

EN4720: Security in Cyber-Physical Systems

Exercise — Infrastructure Security

Name: Thalagala B. P.

Index No: 180631J

June 19, 2023

This is an individual exercise!
Due Date: 20 June 2023 by 11.59 PM

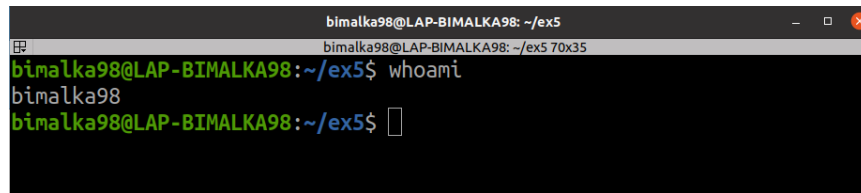
This exercise has to be carried out using a Linux-based PC/virtual machine. Read all the instructions and questions before attempting the exercise. Add answers under each question and submit the resulting PDF.

Section 1

In this section, you will implement Firewall rules using **iptables** and **ufw** Linux commands. Moreover, you will scan network ports of a remote device using **nmap** Linux command.

For all the questions in this section, add a screenshot of the terminal (including all the commands you ran to perform the task) unless specified otherwise. The evaluator should be able to see each step that you followed to perform each task. In all screenshots, the areas marked (which are unique to your terminal display) in Figure 1 (the sample answer to Question 1) must be visible.

1. View the currently logged in user.

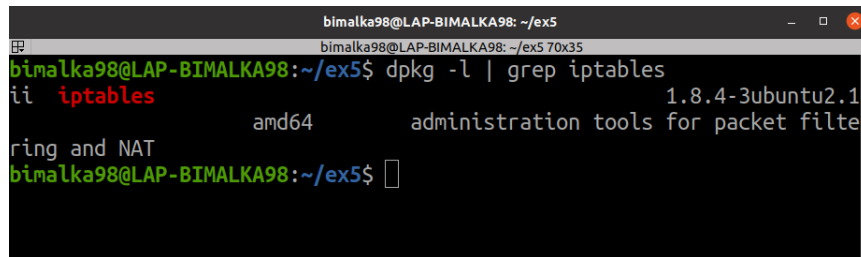
A terminal window titled 'bimalka98@LAP-BIMALKA98: ~/ex5' showing the command 'whoami' being executed. The output is 'bimalka98'.

```
bimalka98@LAP-BIMALKA98: ~/ex5
bimalka98@LAP-BIMALKA98: ~/ex5 70x35
bimalka98@LAP-BIMALKA98:~/ex5$ whoami
bimalka98
bimalka98@LAP-BIMALKA98:~/ex5$
```

Figure 1: Currently logged-in user

Creating Firewall Rules with iptables

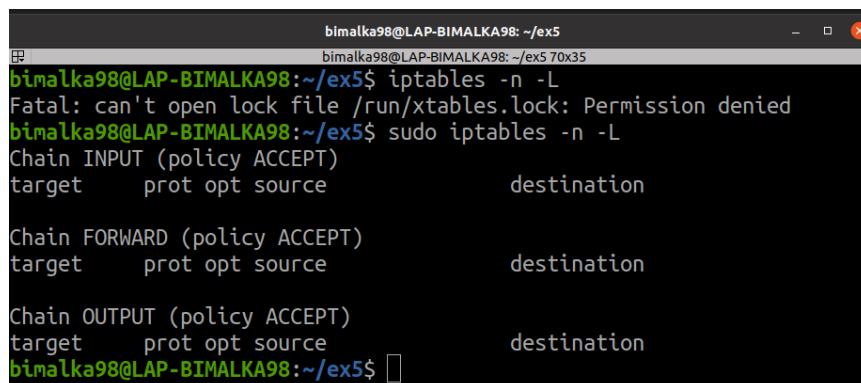
2. Use `dpkg -l | grep iptables` command to check whether iptables is installed on your system. If it does not exist in your system, install it by running `sudo apt-get install iptables`.

A terminal window titled 'bimalka98@LAP-BIMALKA98: ~/ex5' showing the command 'dpkg -l | grep iptables'. The output shows 'ii iptables 1.8.4-3ubuntu2.1 amd64 administration tools for packet filtering and NAT'.

```
bimalka98@LAP-BIMALKA98: ~/ex5
bimalka98@LAP-BIMALKA98: ~/ex5 70x35
bimalka98@LAP-BIMALKA98:~/ex5$ dpkg -l | grep iptables
ii  iptables 1.8.4-3ubuntu2.1
    amd64  administration tools for packet filtering and NAT
bimalka98@LAP-BIMALKA98:~/ex5$
```

Figure 2: Checking whether iptables is installed on the system

3. Check all available iptables rules in your system using the command `/sbin/iptables -n -L`.

A terminal window titled 'bimalka98@LAP-BIMALKA98: ~/ex5' showing the command 'iptables -n -L' which fails with a permission error. Then 'sudo iptables -n -L' is run, showing the default iptables rules for INPUT, FORWARD, and OUTPUT chains.

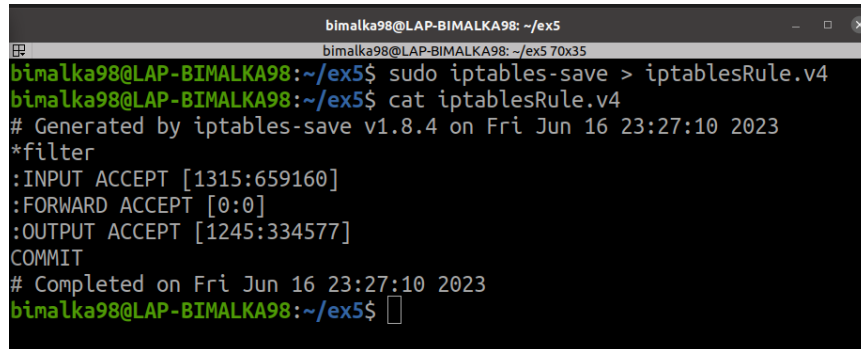
```
bimalka98@LAP-BIMALKA98: ~/ex5
bimalka98@LAP-BIMALKA98: ~/ex5 70x35
bimalka98@LAP-BIMALKA98:~/ex5$ iptables -n -L
Fatal: can't open lock file /run/xtables.lock: Permission denied
bimalka98@LAP-BIMALKA98:~/ex5$ sudo iptables -n -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
bimalka98@LAP-BIMALKA98:~/ex5$
```

