

Lab 7: Enforcing the Security Rules with 'iptables'

Objective: This lab aims to provide students with a deeper understanding of network security using `iptables` on Linux-based systems. The goal is to configure `iptables` rules on the CentOS server to secure it against malicious activities while allowing legitimate traffic. This lab will enable students to simulate various attack scenarios, including port scanning, denial-of-service (DoS), and unauthorized access attempts.

Network Setup: We are working in a virtualized network with the following machines and IP addresses:

- Kali Linux VM (IP: 192.168.1.6) – Attacker machine
- Metasploitable VM (IP: 192.168.1.8) – Vulnerable target machine
- CentOS VM (IP: 192.168.1.10) – Web Server and Firewall

The IP addresses could be different in your case but all VMs should be connected to the same internal network (192.168.1.0/24). Place multiple screenshots here depicting the IP assignment of each machine by issuing an `ifconfig` command.

```

0 access of their shared documentation, please visit:
http://help.ubuntu.com/
to mail.
sfadmin@metasploitable:~$ ifconfig
eth0: Link encap:Ethernet HWaddr 08:00:27:61:0c:49
      inet addr:192.168.1.4 Bcast:192.168.1.255 Mask:255.255.255.0
      inet6 addr: fe80::a00:27ff:fe61:c49/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:44 errors:0 dropped:0 overruns:0 frame:0
      TX packets:66 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:5884 (5.7 KB) TX bytes:7126 (6.9 KB)
      Base address:0xd020 Memory:f0200000-f0220000

lo: Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:96 errors:0 dropped:0 overruns:0 frame:0
      TX packets:96 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:21437 (20.9 KB) TX bytes:21437 (20.9 KB)

osboxes@osboxes ~]$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.6 netmask 255.255.255.0 broadcast 0.0.0.0
      ether 08:00:27:61:c2:ce txqueuelen 1000 (Ethernet)
      RX packets 60 bytes 9715 (9.4 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 8 bytes 882 (882.0 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 402 bytes 31850 (31.1 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 402 bytes 31850 (31.1 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.5 netmask 255.255.255.0 broadcast 192.168.1.255
      inet6 fe80::533c:ab58:b6ab:11ff prefixlen 64 scopeid 0x20<link>
      ether 08:00:27:5c:63:16 txqueuelen 1000 (Ethernet)
      RX packets 32 bytes 5752 (5.6 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 26 bytes 3276 (3.1 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 8 bytes 480 (480.0 B)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 8 bytes 480 (480.0 B)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

The task is to configure iptables on the CentOS machine to secure the network and protect the system from attack, while still allowing legitimate traffic between the VMs.

Pre-requisites: Ensure the following services are running on each VM:

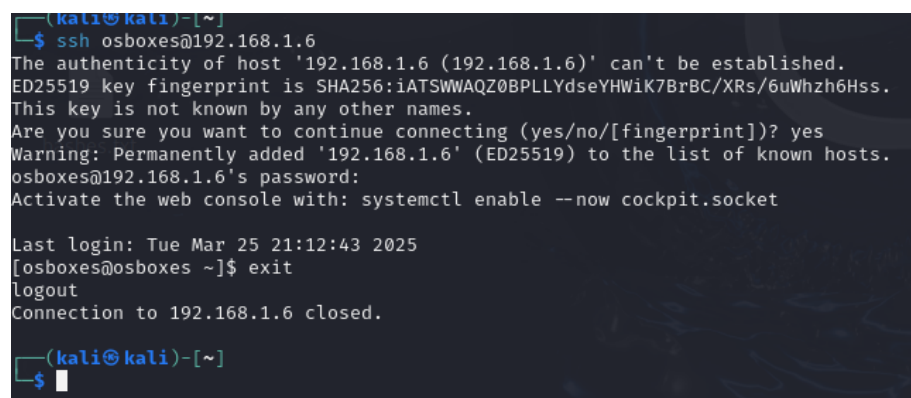
- Metasploitable: SSH (Port 22), HTTP (Port 80), and other vulnerable services.
- CentOS: Apache web server (Port 80), SSH (Port 22).

Hint: Use **Nmap** tool to scan for open ports. If ports are not open, then investigate a little to find the useful commands that can be used to open the ports on each target machine (e.g., CentOS).

What could be the possible commands that can be used to open up the ports?

