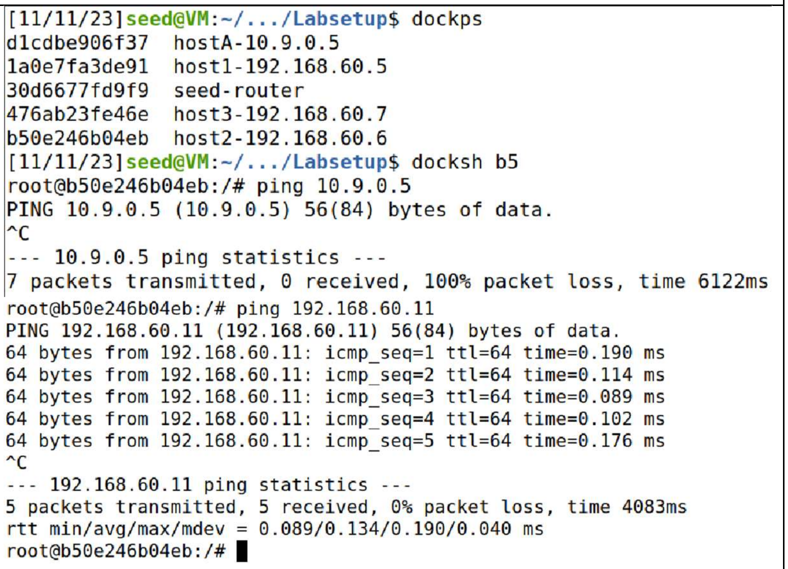
- On 192.168.60.6, run **$ ping 10.9.0.5** and then ping 192.168.60.11. Does it succeed? Explain your observation.
- Change **DROP** to **ACCEPT,** for FORWARD case. Try the pings in the above step again. Now does it succeed?

My Implementation of above question 1:-

Commands executed in	
Terminal 1	**Terminal 2**
docksh 30 ls iptables -P INPUT ACCEPT iptables -P OUTPUT ACCEPT iptables -P FORWARD DROP	dockps dockcsh b5 ping 10.9.0.5 ping 192.168.60.11



```
[11/11/23]seed@VM:~/.../Labsetup$ dockps
d1cdbe906f37  hostA-10.9.0.5
1a0e7fa3de91  host1-192.168.60.5
30d6677fd9f9  seed-router
476ab23fe46e  host3-192.168.60.7
b50e246b04eb  host2-192.168.60.6
[11/11/23]seed@VM:~/.../Labsetup$ docksh 30
root@30d6677fd9f9:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run
   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbi
n  sys  usr  volumes
root@30d6677fd9f9:/# iptables -P INPUT ACCEPT
root@30d6677fd9f9:/# iptables -P OUTPUT ACCEPT
root@30d6677fd9f9:/# iptables -P FORWARD DROP
root@30d6677fd9f9:/# 
```



```
[11/11/23]seed@VM:~/.../Labsetup$ dockps
d1cdbe906f37  hostA-10.9.0.5
1a0e7fa3de91  host1-192.168.60.5
30d6677fd9f9  seed-router
476ab23fe46e  host3-192.168.60.7
b50e246b04eb  host2-192.168.60.6
[11/11/23]seed@VM:~/.../Labsetup$ docksh b5
root@b50e246b04eb:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
^C
--- 10.9.0.5 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6122ms
root@b50e246b04eb:/# ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.190 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.114 ms
64 bytes from 192.168.60.11: icmp_seq=3 ttl=64 time=0.089 ms
64 bytes from 192.168.60.11: icmp_seq=4 ttl=64 time=0.102 ms
64 bytes from 192.168.60.11: icmp_seq=5 ttl=64 time=0.176 ms
^C
--- 192.168.60.11 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4083ms
rtt min/avg/max/mdev = 0.089/0.134/0.190/0.040 ms
root@b50e246b04eb:/# 
```

Observations:

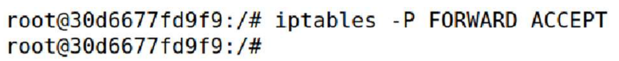
After executing the iptables command on seed-router in Terminal, we observed the below results in Terminal 2:

- **Ping to 10.9.0.5 (hostA):** In the "host2-192.168.60.6" container, the ping to "10.9.0.5" results in 100% packet loss. This is due to the FORWARD chain in the "seed-router" container being set to **DROP**, preventing the forwarding of packets between containers.
- **Ping to 192.168.60.11:** The ping to "192.168.60.11" is successful with 0% packet loss. This signifies that the "192.168.60.11" IP address is reachable from the "host2-192.168.60.6" container. The successful ping indicates that the network allows communication between the containers, or the target IP is likely on the same container host.

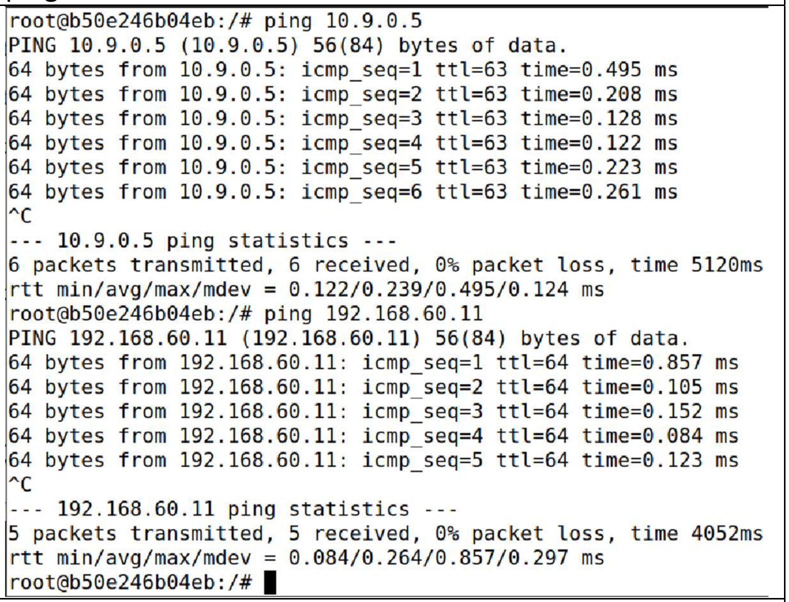
- To summarize the above, pinging "10.9.0.5" fails due to the DROP policy in the FORWARD chain of the "seed-router" container whereas pinging "192.168.60.11" is successful, indicating that the network allows communication between containers or the target IP is on the same host and are not subject to the FORWARD chain policies.