Figure 3: Checking all available iptables rules

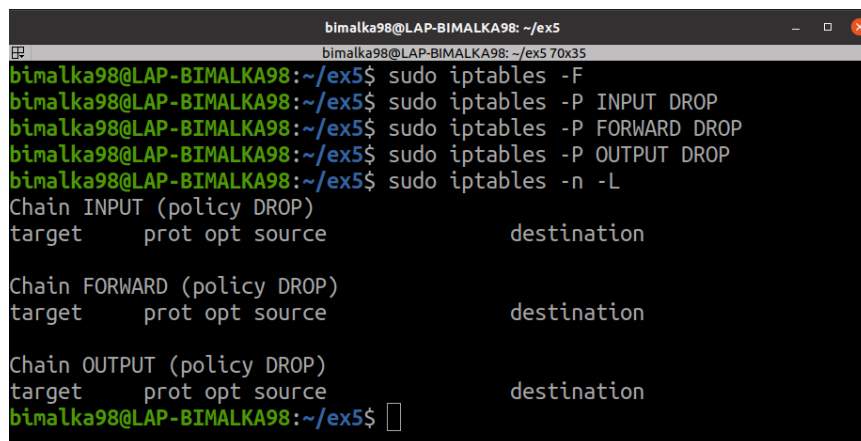
4. Save all available iptables rules to a file named **iptablesRule.v4** using **iptables-save** command.



```
bimalka98@LAP-BIMALK98: ~/ex5
bimalka98@LAP-BIMALK98: ~/ex5 70x35
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables-save > iptablesRule.v4
bimalka98@LAP-BIMALK98:~/ex5$ cat iptablesRule.v4
# Generated by iptables-save v1.8.4 on Fri Jun 16 23:27:10 2023
*filter
:INPUT ACCEPT [1315:659160]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [1245:334577]
COMMIT
# Completed on Fri Jun 16 23:27:10 2023
bimalka98@LAP-BIMALK98:~/ex5$
```

Figure 4: Saving all available iptables rules

5. Flush all the iptables rules that exist in your system and set a default policy to drop packets.



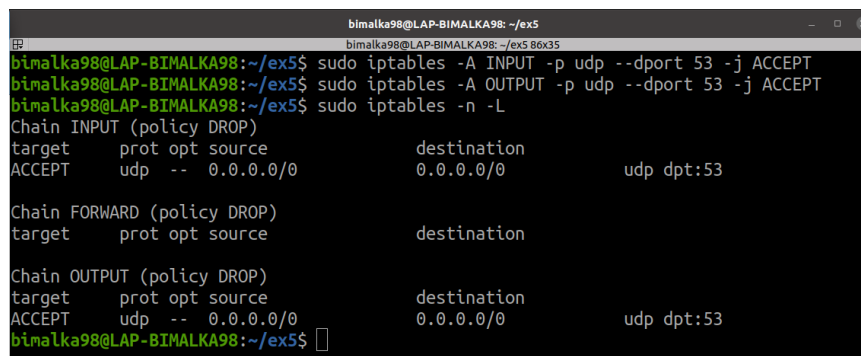
```
bimalka98@LAP-BIMALK98: ~/ex5
bimalka98@LAP-BIMALK98: ~/ex5 70x35
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -F
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -P INPUT DROP
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -P FORWARD DROP
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -P OUTPUT DROP
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -n -L
Chain INPUT (policy DROP)
target      prot opt source                destination

Chain FORWARD (policy DROP)
target      prot opt source                destination

Chain OUTPUT (policy DROP)
target      prot opt source                destination
bimalka98@LAP-BIMALK98:~/ex5$
```

Figure 5: Flushing all the rules and setting a default policy to drop packets

6. Set iptables rules to permit input and output DNS traffic in your system.



```
bimalka98@LAP-BIMALK98: ~/ex5
bimalka98@LAP-BIMALK98: ~/ex5 86x35
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -A OUTPUT -p udp --dport 53 -j ACCEPT
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -n -L
Chain INPUT (policy DROP)
target      prot opt source                destination
ACCEPT      udp  --  0.0.0.0/0              0.0.0.0/0              udp dpt:53

Chain FORWARD (policy DROP)
target      prot opt source                destination

Chain OUTPUT (policy DROP)
target      prot opt source                destination
ACCEPT      udp  --  0.0.0.0/0              0.0.0.0/0              udp dpt:53
bimalka98@LAP-BIMALK98:~/ex5$
```

Figure 6: Permitting input and output DNS traffic

7. Add iptables rules to accept local network incoming and outgoing traffic from the network 192.168.1.0/24.

```
bimalka98@LAP-BIMALKA98: ~/ex5
bimalka98@LAP-BIMALKA98: ~/ex5$ ifconfig
enp4s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 2c:fd:a1:2c:4e:fb txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enx0c5b8f279a64: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 0c:5b:8f:27:9a:64 txqueuelen 1000 (Ethernet)
    RX packets 1888 bytes 625794 (625.7 KB)
    RX errors 1321 dropped 0 overruns 0 frame 1321
    TX packets 0 bytes 0 (0.0 B)
    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 5113 bytes 544815 (544.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 5113 bytes 544815 (544.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.8.102 netmask 255.255.255.0 broadcast 192.168.8.255
    inet6 fe80::f2c8:9c87:6f2a:2d71 prefixlen 64 scopeid 0x20<link>
    ether 00:e1:8c:41:ce:1d txqueuelen 1000 (Ethernet)
    RX packets 221665 bytes 251795655 (251.7 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 120792 bytes 26998692 (26.9 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

bimalka98@LAP-BIMALKA98: ~/ex5$
```