```
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp -s 192.168.1.4 --dport 22 -j ACCEPT
[sudo] password for osboxes:
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp -s 192.168.1.5 --dport 22 -j ACCEPT
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
[osboxes@osboxes ~]$ sudo iptables -A INPUT -j DROP
```

Re-scan the VMs to verify that the desired ports are now open as indicated above.

Place screenshots here for the above task.

```
(kali@kali)-[~]
$ ssh osboxes@192.168.1.6
The authenticity of host '192.168.1.6 (192.168.1.6)' can't be established.
ED25519 key fingerprint is SHA256:iATSWWAQZ0BPLLYdseYHWiK7BrBC/XRs/6uWhzh6Hss.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.6' (ED25519) to the list of known hosts.
osboxes@192.168.1.6's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Tue Mar 25 21:12:43 2025
[osboxes@osboxes ~]$ exit
logout
Connection to 192.168.1.6 closed.

(kali@kali)-[~]
```

We also need to ensure that the ping is working correctly between all VMs to verify network connectivity.

Scenario Overview: You are tasked with securing the CentOS web server using `iptables`. At the same time, you must allow authorized services to function properly, such as SSH for administrative access and HTTP for web traffic. The attacker (Kali Linux) will attempt to exploit vulnerabilities and perform port scanning to launch attacks (e.g., DoS). The defender (CentOS) needs to use `iptables` to defend against these threats.

Lab Tasks:**1. Initial Setup and Network Testing**

Ping Test: Ensure that all VMs can ping each other: (Your IP scheme may differ)

Place screenshots here for the above task.

```

(kali@kali)-[~]
$ ping 192.168.1.6 -c 4
PING 192.168.1.6 (192.168.1.6) 56(84) bytes of data.
64 bytes from 192.168.1.6: icmp_seq=1 ttl=64 time=0.475 ms
^[[A^[[A64 bytes from 192.168.1.6: icmp_seq=2 ttl=64 time=1.15 ms
64 bytes from 192.168.1.6: icmp_seq=3 ttl=64 time=0.377 ms
64 bytes from 192.168.1.6: icmp_seq=4 ttl=64 time=0.478 ms

--- 192.168.1.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3092ms
rtt min/avg/max/mdev = 0.377/0.619/1.146/0.306 ms

(kali@kali)-[~]
$ ping 192.168.1.4 -c 4
PING 192.168.1.4 (192.168.1.4) 56(84) bytes of data.
64 bytes from 192.168.1.4: icmp_seq=1 ttl=64 time=0.352 ms
64 bytes from 192.168.1.4: icmp_seq=2 ttl=64 time=0.997 ms
64 bytes from 192.168.1.4: icmp_seq=3 ttl=64 time=0.876 ms
64 bytes from 192.168.1.4: icmp_seq=4 ttl=64 time=1.04 ms

--- 192.168.1.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3069ms
rtt min/avg/max/mdev = 0.352/0.816/1.042/0.275 ms

[osboxes@osboxes ~]$ sudo iptables -I INPUT 1 -p icmp -j ACCEPT
[sudo] password for osboxes:
[osboxes@osboxes ~]$ ping -c 4 192.168.1.5
PING 192.168.1.5 (192.168.1.5) 56(84) bytes of data.
64 bytes from 192.168.1.5: icmp_seq=1 ttl=64 time=0.416 ms
64 bytes from 192.168.1.5: icmp_seq=2 ttl=64 time=1.05 ms
64 bytes from 192.168.1.5: icmp_seq=3 ttl=64 time=0.862 ms
64 bytes from 192.168.1.5: icmp_seq=4 ttl=64 time=1.14 ms

--- 192.168.1.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3094ms
rtt min/avg/max/mdev = 0.416/0.865/1.135/0.277 ms
[osboxes@osboxes ~]$ ping -c 4 192.168.1.4
PING 192.168.1.4 (192.168.1.4) 56(84) bytes of data.
64 bytes from 192.168.1.4: icmp_seq=1 ttl=64 time=0.368 ms
64 bytes from 192.168.1.4: icmp_seq=2 ttl=64 time=0.788 ms
64 bytes from 192.168.1.4: icmp_seq=3 ttl=64 time=0.924 ms
64 bytes from 192.168.1.4: icmp_seq=4 ttl=64 time=0.513 ms