Now, Changing **DROP** to **ACCEPT,** for FORWARD case and Trying the pings in the above step again.

iptables -P FORWARD ACCEPT	ping 10.9.0.5 ping 192.168.60.11



```
root@30d6677fd9f9:/# iptables -P FORWARD ACCEPT
root@30d6677fd9f9:/#
```



```
root@b50e246b04eb:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.495 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.208 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.128 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.122 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.223 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=63 time=0.261 ms
^C
--- 10.9.0.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5120ms
rtt min/avg/max/mdev = 0.122/0.239/0.495/0.124 ms
root@b50e246b04eb:/# ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.857 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.105 ms
64 bytes from 192.168.60.11: icmp_seq=3 ttl=64 time=0.152 ms
64 bytes from 192.168.60.11: icmp_seq=4 ttl=64 time=0.084 ms
64 bytes from 192.168.60.11: icmp_seq=5 ttl=64 time=0.123 ms
^C
--- 192.168.60.11 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4052ms
rtt min/avg/max/mdev = 0.084/0.264/0.857/0.297 ms
root@b50e246b04eb:/#
```

Observations:

- **Ping to 10.9.0.5 (hostA):** The change in the FORWARD chain policy to ACCEPT allows the "host2-192.168.60.6" container to successfully communicate with the "hostA-10.9.0.5" container. All packets are received without any loss.
- **Ping to 192.168.60.11:** Similar to the previous observation mentioned in "**Ping to 10.9.0.5 (hostA)**", the change in the FORWARD chain policy to ACCEPT enables successful communication with the specified IP address "192.168.60.11." All packets are received without any loss.

To summarize, Changing the FORWARD chain policy to ACCEPT has resolved the packet loss issue, and now both pings are successful. The containers can communicate with each other without restrictions imposed by the FORWARD chain policies.

2. [**blocking an IP**]

• On 192.168.60.11, if we want to block packets **from** an ip address IP1, use command

sudo iptables -A INPUT -s IP1 -j DROP

/*this uses INPUT chain because it is incoming packet***/**

On IP1, ping 192.168.60.11 and what can be observed? Explain.

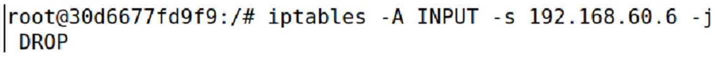
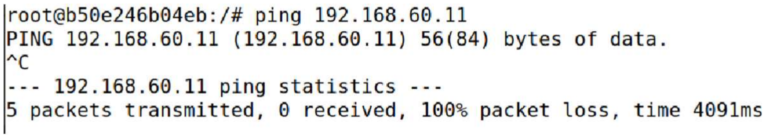
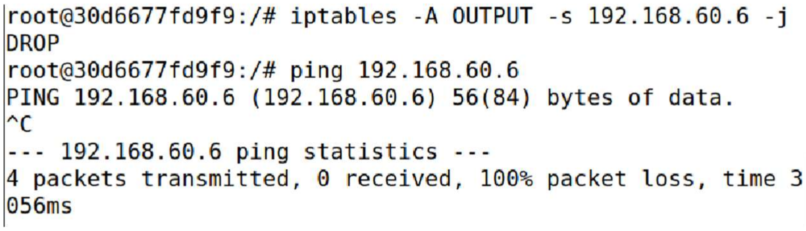
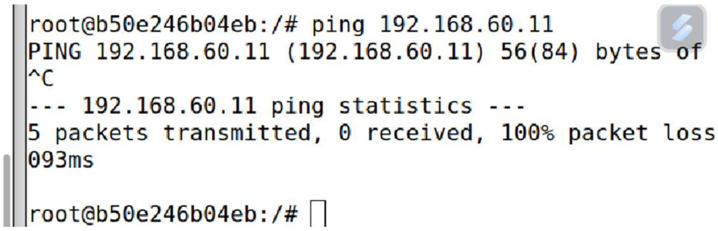
- On 192.168.60.11, if we want to block packets **to** an ip address IP1, use command

sudo iptables -A OUTPUT -d IP1 -j DROP

/*this uses OUTPUT chain because it is outgoing packet***/**

On 192.168.60.11, ping IP1 and what can be observed? Explain.

My Implementation of above question 2:-

iptables -A INPUT -s 192.168.60.6 -j DROP	ping 192.168.60.11
 root@30d6677fd9f9:/# iptables -A INPUT -s 192.168.60.6 -j DROP	 root@b50e246b04eb:/# ping 192.168.60.11 PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data. ^C --- 192.168.60.11 ping statistics --- 5 packets transmitted, 0 received, 100% packet loss, time 4091ms
Observation from above execution of commands: As per the above commands, we blocked the 192.168.60.6 on router, and if we try to ping from 192.168.60.6 to router then all the packets will be lost, and there will be no received packets on router end.	
iptables -A OUTPUT -s 192.168.60.6 -j DROP ping 192.168.60.6	ping 192.168.60.11
 root@30d6677fd9f9:/# iptables -A OUTPUT -s 192.168.60.6 -j DROP root@30d6677fd9f9:/# ping 192.168.60.6 PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data. ^C --- 192.168.60.6 ping statistics --- 4 packets transmitted, 0 received, 100% packet loss, time 3 056ms	 root@b50e246b04eb:/# ping 192.168.60.11 PING 192.168.60.11 (192.168.60.11) 56(84) bytes of ^C --- 192.168.60.11 ping statistics --- 5 packets transmitted, 0 received, 100% packet loss 093ms root@b50e246b04eb:/#
Observation from above execution of commands: As per the above commands, we blocked the OUTPUT to the IP Address - 192.168.60.6, so when we are trying to send packets from router to this IP, it will drop all the packets.	

3. [List all rules] do it on Router.