Figure 7: Checking the CIDR of the current local network

```
bimalka98@LAP-BIMALKA98: ~/ex5
bimalka98@LAP-BIMALKA98: ~/ex5$ sudo iptables -A INPUT -s 192.168.8.0/24 -j ACCEPT
bimalka98@LAP-BIMALKA98: ~/ex5$ sudo iptables -A OUTPUT -d 192.168.8.0/24 -j ACCEPT
bimalka98@LAP-BIMALKA98: ~/ex5$ sudo iptables -n -L

Chain INPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0
ACCEPT     all  --  192.168.8.0/24        0.0.0.0/0

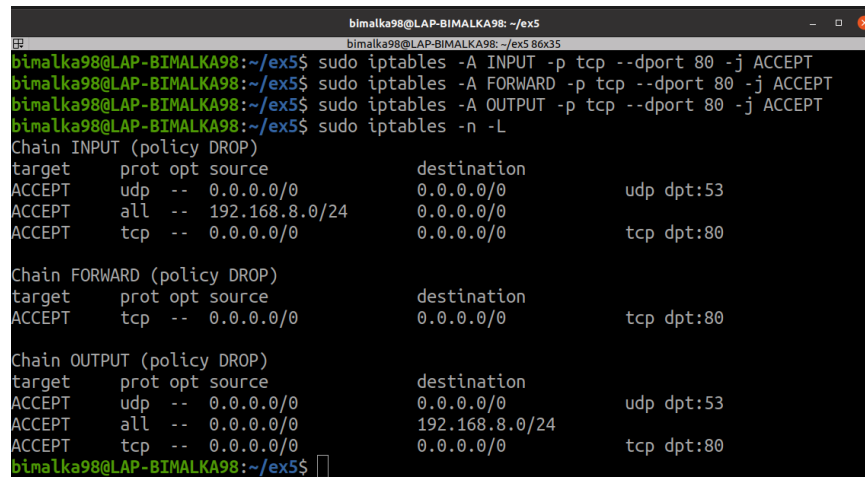
Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0              192.168.8.0/24

bimalka98@LAP-BIMALKA98: ~/ex5$
```

Figure 8: Adding rules to accept local network incoming and outgoing traffic

8. Configure iptables rules to allow all HTTP traffic.



```
bimalka98@LAP-BIMALKA98: ~/ex5
bimalka98@LAP-BIMALKA98:~/ex5$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
bimalka98@LAP-BIMALKA98:~/ex5$ sudo iptables -A FORWARD -p tcp --dport 80 -j ACCEPT
bimalka98@LAP-BIMALKA98:~/ex5$ sudo iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
bimalka98@LAP-BIMALKA98:~/ex5$ sudo iptables -n -L

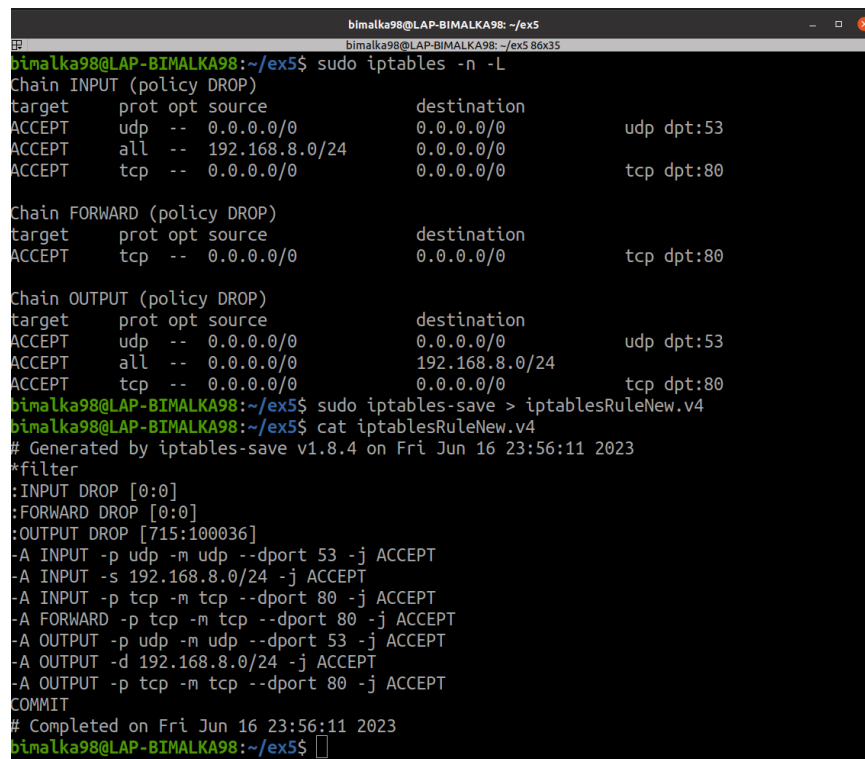
Chain INPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0
ACCEPT     all  --  192.168.8.0/24         0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:80

Chain FORWARD (policy DROP)
target     prot opt source                destination            tcp dpt:80
ACCEPT     tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:80

Chain OUTPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0              192.168.8.0/24
ACCEPT     tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:80
bimalka98@LAP-BIMALKA98:~/ex5$
```