--- 192.168.1.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3087ms
rtt min/avg/max/mdev = 0.368/0.648/0.924/0.219 ms
[osboxes@osboxes ~]$
msfadmin@metasploitable:~$ ping -c 4 192.168.1.5
PING 192.168.1.5 (192.168.1.5) 56(84) bytes of data.
64 bytes from 192.168.1.5: icmp_seq=1 ttl=64 time=11.2 ms
64 bytes from 192.168.1.5: icmp_seq=2 ttl=64 time=1.37 ms
64 bytes from 192.168.1.5: icmp_seq=3 ttl=64 time=1.32 ms
64 bytes from 192.168.1.5: icmp_seq=4 ttl=64 time=0.575 ms

--- 192.168.1.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 0.575/3.639/11.277/4.421 ms
msfadmin@metasploitable:~$ ping -c 4 192.168.1.6
PING 192.168.1.6 (192.168.1.6) 56(84) bytes of data.
64 bytes from 192.168.1.6: icmp_seq=1 ttl=64 time=0.508 ms
64 bytes from 192.168.1.6: icmp_seq=2 ttl=64 time=0.645 ms
64 bytes from 192.168.1.6: icmp_seq=3 ttl=64 time=0.757 ms
64 bytes from 192.168.1.6: icmp_seq=4 ttl=64 time=0.879 ms

--- 192.168.1.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3035ms
rtt min/avg/max/mdev = 0.508/0.697/0.879/0.138 ms
msfadmin@metasploitable:~$ _

```

2. Basic iptables Setup on CentOS

- Configure iptables to allow incoming traffic on ports 22 (SSH) and 80 (HTTP) from Kali Linux and Metasploitable.

```

sudo iptables -A INPUT -p tcp -s 192.168.1.8 --dport 22 -j ACCEPT
# Allow SSH from Metasploitable
sudo iptables -A INPUT -p tcp -s 192.168.1.6 --dport 22 -j ACCEPT
# Allow SSH from Kali
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
# Allow HTTP from all

```

- Block all incoming traffic by default except for the allowed ports.

```

sudo iptables -A INPUT -j DROP

```

Place screenshots here for the above task.

```
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp -s 192.168.1.4 --dport 22 -j ACCEPT
[sudo] password for osboxes:
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp -s 192.168.1.5 --dport 22 -j ACCEPT
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
[osboxes@osboxes ~]$ sudo iptables -A INPUT -j DROP
```

3. Denial-of-Service (DoS) Prevention

- Configure iptables to rate limit incoming HTTP requests to prevent a DoS attack. Allow a maximum of 10 requests per second with a burst of up to 20.

```
sudo iptables -A INPUT -p tcp --dport 80 -m limit --limit 10/s --limit-burst 20 -j ACCEPT
```

- Add a rule to prevent SYN flood attacks (DoS attacks that overwhelm the system with incomplete TCP connections).

```
sudo iptables -A INPUT -p tcp --syn -m limit --limit 1/s --limit-burst 5 -j ACCEPT
```

```
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp --dport 80 -m limit --limit 10/s --limit-burst 20 -j ACCEPT
[sudo] password for osboxes:
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp --syn -m limit --limit 1/s --limit-burst 5 -j ACCEPT
```

4. Port Scanning Detection and Mitigation

- Configure iptables to log all incoming connection attempts on high-numbered ports (1-1023) to detect possible port scans.

```
sudo iptables -A INPUT -p tcp --dport 1:1023 -j LOG --log-prefix "PORT SCAN: " --log-level 4
sudo iptables -A INPUT -p tcp --dport 1:1023 -j DROP
```

- Now use Nmap scan against CentOS (192.168.1.10) to test the port scanning detection.

Place screenshots here for the above task.

```
nmap -p 1-1023 192.168.1.10
```

- Check the log output to ensure that suspicious traffic is being logged correctly.