- You can see all the firewall rules by the following command

$ sudo iptables -L

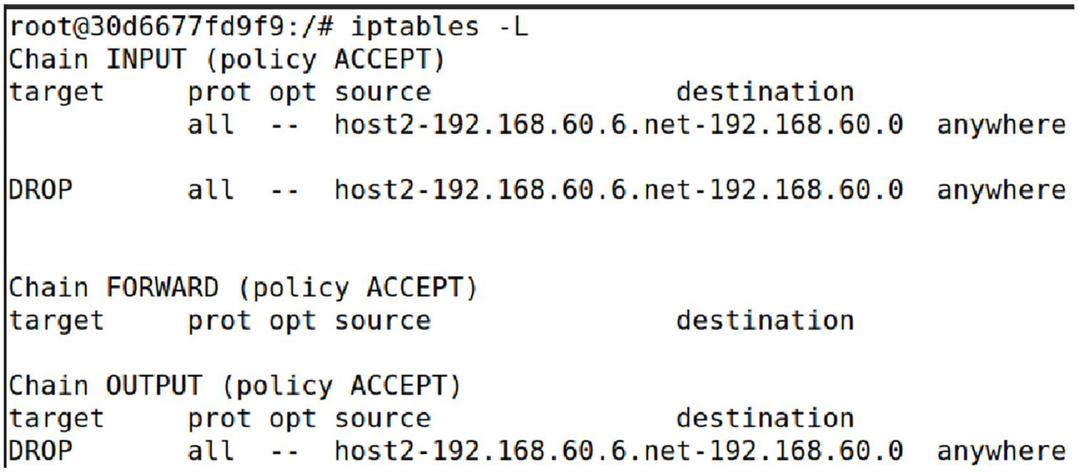
/* again, this assume filter table (i.e., **-t filter**) by default*/

- You can see all the fire rules in each chain with index number. The index will be used for other operation such as deletion later.

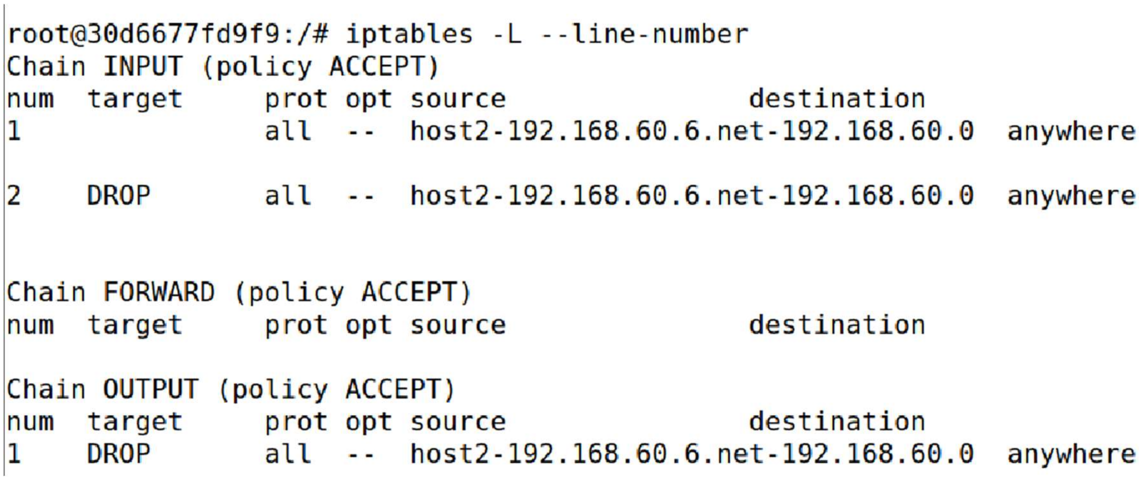
$ sudo iptables -L --line-number

My Implementation of above question 3 Part:-

iptables -L (Helps to view firewall rules)
iptables -L --line-number (Helps to view firewall rules in each chain with index number)



```
root@30d6677fd9f9:/# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
           all  --  host2-192.168.60.6.net-192.168.60.0  anywhere

DROP       all  --  host2-192.168.60.6.net-192.168.60.0  anywhere


Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  host2-192.168.60.6.net-192.168.60.0  anywhere
```



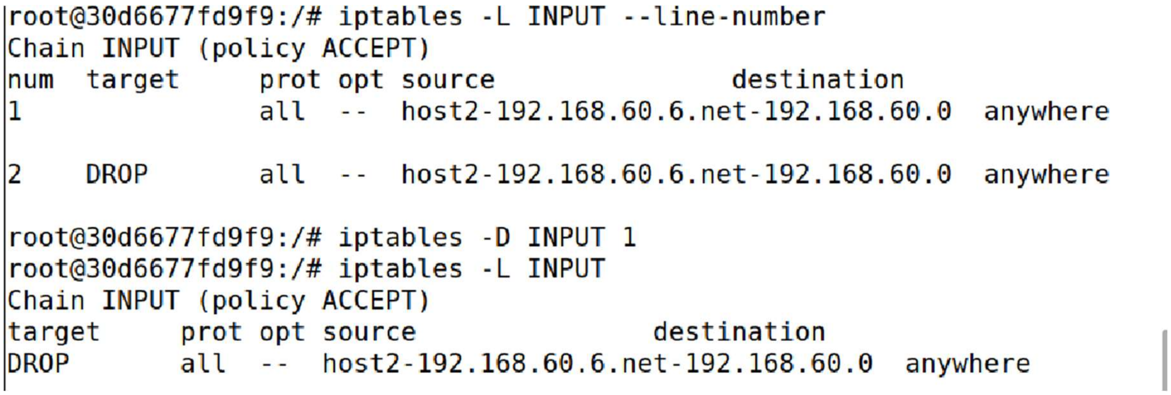
```
root@30d6677fd9f9:/# iptables -L --line-number
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1               all  --  host2-192.168.60.6.net-192.168.60.0  anywhere

2    DROP       all  --  host2-192.168.60.6.net-192.168.60.0  anywhere


Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
1    DROP       all  --  host2-192.168.60.6.net-192.168.60.0  anywhere
```

4. [Delete a rule] on **Router,** delete a rule in a chain (such as INPUT) in two steps:
first, list with index:
$ sudo iptables –L INPUT --line-number
Then, delete the rule using the index:
$sudo iptables -D INPUT 1
Now use the method to delete the first rule in your current INPUT table and then
$sudo iptables -L INPUT to verify whether rule 1 is deleted or not.