Figure 9: Configuring rules to allow all HTTP traffic

9. View all iptables rules in your system and save them to a file **iptablesRuleNew.v4**.



```
bimalka98@LAP-BIMALKA98: ~/ex5
bimalka98@LAP-BIMALKA98:~/ex5$ sudo iptables -n -L

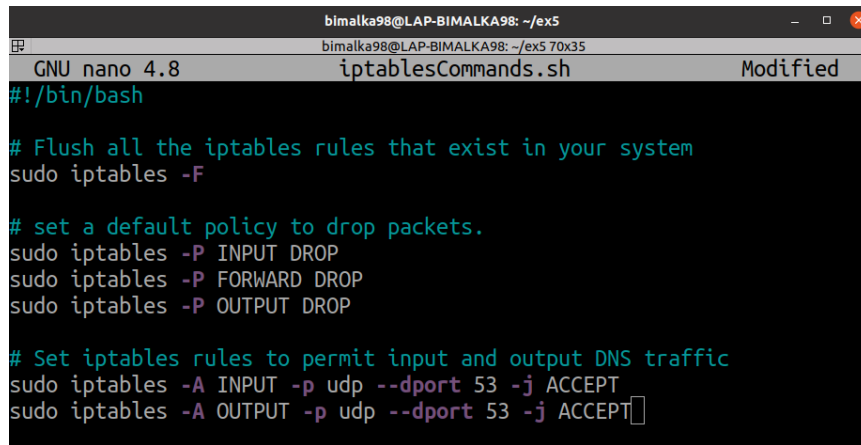
Chain INPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0
ACCEPT     all  --  192.168.8.0/24         0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:80

Chain FORWARD (policy DROP)
target     prot opt source                destination            tcp dpt:80
ACCEPT     tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:80

Chain OUTPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0              192.168.8.0/24
ACCEPT     tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:80
bimalka98@LAP-BIMALKA98:~/ex5$ sudo iptables-save > iptablesRuleNew.v4
bimalka98@LAP-BIMALKA98:~/ex5$ cat iptablesRuleNew.v4
# Generated by iptables-save v1.8.4 on Fri Jun 16 23:56:11 2023
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [715:100036]
-A INPUT -p udp -m udp --dport 53 -j ACCEPT
-A INPUT -s 192.168.8.0/24 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A FORWARD -p tcp -m tcp --dport 80 -j ACCEPT
-A OUTPUT -p udp -m udp --dport 53 -j ACCEPT
-A OUTPUT -d 192.168.8.0/24 -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 80 -j ACCEPT
COMMIT
# Completed on Fri Jun 16 23:56:11 2023
bimalka98@LAP-BIMALKA98:~/ex5$
```

Figure 10: Viewing all rules and saving them to a file

10. Create a file called **iptablesCommands.sh** and put all commands you ran from steps 4, 5 and 6 in the file. After creating the file, flush your iptables commands again and run **iptablesCommands.sh** file. View the iptables rules now and compare with the previous result.



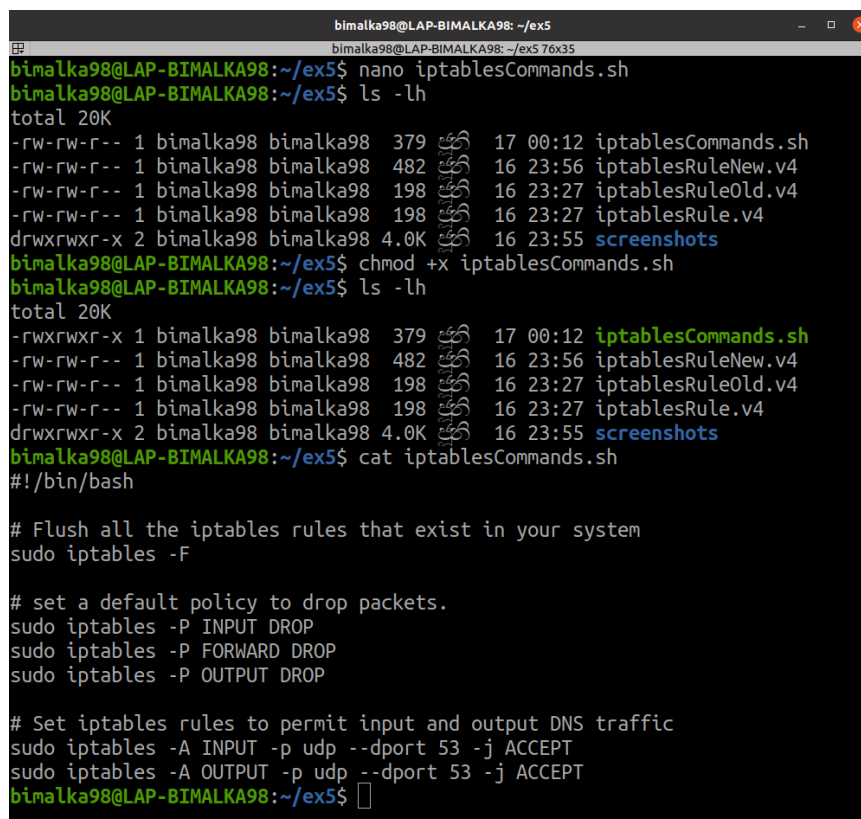
```
bimalka98@LAP-BIMALK98: ~/ex5
bimalka98@LAP-BIMALK98: ~/ex5 70x35
GNU nano 4.8      iptablesCommands.sh      Modified
#!/bin/bash

# Flush all the iptables rules that exist in your system
sudo iptables -F

# set a default policy to drop packets.
sudo iptables -P INPUT DROP
sudo iptables -P FORWARD DROP
sudo iptables -P OUTPUT DROP

# Set iptables rules to permit input and output DNS traffic
sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT
sudo iptables -A OUTPUT -p udp --dport 53 -j ACCEPT
```

Figure 11: Creating the shell script



```
bimalka98@LAP-BIMALK98: ~/ex5
bimalka98@LAP-BIMALK98: ~/ex5 76x35
bimalka98@LAP-BIMALK98:~/ex5$ nano iptablesCommands.sh
bimalka98@LAP-BIMALK98:~/ex5$ ls -lh
total 20K
-rw-rw-r-- 1 bimalka98 bimalka98 379 17 00:12 iptablesCommands.sh
-rw-rw-r-- 1 bimalka98 bimalka98 482 16 23:56 iptablesRuleNew.v4
-rw-rw-r-- 1 bimalka98 bimalka98 198 16 23:27 iptablesRuleOld.v4
-rw-rw-r-- 1 bimalka98 bimalka98 198 16 23:27 iptablesRule.v4
drwxrwxr-x 2 bimalka98 bimalka98 4.0K 16 23:55 screenshots
bimalka98@LAP-BIMALK98:~/ex5$ chmod +x iptablesCommands.sh
bimalka98@LAP-BIMALK98:~/ex5$ ls -lh
total 20K
-rwxrwxr-x 1 bimalka98 bimalka98 379 17 00:12 iptablesCommands.sh
-rw-rw-r-- 1 bimalka98 bimalka98 482 16 23:56 iptablesRuleNew.v4
-rw-rw-r-- 1 bimalka98 bimalka98 198 16 23:27 iptablesRuleOld.v4
-rw-rw-r-- 1 bimalka98 bimalka98 198 16 23:27 iptablesRule.v4
drwxrwxr-x 2 bimalka98 bimalka98 4.0K 16 23:55 screenshots
bimalka98@LAP-BIMALK98:~/ex5$ cat iptablesCommands.sh
#!/bin/bash

# Flush all the iptables rules that exist in your system
sudo iptables -F

# set a default policy to drop packets.
sudo iptables -P INPUT DROP
sudo iptables -P FORWARD DROP
sudo iptables -P OUTPUT DROP

# Set iptables rules to permit input and output DNS traffic
sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT
sudo iptables -A OUTPUT -p udp --dport 53 -j ACCEPT
bimalka98@LAP-BIMALK98:~/ex5$
```