```
sudo tail -f /var/log/messages
```

```
(kali@kali)-[~]
$ nmap -p 1-1023 192.168.1.6
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-25 22:30 EDT
Nmap scan report for 192.168.1.6
Host is up (0.00089s latency).
Not shown: 1021 filtered tcp ports (no-response), 1 filtered tcp ports (admin-prohibited)
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 08:00:27:61:C2:CE (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 5.54 seconds

[osboxes@osboxes ~]$ sudo journalctl \ grep "PORT SCAN"
Failed to add match 'grep': Invalid argument
[osboxes@osboxes ~]$ sudo journalctl | grep "PORT SCAN"
Mar 25 22:29:39 osboxes sudo[3548]: osboxes : TTY=pts/0 ; PWD=/home/osboxes ;
USER=root ; COMMAND=/sbin/iptables -A INPUT -p tcp --dport 1:1023 -j LOG --log-
prefix PORT SCAN: --log-level 4
Mar 25 22:39:40 osboxes sudo[3713]: osboxes : TTY=pts/0 ; PWD=/home/osboxes ;
USER=root ; COMMAND=/sbin/iptables -A INPUT -p tcp --dport 1:1023 -j LOG --log-
prefix PORT SCAN: --log-level 4
Mar 25 22:42:07 osboxes sudo[3752]: osboxes : TTY=pts/0 ; PWD=/home/osboxes ;
USER=root ; COMMAND=/bin/grep PORT SCAN /var/log/messages
Mar 25 22:58:14 osboxes sudo[3915]: osboxes : TTY=pts/0 ; PWD=/home/osboxes ;
USER=root ; COMMAND=/bin/journalctl grep PORT SCAN
[osboxes@osboxes ~]$
```

Do you see anything? Are you able to interpret some important information? Explain briefly. - After adding the logging rule, I ran an Nmap scan from Kali targeting CentOS ports 1–1023. I then verified detection using `journalctl`, which showed multiple logs with the "PORT SCAN:" prefix, confirming the scan was detected and logged by iptables.

5. Blocking Unauthorized IP Addresses

- After detecting suspicious activity or an attack attempt from Kali Linux (192.168.1.6), you can block all traffic from this IP to the CentOS server.

```
sudo iptables -A INPUT -s 192.168.1.6 -j DROP
```

- Attempt to ping CentOS and access HTTP services from Kali Linux after the block is applied. The requests should be dropped.

Place screenshots here for the above task.

```
023
14 LOG tcp -- 0.0.0.0/0 0.0.0.0/0
023 LOG flags 0 level 4 prefix "PORT SCAN: "
15 DROP all -- 192.168.1.5 0.0.0.0/0
16 DROP all -- 192.168.1.5 0.0.0.0/0

(kali@kali)-[~]
$ ping 192.168.1.6 -c 4
PING 192.168.1.6 (192.168.1.6) 56(84) bytes of data.

— 192.168.1.6 ping statistics —
4 packets transmitted, 0 received, 100% packet loss, time 3073ms
```

6. Advanced Rule Configuration

- Allow only Metasploitable (192.168.1.8) to access the web server (HTTP on port 80).

```
sudo iptables -A INPUT -p tcp -s 192.168.1.8 --dport 80 -j ACCEPT
```

```
sudo iptables -A INPUT -p tcp --dport 80 -j DROP
# Block all other traffic to port 80
```

- Allow Metasploitable (192.168.1.8) and Kali Linux (192.168.1.6) to access SSH, but block all other sources.

```
sudo iptables -A INPUT -p tcp -s 192.168.1.8 --dport 22 -j ACCEPT
```

```
# Allow Metasploitable
```

```
sudo iptables -A INPUT -p tcp -s 192.168.1.6 --dport 22 -j ACCEPT
```

```
# Allow Kali Linux
```

```
sudo iptables -A INPUT -p tcp --dport 22 -j DROP
# Block others
```


Place screenshots here for the above task.

```
osboxes@osboxes:~
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp -s 192.168.1.4 --dport 80 -j ACCEPT
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp --dport 80 -j DROP
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp -s 192.168.1.4 --dport 22 -j ACCEPT
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp -s 192.168.1.5 --dport 22 -j ACCEPT
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp --dport 22 -j DROP
bash: sudo: command not found...
[osboxes@osboxes ~]$ sudo iptables -A INPUT -p tcp --dport 22 -j DROP
[osboxes@osboxes ~]$
```

7. Testing and Validation

- Use Kali Linux to perform a port scan against CentOS. The iptables rules should prevent unauthorized access.

nmap 192.168.1.10

Verify that only the allowed ports (22, 80) are open.

Place screenshots here for the above task.