My Implementation of above question 4 Part:-

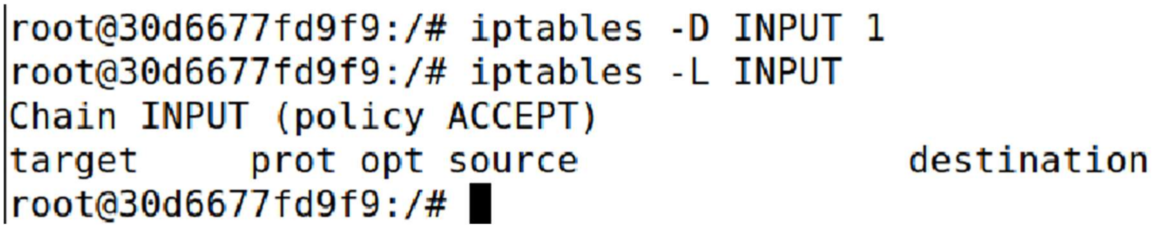
i. iptables -L INPUT --line-number
ii. iptables -D INPUT 1
iii. iptables -L INPUT

The above commands are used to display, manage, and update the firewall rules for incoming network traffic on a Linux system.
The first command shows the rules with line numbers,
the second command deletes a specific rule, and
the third command shows the updated list of rules after the deletion.



```
root@30d6677fd9f9:/# iptables -L INPUT --line-number
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1               all  --  host2-192.168.60.6.net-192.168.60.0  anywhere

2    DROP       all  --  host2-192.168.60.6.net-192.168.60.0  anywhere

root@30d6677fd9f9:/# iptables -D INPUT 1
root@30d6677fd9f9:/# iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  host2-192.168.60.6.net-192.168.60.0  anywhere
```

Again we deleted the remaining rule by using the below commands:-
iptables -D INPUT 1
iptables -L INPUT



```
root@30d6677fd9f9:/# iptables -D INPUT 1
root@30d6677fd9f9:/# iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
root@30d6677fd9f9:/# █
```

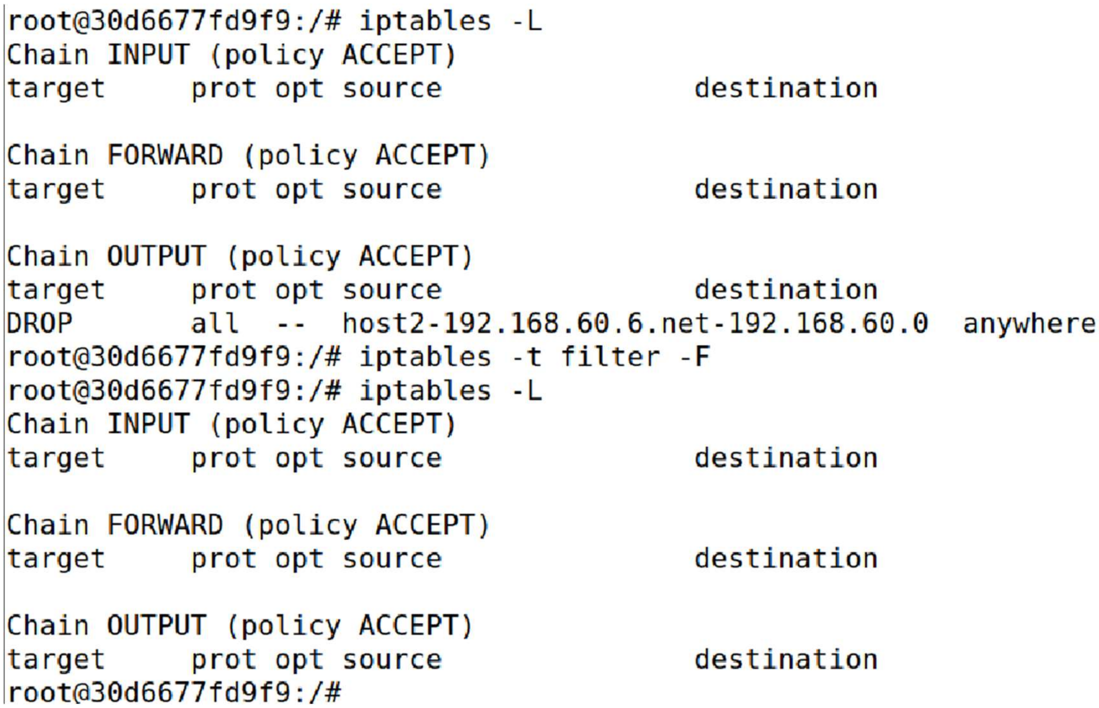
5.[Delete all rules in a TABLE] On **router,** flush the rules in a table (e.g., **filter**):
$sudo iptables -t filter -F
/*again,**-t filter** can be omitted*/
Then, run **$sudo iptables -L** and you will not see any rule.

My Implementation of above question 5 Part:-

i. iptables -L (First we will preview)
ii. iptables -t filter -F
iii. iptables -L

The above written commands are used to (i) preview the existing firewall rules, (ii) then clear all the rules from the filter table, and (iii) finally, show the updated list of rules after the flush operation.

It's a way to start with a clean slate for configuring new firewall rules.



```
root@30d6677fd9f9:/# 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
DROP       all  --  host2-192.168.60.6.net-192.168.60.0  anywhere
root@30d6677fd9f9:/# iptables -t filter -F
root@30d6677fd9f9:/# 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@30d6677fd9f9:/#
```

6. [**Drop all incoming connections, except telnet**] On **router**, block incoming connections to any service except for telnet. To do this, we can set default policy for INPUT chain of filter Table to be DROP and then specify a rule to accept incoming telnet connection.