Figure 12: Making the script executable

```
bimalka98@LAP-BIMALK98: ~/ex5
bimalka98@LAP-BIMALK98: ~/ex5$ sudo iptables -n -L
[sudo] password for bimalka98:
Chain INPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0
ACCEPT     all  --  192.168.8.0/24         0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0              0.0.0.0/0            tcp dpt:80

Chain FORWARD (policy DROP)
target     prot opt source                destination            tcp dpt:80
ACCEPT     tcp  --  0.0.0.0/0              0.0.0.0/0

Chain OUTPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     all  --  0.0.0.0/0              192.168.8.0/24
ACCEPT     tcp  --  0.0.0.0/0              0.0.0.0/0            tcp dpt:80

bimalka98@LAP-BIMALK98:~/ex5$ ./iptablesCommands.sh
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -n -L
Chain INPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0

Chain FORWARD (policy DROP)
target     prot opt source                destination

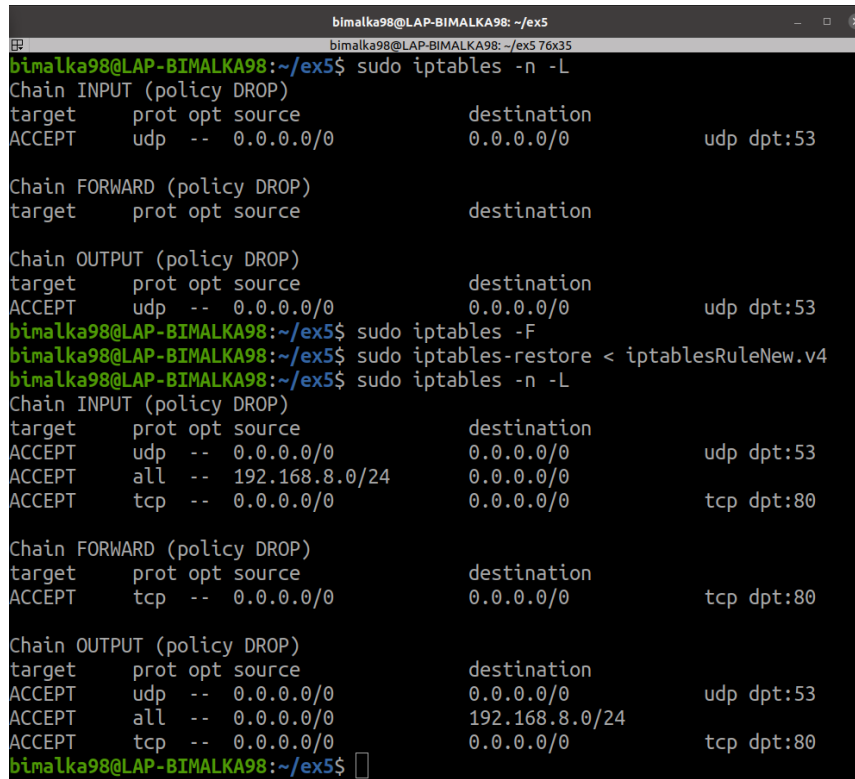
Chain OUTPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0

bimalka98@LAP-BIMALK98:~/ex5$
```

Figure 13: Executing the shell script

As it can be observed in the Figure 13, it is possible to set `iptables` rules through shell scripts, which is an efficient way when it comes to setting rules in multiple host devices. In terms of the differences there is no difference between setting the rules manually and setting the rules through the script. The reason for the change in the number of rules is just because some rules were not included in the script (only the rules from item 4, 5 and 6 are there).

11. Finally, flush your iptables rules again. But this time, load the saved iptables rules from the file **iptablesRuleNew.v4** using **iptables-restore** command. View the iptables rules and compare them with the ones you have in step 8.



```
bimalka98@LAP-BIMALK98: ~/ex5
bimalka98@LAP-BIMALK98: ~/ex5 76x35
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -n -L
Chain INPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0

Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -F
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables-restore < iptablesRuleNew.v4
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -n -L
Chain INPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0
ACCEPT     all  --  192.168.8.0/24         0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:80

Chain FORWARD (policy DROP)
target     prot opt source                destination            tcp dpt:80
ACCEPT     tcp  --  0.0.0.0/0              0.0.0.0/0

Chain OUTPUT (policy DROP)
target     prot opt source                destination            udp dpt:53
ACCEPT     udp  --  0.0.0.0/0              0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0              192.168.8.0/24
ACCEPT     tcp  --  0.0.0.0/0              0.0.0.0/0              tcp dpt:80
bimalka98@LAP-BIMALK98:~/ex5$
```

Figure 14: Flushing current rules and restoring saved rules

When comparing the output shown in the Figure 14 and the Figure 9, it can be observed that there is no difference. This means it is possible to save iptables of a host at a given moment and restore them at another moment as required. This can be a useful tool when it comes to backing up rules before any significant change to the rules, because if anything goes wrong we have a fallback position.

Creating Firewall Rules with UFW

The scenario comprises of two virtual machines (VM1 IP - 192.168.46.140 and VM2 IP - 192.168.46.141) running on a host (HOST IP - 192.168.46.1) machine. VM1 is an Ubuntu virtual machine that has a firewall implemented/configured.

The current firewall ruleset is as below.