```
(kali㉿kali)-[~]
$ nmap 192.168.1.6
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-25 23:14 EDT
Nmap scan report for 192.168.1.6
Host is up (0.00064s latency).
All 1000 scanned ports on 192.168.1.6 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 08:00:27:61:C2:CE (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 25.17 seconds
```

- SSH and Web Access:
 - From Kali Linux: Try SSH into CentOS and access the web server (HTTP).
 - **How do you do this? What commands would you use? Briefly explain.**
 - **ANS: Use the `ssh osboxes@192.168.1.6` command and the `curl http://192.168.1.6` commands on separate lines to get each**
 - From Metasploitable: Try to SSH and access the web server (HTTP).
 - **What was the outcome?**
 - **ANS: Use the `ssh osboxes@192.168.1.6` command and the `curl http://192.168.1.6` commands on separate lines to get each**

Place screenshots here for the above tasks.

```
(kali㉿kali)-[~]
└─$ ssh osboxes@192.168.1.6
The authenticity of host '192.168.1.6 (192.168.1.6)' can't be established.
ED25519 key fingerprint is SHA256:iATSWWAQZ0BPLLYdseYHWiK7BrBC/XRs/6uWhzh6Hss.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.6' (ED25519) to the list of known hosts.
osboxes@192.168.1.6's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Tue Mar 25 21:12:43 2025
[osboxes@osboxes ~]$ exit
logout
Connection to 192.168.1.6 closed.

msfadmin@metasploitable:~$ curl http://192.168.1.6
curl: (7) couldn't connect to host
msfadmin@metasploitable:~$ ssh osboxes@192.168.1.6
no hostkey alg
```

- Now, Check the logs to ensure that unauthorized access and port scans are being logged appropriately.

```
sudo tail -f /var/log/messages
```

Place screenshots here for the above task. (journalctl -f instead)

```
sudo] password for osboxes:
- Journal begins at Tue 2025-03-25 21:09:01 EDT. --
Mar 25 23:12:15 osboxes sudo[4060]: pam_unix(sudo:session): session opened for
user root(uid=0) by (uid=1000)
Mar 25 23:12:15 osboxes sudo[4060]: pam_unix(sudo:session): session closed for
user root
Mar 25 23:12:56 osboxes systemd[1]: fwupd.service: Deactivated successfully.
Mar 25 23:12:56 osboxes sshd[4069]: Unable to negotiate with 192.168.1.4 port 5
075: no matching host key type found. Their offer: ssh-rsa,ssh-dss [preauth]
Mar 25 23:20:21 osboxes systemd[1]: Starting Fingerprint Authentication Daemon.
.
Mar 25 23:20:21 osboxes systemd[1]: Started Fingerprint Authentication Daemon.
Mar 25 23:20:24 osboxes gdm-password[4083]: gkr-pam: unlocked login keyring
Mar 25 23:20:24 osboxes NetworkManager[790]: <info> [1742959224.7957] agent-ma
nager: agent[eacf27a10148b498,:1.72/org.gnome.Shell.NetworkAgent/1000]: agent r
egistered
Mar 25 23:20:35 osboxes sudo[4106]: osboxes : TTY=pts/0 ; PWD=/home/osboxes ;
USER=root ; COMMAND=/bin/journalctl -f
Mar 25 23:20:35 osboxes sudo[4106]: pam_unix(sudo:session): session opened for
user root(uid=0) by (uid=1000)
```

As a last step, to list all iptables rules with line numbers (to identify specific rules to delete):

```
sudo iptables -L -n --line-numbers
```

This will display all rules for the INPUT, OUTPUT, and FORWARD chains with line numbers.

Place screenshots here for the above task.