$ sudo iptables -P INPUT DROP

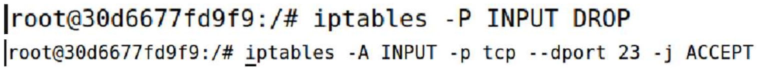
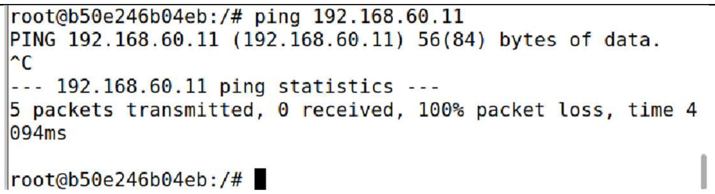
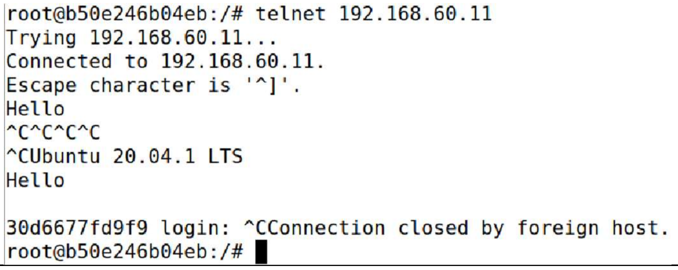
$ sudo iptables -A INPUT -p tcp - -dport 23 -j ACCEPT

/* A default policy is applied only if all the rules in the chain have been executed without making a decision (either ACCEPT or DROP or REJECT). For example, if we ssh to router, then the rule does not ACCEPT but also not REJECT. So the default policy applies. Note: here **-p** stands for protocol. */

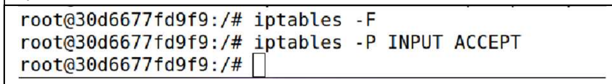
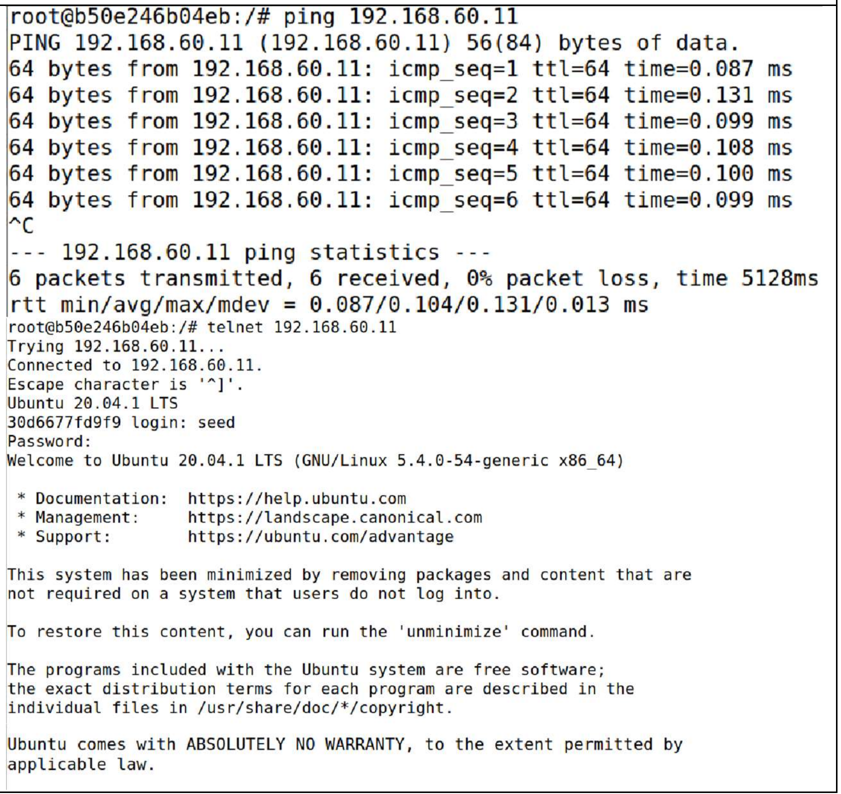
Then, ping and telnet to 192.168.60.11 (from other VM). Which succeeds (telnet or ping)?

/*after this problem, run **$ sudo iptables -F** to flush all rules in **filter** table and recover the default policy: **$ sudo iptables -P INPUT ACCEPT** */

My Implementation of above question 6 Part:-

iptables -P INPUT DROP iptables -A INPUT -p tcp –dport 23 -j ACCEPT	ping 192.168.60.11 telnet 192.168.60.11
 root@30d6677fd9f9:/# iptables -P INPUT DROP root@30d6677fd9f9:/# iptables -A INPUT -p tcp --dport 23 -j ACCEPT	 root@b50e246b04eb:/# ping 192.168.60.11 PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data. ^C --- 192.168.60.11 ping statistics --- 5 packets transmitted, 0 received, 100% packet loss, time 4 094ms root@b50e246b04eb:/#  root@b50e246b04eb:/# telnet 192.168.60.11 Trying 192.168.60.11... Connected to 192.168.60.11. Escape character is '^]'. Hello ^C^C^C^C ^CUbuntu 20.04.1 LTS Hello 30d6677fd9f9 login: ^CConnection closed by foreign host. root@b50e246b04eb:/#

Observation -Telnet to "192.168.60.11" from the "host2-192.168.60.6" container succeeds. The connection is established, and the prompt is received.