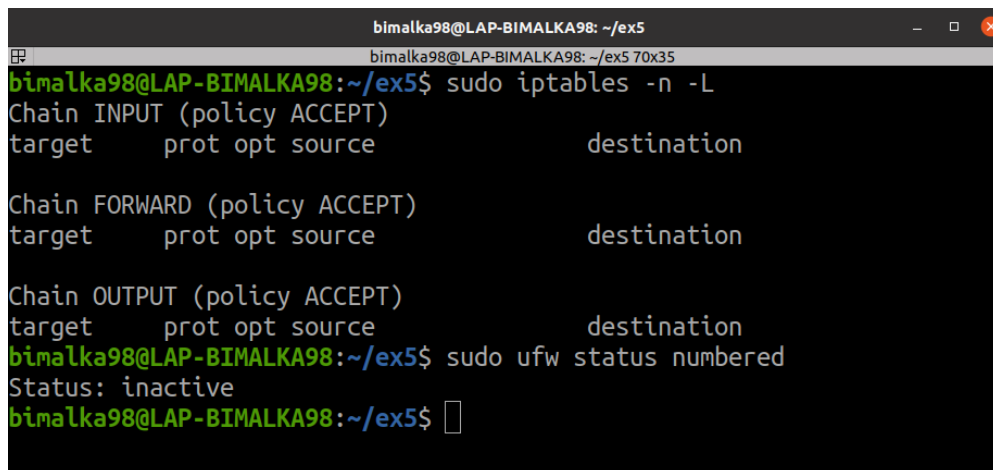
```
Chain INPUT (policy DROP)
target     prot opt source                destination

Chain FORWARD (policy DROP)
target     prot opt source                destination

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

All chain policies are set to drop traffic. To implement base rules, you can use the following commands:

- Delete any current rules associated with UFW using `sudo ufw reset`
- Disable UFW using `sudo ufw disable`
- Flush all iptables rules using `sudo iptables -F`
- Enable UFW using `sudo ufw enable`
- Deny outgoing traffic using `sudo ufw default deny outgoing`

A terminal window titled 'bimalka98@LAP-BIMALKA98: ~/ex5' showing the output of 'sudo iptables -n -L' and 'sudo ufw status numbered'. The iptables output shows three chains (INPUT, FORWARD, OUTPUT) all with a policy of ACCEPT. The UFW status is inactive.

```
bimalka98@LAP-BIMALKA98: ~/ex5
bimalka98@LAP-BIMALKA98: ~/ex5 70x35
bimalka98@LAP-BIMALKA98:~/ex5$ sudo iptables -n -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
bimalka98@LAP-BIMALKA98:~/ex5$ sudo ufw status numbered
Status: inactive
bimalka98@LAP-BIMALKA98:~/ex5$
```

Figure 15: Current firewall rules

```
bimalka98@LAP-BIMALK98: ~/ex5
bimalka98@LAP-BIMALK98: ~/ex5 70x35
bimalka98@LAP-BIMALK98:~/ex5$ sudo ufw reset
Resetting all rules to installed defaults. Proceed with operation (y|n)? y
Backing up 'user.rules' to '/etc/ufw/user.rules.20230617_233502'
Backing up 'before.rules' to '/etc/ufw/before.rules.20230617_233502'
Backing up 'after.rules' to '/etc/ufw/after.rules.20230617_233502'
Backing up 'user6.rules' to '/etc/ufw/user6.rules.20230617_233502'
Backing up 'before6.rules' to '/etc/ufw/before6.rules.20230617_233502'
Backing up 'after6.rules' to '/etc/ufw/after6.rules.20230617_233502'

bimalka98@LAP-BIMALK98:~/ex5$ sudo ufw disable
Firewall stopped and disabled on system startup
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -F
bimalka98@LAP-BIMALK98:~/ex5$ sudo ufw enable
Firewall is active and enabled on system startup
bimalka98@LAP-BIMALK98:~/ex5$ sudo ufw default deny outgoing
Default outgoing policy changed to 'deny'
(be sure to update your rules accordingly)
bimalka98@LAP-BIMALK98:~/ex5$
```

Figure 16: Implementing the base rules