```
[osboxes@osboxes ~]$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num  target      prot opt source                destination
1    DROP        all  --  192.168.1.5            0.0.0.0/0
2    ACCEPT      icmp --  0.0.0.0/0              0.0.0.0/0
3    ACCEPT      tcp  --  192.168.1.4            0.0.0.0/0      tcp
4    ACCEPT      tcp  --  192.168.1.5            0.0.0.0/0      tcp
5    ACCEPT      tcp  --  0.0.0.0/0              0.0.0.0/0      tcp
6    DROP        all  --  0.0.0.0/0              0.0.0.0/0
7    ACCEPT      tcp  --  192.168.1.4            0.0.0.0/0      tcp
8    ACCEPT      tcp  --  192.168.1.5            0.0.0.0/0      tcp
9    ACCEPT      tcp  --  0.0.0.0/0              0.0.0.0/0      tcp
10   DROP        all  --  0.0.0.0/0              0.0.0.0/0
11   ACCEPT      tcp  --  0.0.0.0/0              0.0.0.0/0      tcp
g 10/sec burst 20
12   ACCEPT      tcp  --  0.0.0.0/0              0.0.0.0/0      tcp
limit: avg 1/sec burst 5
13   LOG         tcp  --  0.0.0.0/0              0.0.0.0/0      tcp
flags 0 level 4 prefix "PORT SCAN: "
14   DROP        tcp  --  0.0.0.0/0              0.0.0.0/0      tcp
15   LOG         tcp  --  0.0.0.0/0              0.0.0.0/0      tcp
flags 0 level 4 prefix "PORT SCAN: "
16   DROP        all  --  192.168.1.5            0.0.0.0/0
17   DROP        all  --  192.168.1.5            0.0.0.0/0
18   ACCEPT      tcp  --  192.168.1.4            0.0.0.0/0      tcp
19   DROP        tcp  --  0.0.0.0/0              0.0.0.0/0      tcp
20   ACCEPT      tcp  --  192.168.1.4            0.0.0.0/0      tcp
21   ACCEPT      tcp  --  192.168.1.5            0.0.0.0/0      tcp
22   DROP        tcp  --  0.0.0.0/0              0.0.0.0/0      tcp
```

Answer the following questions:

- Explain the difference between ACCEPT, DROP, and REJECT targets in iptables.
 - ACCEPT: Allows the packet through.
 - DROP: Silently discards the packet (no response sent).
 - REJECT: Blocks the packet and sends back an error response to the sender.
- How do you add a rule to allow incoming SSH (port 22) connections from a specific IP (e.g., 192.168.1.100)?
 - `sudo iptables -A INPUT -p tcp -s 192.168.1.100 --dport 22 -j ACCEPT`
- How do you block all incoming traffic except for traffic on ports 80 (HTTP) and 443 (HTTPS)?

1. `sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT`
2. `sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT`
3. `sudo iptables -A INPUT -j DROP`
4. How can you persist iptables rules so they remain after a reboot?
 1. `sudo service iptables save`
5. How do you allow outgoing traffic but block all incoming connections except SSH?
 1. `sudo iptables -P OUTPUT ACCEPT`
 2. `sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT`
 3. `sudo iptables -A INPUT -j DROP`
6. What is stateful packet filtering, and how does iptables handle connection states?
 1. Stateful filtering tracks connection status (like NEW, ESTABLISHED, RELATED). iptables can allow traffic only if it's part of an existing connection.
7. How can you use iptables to limit the rate of incoming connections to prevent DoS attacks?
 1. `sudo iptables -A INPUT -p tcp --dport 80 -m limit --limit 10/s --limit-burst 20 -j ACCEPT`
8. What is NAT (Network Address Translation), and how can iptables be used for NAT?
 1. NAT changes the source or destination IP address in a packet to route traffic between private and public networks.
9. How do you forward packets from one network interface to another using iptables?
 1. `sudo iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT`
 2. `sudo iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT`
10. How can you log dropped packets using iptables?
 1. `sudo iptables -A INPUT -j LOG --log-prefix "DROPPED: " --log-level 4`

Additional Task: Managing the 'iptables' through some useful Commands

If you want to delete one rule, find its line number from the output of the previous command and delete it using:

```
sudo iptables -D INPUT <line_number>
```

For example, we can use the following command if rule 3 in the INPUT chain needs to be removed:

```
sudo iptables -D INPUT 3
```

To remove all rules from a specific chain (e.g., INPUT):

```
sudo iptables -F INPUT
```

To flush all chains (i.e., reset iptables):

```
sudo iptables -F
```

Save and Reload iptables Rules:

After modifying or clearing rules, save changes so they persist after a reboot:

```
sudo service iptables save
```

or

```
sudo iptables-save > /etc/sysconfig/iptables
```

To reload iptables:

```
sudo systemctl restart iptables
```

At this stage, all VMs should be able to PING each other as we have removed all the rules that were being enforced previously.

Deliverables:

1. A list of all `iptables` rules you have implemented.
2. Screenshots of your testing, including
 - Ping and Nmap scan results.
 - Log outputs for detected port scans and blocked traffic.
 - Results of SSH and HTTP access tests.
3. Submitting a report via Brightspace as a PDF file explaining:
 - The configuration of `iptables` rules for securing the CentOS server.
 - How you mitigated DoS attacks, port scanning, and unauthorized access.
 - Answering the questions.