iptables -F iptables -P INPUT ACCEPT	ping 192.168.60.11 telnet 192.168.60.11
	

```
root@30d6677fd9f9:/# iptables -F
root@30d6677fd9f9:/# iptables -P INPUT ACCEPT
root@30d6677fd9f9:/# 
```

```
root@b50e246b04eb:/# ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.087 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.131 ms
64 bytes from 192.168.60.11: icmp_seq=3 ttl=64 time=0.099 ms
64 bytes from 192.168.60.11: icmp_seq=4 ttl=64 time=0.108 ms
64 bytes from 192.168.60.11: icmp_seq=5 ttl=64 time=0.100 ms
64 bytes from 192.168.60.11: icmp_seq=6 ttl=64 time=0.099 ms
^C
--- 192.168.60.11 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5128ms
rtt min/avg/max/mdev = 0.087/0.104/0.131/0.013 ms
root@b50e246b04eb:/# telnet 192.168.60.11
Trying 192.168.60.11...
Connected to 192.168.60.11.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
30d6677fd9f9 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

Observation - After executing **iptables -F** and **iptables -P INPUT ACCEPT commands**, all the incoming traffic is allowed. Therefore, both the ping and telnet commands to 192.168.60.11 are succeeded while assuming there are no network issues or restrictions on the target host.

The firewall has been effectively disabled by allowing all incoming traffic, so both ping and telnet worked as per above screenshot without any restrictions.

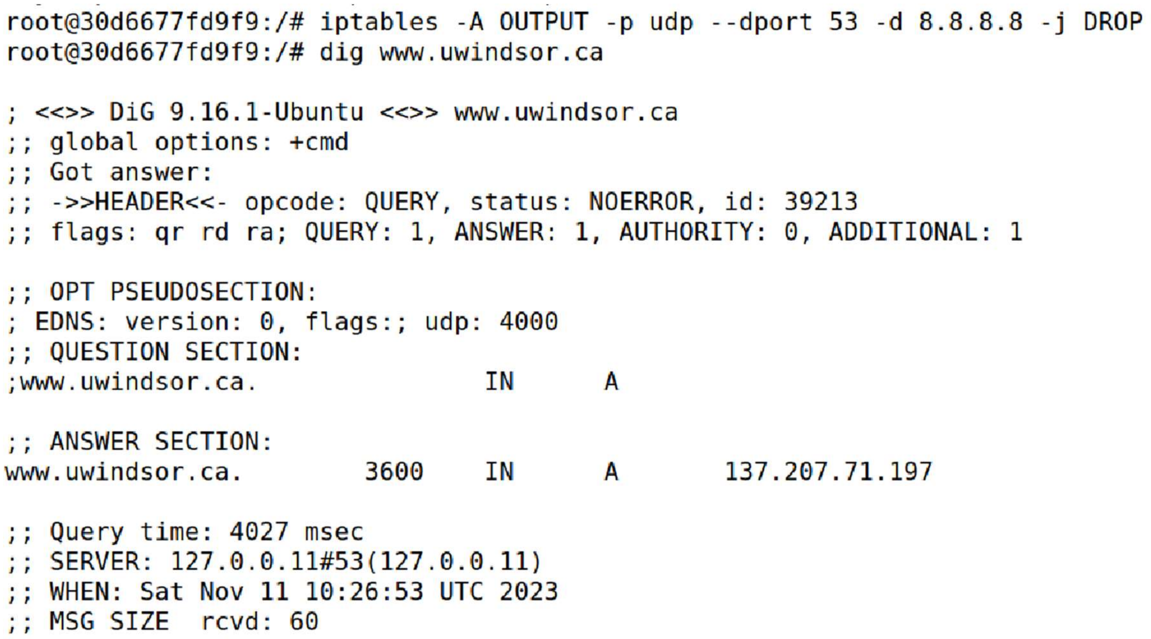
7. [**drop outgoing DNS request to 8.8.8.8**] In this case, since it is outgoing packet, we add rule to OUTPUT chain. Since it is DNS request, the destination should be the DNS server, which has a port number 53. Finally, since DNS is implemented using UDP, we use protocol UDP. Hence, we add the following rule:
$ sudo iptables -A OUTPUT -p udp - - dport 53 -d 8.8.8.8 -j DROP
Then, try **$ dig www.uwindsor.ca** and **dig @8.8.8.8 www.uwindsor.ca**. Which succeeds?
/* delete the rule in order not to affect the following experiment */

My Implementation of above question 7 Part:-

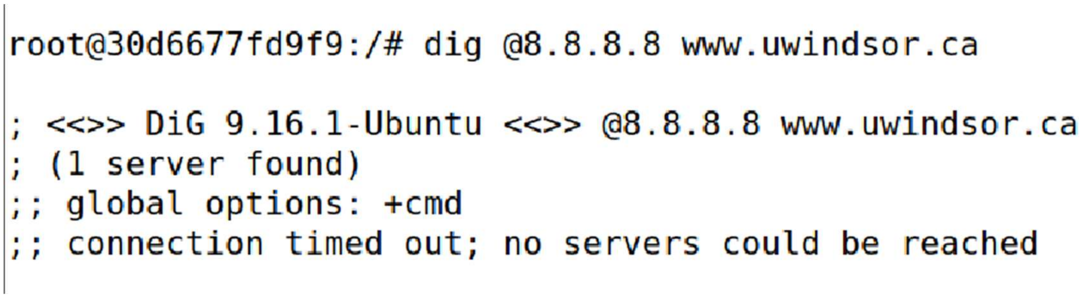
iptables -A OUTPUT -p udp - - dport 53 -d 8.8.8.8 -j DROP

dig www.uwindsor.ca

dig @8.8.8.8 www.uwindsor.ca



```
root@30d6677fd9f9:/# iptables -A OUTPUT -p udp --dport 53 -d 8.8.8.8 -j DROP
root@30d6677fd9f9:/# dig www.uwindsor.ca

; <<>> DiG 9.16.1-Ubuntu <<>> www.uwindsor.ca
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39213
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
;www.uwindsor.ca.               IN      A

;; ANSWER SECTION:
www.uwindsor.ca.        3600    IN      A       137.207.71.197

;; Query time: 4027 msec
;; SERVER: 127.0.0.11#53(127.0.0.11)
;; WHEN: Sat Nov 11 10:26:53 UTC 2023
;; MSG SIZE  rcvd: 60
```



```
root@30d6677fd9f9:/# dig @8.8.8.8 www.uwindsor.ca

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 www.uwindsor.ca
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

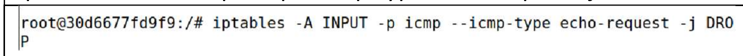
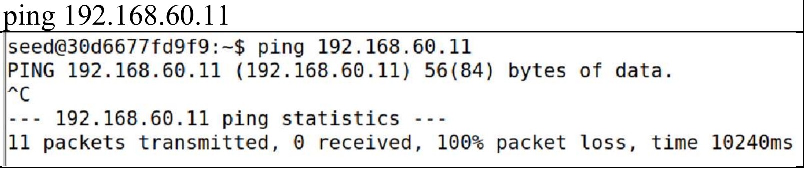
dig www.uwindsor.ca - succeeds where dig @8.8.8.8 www.uwindsor.ca - connection timed out.

Reason - The first dig command (dig www.uwindsor.ca) succeeds because it uses the local DNS resolver (127.0.0.11), and the iptables rule only blocks traffic to the external DNS server ("8.8.8.8").

The second dig command (dig @8.8.8.8 www.uwindsor.ca) fails due to the iptables rule, as it explicitly attempts to reach an external DNS server ("8.8.8.8"), which is blocked by the rule.