```
bimalka98@LAP-BIMALK98: ~/ex5
bimalka98@LAP-BIMALK98: ~/ex5 95x35
bimalka98@LAP-BIMALK98:~/ex5$ sudo iptables -n -L
Chain INPUT (policy DROP)
target     prot opt source                destination
ufw-before-logging-input  all  --  0.0.0.0/0            0.0.0.0/0
ufw-before-input          all  --  0.0.0.0/0            0.0.0.0/0
ufw-after-input           all  --  0.0.0.0/0            0.0.0.0/0
ufw-after-logging-input   all  --  0.0.0.0/0            0.0.0.0/0
ufw-reject-input          all  --  0.0.0.0/0            0.0.0.0/0
ufw-track-input           all  --  0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy DROP)
target     prot opt source                destination
ufw-before-logging-forward all  --  0.0.0.0/0            0.0.0.0/0
ufw-before-forward        all  --  0.0.0.0/0            0.0.0.0/0
ufw-after-forward         all  --  0.0.0.0/0            0.0.0.0/0
ufw-after-logging-forward all  --  0.0.0.0/0            0.0.0.0/0
ufw-reject-forward        all  --  0.0.0.0/0            0.0.0.0/0
ufw-track-forward         all  --  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy DROP)
target     prot opt source                destination
ufw-before-logging-output all  --  0.0.0.0/0            0.0.0.0/0
ufw-before-output         all  --  0.0.0.0/0            0.0.0.0/0
ufw-after-output          all  --  0.0.0.0/0            0.0.0.0/0
ufw-after-logging-output  all  --  0.0.0.0/0            0.0.0.0/0
ufw-reject-output         all  --  0.0.0.0/0            0.0.0.0/0
ufw-track-output          all  --  0.0.0.0/0            0.0.0.0/0
```

Figure 17: iptables after setting up the base rules

12. Implement the following network administration in VM1:

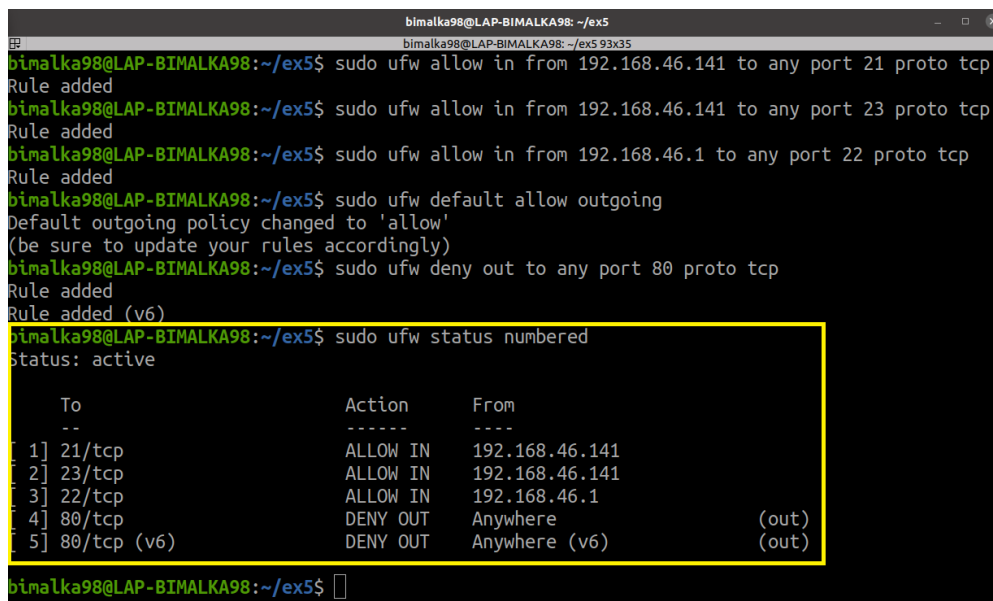
- Access to VM1 from VM2 must only be allowed over FTP and Telnet.
- Access to VM1 from HOST must only be allowed over SSH
- Allow all outgoing traffic from VM1 with the exception of access to HTTP websites

In this task, you are asked to implement UFW rules on the ubuntu machine. You can pretend that VM2 and HOST exist in your network. List the commands you used to achieve the above. Add a screenshot of the terminal output after running the command `sudo ufw status numbered`.

If the firewall is physically implemented, you could have tested the connections using PuTTY or the command line.

Following network administration in VM1 is configured using the associated commands given below.

- *Access to VM1 from VM2 must only be allowed over FTP and Telnet*
`sudo ufw allow in from 192.168.46.141 to any port 21 proto tcp`
`sudo ufw allow in from 192.168.46.141 to any port 23 proto tcp`
- *Access to VM1 from HOST must only be allowed over SSH*
`sudo ufw allow in from 192.168.46.1 to any port 22 proto tcp`
- *Allow all outgoing traffic from VM1 with the exception of access to HTTP websites*
`sudo ufw default allow outgoing`
`sudo ufw deny out to any port 80 proto tcp`

A terminal window titled 'bimalka98@LAP-BIMALKA98: ~/ex5' showing the execution of UFW commands. The commands are: 'sudo ufw allow in from 192.168.46.141 to any port 21 proto tcp', 'sudo ufw allow in from 192.168.46.141 to any port 23 proto tcp', 'sudo ufw allow in from 192.168.46.1 to any port 22 proto tcp', 'sudo ufw default allow outgoing', and 'sudo ufw deny out to any port 80 proto tcp'. The output shows 'Rule added' for each rule and 'Default outgoing policy changed to 'allow'' for the default rule. The final command 'sudo ufw status numbered' is highlighted with a yellow box, and its output is shown below it.

```
bimalka98@LAP-BIMALKA98: ~/ex5$ sudo ufw allow in from 192.168.46.141 to any port 21 proto tcp
Rule added
bimalka98@LAP-BIMALKA98: ~/ex5$ sudo ufw allow in from 192.168.46.141 to any port 23 proto tcp
Rule added
bimalka98@LAP-BIMALKA98: ~/ex5$ sudo ufw allow in from 192.168.46.1 to any port 22 proto tcp
Rule added
bimalka98@LAP-BIMALKA98: ~/ex5$ sudo ufw default allow outgoing
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
bimalka98@LAP-BIMALKA98: ~/ex5$ sudo ufw deny out to any port 80 proto tcp
Rule added
Rule added (v6)
bimalka98@LAP-BIMALKA98: ~/ex5$ sudo ufw status numbered
Status: active

    To      Action      From
    --      -
1] 21/tcp    ALLOW IN    192.168.46.141
2] 23/tcp    ALLOW IN    192.168.46.141
3] 22/tcp    ALLOW IN    192.168.46.1
4] 80/tcp    DENY OUT    Anywhere          (out)
5] 80/tcp (v6) DENY OUT    Anywhere (v6)      (out)

bimalka98@LAP-BIMALKA98: ~/ex5$
```

Figure 18: Output after running the command `sudo ufw status numbered`

```

bimalka98@LAP-BIMALKA98: ~/ex5
bimalka98@LAP-BIMALKA98: ~/ex5$ sudo iptables -n -L
Chain INPUT (policy DROP)
target     prot opt source                destination
ufw-before-logging-input all -- 0.0.0.0/0             0.0.0.0/0
ufw-before-input all -- 0.0.0.0/0             0.0.0.0/0
ufw-after-input all -- 0.0.0.0/0             0.0.0.0/0
ufw-after-logging-input all -- 0.0.0.0/0             0.0.0.0/0
ufw-reject-input all -- 0.0.0.0/0             0.0.0.0/0
ufw-track-input all -- 0.0.0.0/0             0.0.0.0/0

Chain FORWARD (policy DROP)
target     prot opt source                destination
ufw-before-logging-forward all -- 0.0.0.0/0             0.0.0.0/0
ufw-before-forward all -- 0.0.0.0/0             0.0.0.0/0
ufw-after-forward all -- 0.0.0.0/0             0.0.0.0/0
ufw-after-logging-forward all -- 0.0.0.0/0             0.0.0.0/0
ufw-reject-forward all -- 0.0.0.0/0             0.0.0.0/0
ufw-track-forward all -- 0.0.0.0/0             0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
ufw-before-logging-output all -- 0.0.0.0/0             0.0.0.0/0
ufw-before-output all -- 0.0.0.0/0             0.0.0.0/0
ufw-after-output all -- 0.0.0.0/0             0.0.0.0/0
ufw-after-logging-output all -- 0.0.0.0/0             0.0.0.0/0
ufw-reject-output all -- 0.0.0.0/0             0.0.0.0/0
ufw-track-output all -- 0.0.0.0/0             0.0.0.0/0

```

Figure 19: iptables ruleset after mentioned configurations

```

Chain ufw-user-input (1 references)
target     prot opt source                destination
ACCEPT     tcp -- 192.168.46.141          0.0.0.0/0             tcp dpt:21
ACCEPT     tcp -- 192.168.46.141          0.0.0.0/0             tcp dpt:23
ACCEPT     tcp -- 192.168.46.1           0.0.0.0/0             tcp dpt:22

Chain ufw-user-limit (0 references)
target     prot opt source                destination
LOG         all -- 0.0.0.0/0             0.0.0.0/0             limit: avg 3/min burst 5 LOG
             flags 0 level 4 prefix "[UFW LIMIT BLOCK]"
REJECT     all -- 0.0.0.0/0             0.0.0.0/0             reject-with icmp-port-unreachable

Chain ufw-user-limit-accept (0 references)
target     prot opt source                destination
ACCEPT     all -- 0.0.0.0/0             0.0.0.0/0

Chain ufw-user-logging-forward (0 references)
target     prot opt source                destination

Chain ufw-user-logging-input (0 references)
target     prot opt source                destination

Chain ufw-user-logging-output (0 references)
target     prot opt source                destination

Chain ufw-user-output (1 references)
target     prot opt source                destination
DROP      tcp -- 0.0.0.0/0             0.0.0.0/0             tcp dpt:80
bimalka98@LAP-BIMALKA98: ~/ex5$

```

Figure 20: Mentioned configurations in the iptables

Scan systems with NMAP

In this section, you will scan for the Ports of a remote host. You will need to have two devices connected to the same local network to perform this task.

13. View ip addresses of both devices using `hostname -I` command.



Figure 21: IP addresses of the two devices (Laptop & a Raspberry Pi)

14. Scan one host from the other host for TCP and UDP ports using `nmap` command.

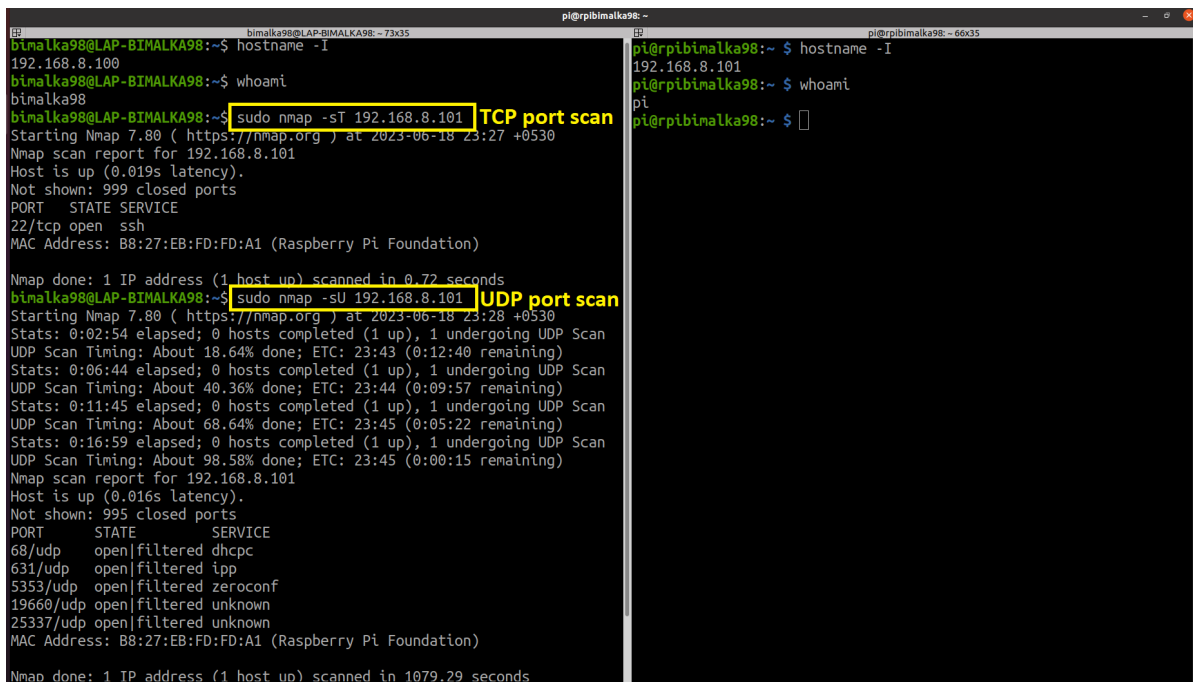


Figure 22: Scan Raspberry Pi device from the Laptop for TCP and UDP ports

Section 2

15. Briefly explain VLANs, VPNs, DMZs and Network Segmentation concepts outlining their similarities and differences.

VLAN (a virtual local area network) allows multiple logical networks to exist on the same physical network. This improves the security and the performance of a given network by limiting the access to the resources within the given VLAN and isolating the traffic of a given VLAN from other VLANs respectively. VLANs assigned for each department of an organization is an example for this.

VPN (a virtual private network) allows an encrypted tunnel to be created between two end points in an unsecured network (eg: internet) for secure communication. This minimize the possibility of being tracked by a malicious third party. VPNs are used to connect remote employees or branches of an organization, to its corporate network for secure communication.

DMZ (a demilitarized zone) or otherwise known as a perimeter network/ screened subnet is a physical or logical network segment (a buffer) that separates an organization's internal LAN from the other unsecured networks (eg: internet). This allows an additional layer of security to be maintained, as it restrict the direct access from the internet to the LAN. The services/ servers provided to public are placed in the DMZ, while organization's private resources are kept in the internal network.

Network Segmentation is a method to isolate/ control the traffic of a given network. As in the VLANs, improved performance due to reduced traffic congestion and improved security due to virtual isolation of networks are the benefits.

16. Perform a comparison between IPsec and SSL.

Table 1: Comparison of IPsec and SSL

IPsec	SSL
Your answer here	Your answer here
Your answer here	Your answer here

17. Explain the differences between an IDS, an IPS, and a firewall?

Your answer here

18. What is the difference between anomaly detection and signature or heuristic-based intrusion detection?

Your answer here