8. [**block incoming ping request**] You can not ping uwindsor webserver. Most likely, this is blocked by firewall of uwindsor. Here is the way to block an incoming icmp request.
$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
Run this on **router** and ping router from another VM. Do you get any reply? Explain.

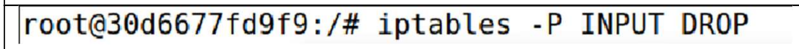
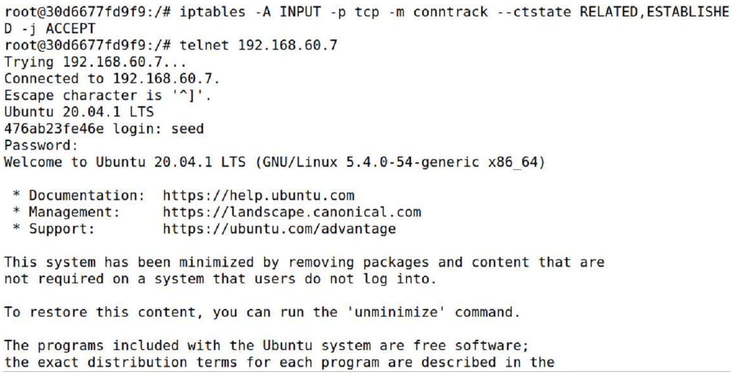
My Implementation of above question 8 Part:-

iptables -A INPUT -p icmp --icmp-type echo-request -j DROP	ping 192.168.60.11
 root@30d6677fd9f9:/# iptables -A INPUT -p icmp --icmp-type echo-request -j DRO P	 seed@30d6677fd9f9:~$ ping 192.168.60.11 PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data. ^C --- 192.168.60.11 ping statistics --- 11 packets transmitted, 0 received, 100% packet loss, time 10240ms

Explanation - The **iptables** rule specifically targets incoming ICMP echo requests (**--icmp-type echo-request**) and drops them (**-j DROP**). As a result, when attempting to ping the router from another VM, the router will not respond to the ICMP echo requests, i.e. , the ping requests will not receive any replies due to the **iptables** rule blocking incoming ICMP echo requests on the router.

9. Suppose that you want to block all incoming connections while you do not want your visit to external servers to be affected. However, if you send a request to an external server, the server will reply to you while this packet will be blocked by your firewall. To resolve this issue, you should regard the response packet (to your request) as related to your outgoing request packet and allowed to come in. This is achieved using the *conntrack* module.
$ sudo iptables -P INPUT DROP
$ sudo iptables -A INPUT -p tcp -m conntrack --ctstate RELATED, ESTABLISHED -j ACCEPT
Try this on **router** VM. Then, telnet to a VM (e.g. 192.168.60.7).
Next, telnet from the latter (192.168.60.7) to **router**. Which telnet session directly succeeds?

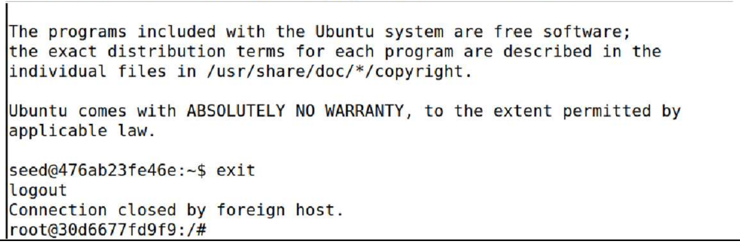
My Implementation of above question 9 Part:-

<table>
<tr><td></td><td></td></tr>
<tr><td>iptables -P INPUT DROP
iptables -A INPUT -p tcp -m conntrack --ctstate RELATED, ESTABLISHED -j ACCEPT
telnet 192.168.60.7 [# Attempt telnet from the router to another VM (e.g., 192.168.60.7)]</td><td></td></tr>
<tr><td>

<pre>
root@30d6677fd9f9:/# iptables -P INPUT DROP
</pre>

<pre>
root@30d6677fd9f9:/# iptables -A INPUT -p tcp -m conntrack --ctstate RELATED,ESTABLISHE
D -j ACCEPT
root@30d6677fd9f9:/# telnet 192.168.60.7
Trying 192.168.60.7...
Connected to 192.168.60.7.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
476ab23fe46e login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

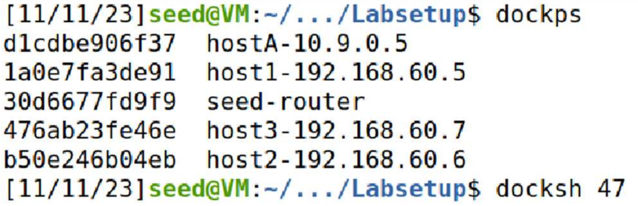
 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

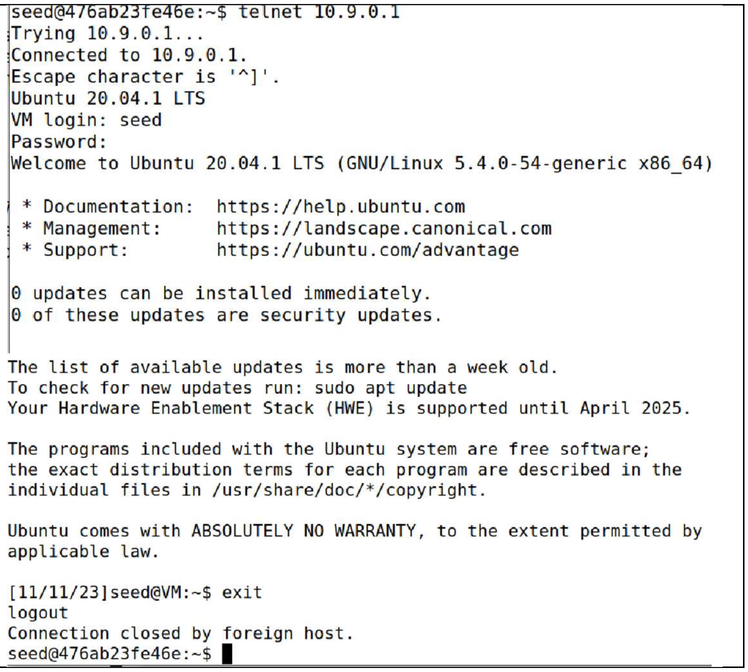
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
</pre>

<pre>
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@476ab23fe46e:~$ exit
logout
Connection closed by foreign host.
root@30d6677fd9f9:/#
</pre>
</td><td></td></tr>
<tr><td></td><td>telnet 10.9.0.1 [# Attempt telnet from another VM (e.g., 192.168.60.7) to the router]</td></tr>
<tr><td></td><td>

<pre>
[11/11/23]seed@VM:~/.../Labsetup$ dockps
d1cdbe906f37 hostA-10.9.0.5
1a0e7fa3de91 host1-192.168.60.5
30d6677fd9f9 seed-router
476ab23fe46e host3-192.168.60.7
b50e246b04eb host2-192.168.60.6
[11/11/23]seed@VM:~/.../Labsetup$ docksh 47
</pre>
</td></tr>
</table>



```
seed@476ab23fe46e:~$ telnet 10.9.0.1
Trying 10.9.0.1...
Connected to 10.9.0.1.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
VM login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

0 updates can be installed immediately.
0 of these updates are security updates.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

[11/11/23]seed@VM:~$ exit
logout
Connection closed by foreign host.
seed@476ab23fe46e:~$ █
```

Explanation – Both telnet sessions directly succeed. The iptables rules ensure that responses related to established connections are allowed through the firewall, providing the necessary bidirectional communication for telnet sessions to work.

This configuration allows outgoing connections to external servers and permits the corresponding response packets to come back in, ensuring that your visits to external servers are not affected by the incoming connection restrictions.

10. (optional) [save your firewall rules and restore it] After you have done firewall, you want to save your rules to a file you can run

$ sudo iptables-save >myiptables.rules

Later, you can restore your rules by running

$ sudo iptables-restore <myiptables.rules

/* to see the effect, you can flush your firewall after running iptables-save command and then run iptables-restore command to see if you have restored your firewall */

My Implementation of above question 10 Part:-

i. iptables-save > myiptables.rules
ii. iptables -F
iii. iptables-restore < myiptables.rules

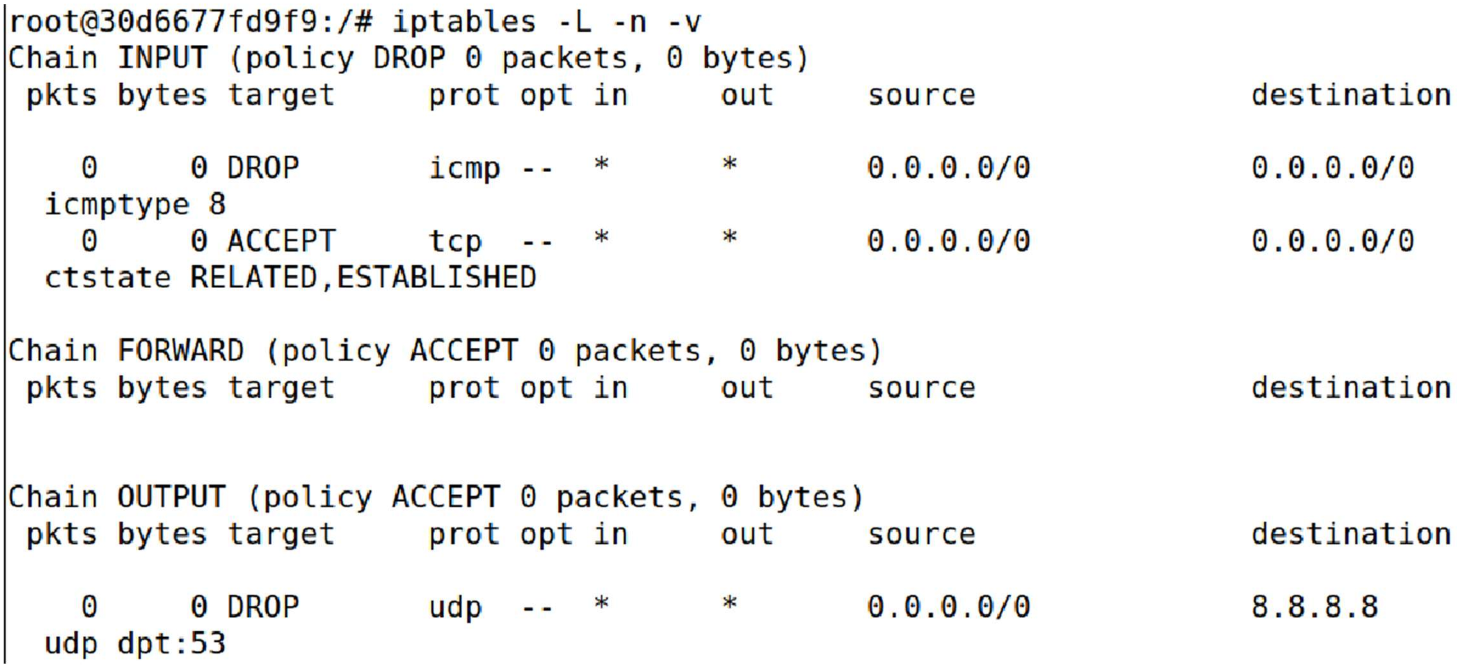
The first command saves the current iptables rules to a file.

The second command clears (flushes) all existing firewall rules.

The third command restores the previously saved rules from the file, effectively applying the saved configuration.

The above sequence of commands is a way to temporarily disable the firewall (by clearing rules) and later restore it to a known state using the saved rules. It can be useful for testing or managing firewall configurations.

Displaying firewall rules after restoring :



```
root@30d6677fd9f9:/# iptables -L -n -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 DROP       icmp --  *      *       0.0.0.0/0            0.0.0.0/0
  icmptype 8
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
  ctstate RELATED,ESTABLISHED

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 DROP       udp  --  *      *       0.0.0.0/0            8.8.8.8
  udp dpt:53
```

References: -

1. Lab Manual for Lab 7 from Brightspace
2. Lecture Notes for Lab 7 from Brightspace

One Drive Link for Lab 7 Solution(Word File and PDF Document) for Lab 7 Work:-

Networking and Data Security - Lab 7 - Submitted to Doc