A5: Firewall

2. Set up the network

adding route on lab2 and lab3:

sudo ip route add 192.168.2.0/24 via 192.168.0.10 dev enp0s8
sudo ip route add 192.168.0.0/24 via 192.168.2.10 dev enp0s8

enabling ipv4 forwarding on lab1:

sudo sysctl -w net.ipv4.conf.enp0s8.forwarding=1
sudo sysctl -w net.ipv4.conf.enp0s9.forwarding=1
sudo sysctl -w net.ipv4.conf.enp0s8.proxy_arp=1
sudo sysctl -w net.ipv4.conf.enp0s9.proxy_arp=1

sudo iptables -L # checking firewall rules

- Chains are associated with hooks
- Chains are triggered by hooks
- Chains determine when rules will be evaluated.
- hooks are implemented in the kernel level, each packet, whether ingress or egress, triggeres hooks
- There are five netfilter kernel hooks
- tables hold rules. Each family type e.g. ip, arp, nat, filter has separate table.

The iptables firewall works by interacting with the packet filtering hooks in the Linux kernel's networking stack. These kernel hooks are known as the netfilter framework. Every packet that passes through the networking layer (incoming or outgoing) will trigger these hooks, allowing programs to interact with the traffic at key points. The kernel modules associated with iptables register with these hooks in order to ensure that the traffic conforms to the conditions laid out by the firewall rules.

Reference:

Tables, chain, hook and rules

3. Implement packet filtering on the router

On lab3:
sudo apt-get install proftpd apache2

On lab2:
sudo apt install nmap
nmap -A -T4 192.168.2.11

```
vagrant@lab2:~$ nmap -A -T4 192.168.2.11
Starting Nmap 7.80 ( https://nmap.org ) at 2024-03-20 23:17 UTC
Nmap scan report for lab3 (192.168.2.11)
Host is up (0.0014s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp     ProFTPD
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.52 ((Ubuntu))
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/
submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.43 seconds
```

With firewall:

```
vagrant@lab2:~$
vagrant@lab2:~$ nmap -A -T4 192.168.2.11
Starting Nmap 7.80 ( https://nmap.org ) at 2024-03-21 15:24 UTC
Nmap scan report for lab3 (192.168.2.11)
Host is up (0.0016s latency).
Not shown: 995 filtered ports
PORT    STATE  SERVICE  VERSION
20/tcp  closed ftp-data
21/tcp  open   ftp      ProFTPD
22/tcp  open   ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; prot
ocol 2.0)
80/tcp  open   http     Apache httpd 2.4.52 ((Ubuntu))
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
443/tcp closed https
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https:/
/nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.30 seconds
```

On lab3:

```
vagrant@lab3:~$ nmap -A -T4 192.168.0.11
Starting Nmap 7.80 ( https://nmap.org ) at 2024-03-20 23:19 UTC
Nmap scan report for lab2 (192.168.0.11)
Host is up (0.0013s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/
submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.13 seconds
vagrant@lab3:~$
```

On lab1:
sudo nft add table ip filter

# creating a base chain associated with FORWARD hook
sudo nft add chain ip filter FORWARD { type filter hook forward priority 0 \; }

# drop all forwarding packet by default
sudo nft add rule ip filter FORWARD counter drop

```
# finds rule with handle
nft -a list table filter

# deleting a rule
sudo nft delete rule filter FORWARD handle 5

# delete all the rules in a chain with the following command:
nft flush chain filter output

# delete all the rules in a table
nft flush table filter


# adding a rule at a given position
nft add rule filter output position 8 ip daddr 127.0.0.8 drop # adding this
rule after handle 8

nft insert rule filter output position 8 ip daddr 127.0.0.8 drop # inserting
this rule before handle 8
```

Allow ping from lab2

```
sudo nft add rule ip filter FORWARD iifname "enp0s8" oifname "enp0s9" ip
protocol icmp icmp type echo-request counter accept

sudo nft add rule ip filter FORWARD iifname "enp0s9" oifname "enp0s8" ip
protocol icmp icmp type echo-reply counter accept
```

dropping and rejecting packet

```
sudo nft insert rule ip filter FORWARD position 3 iifname "enp0s8" oifname
"enp0s9" ip protocol icmp icmp type echo-request counter reject


sudo nft insert rule ip filter FORWARD position 25 iifname "enp0s8" oifname
"enp0s9" ip protocol icmp icmp type echo-request counter drop
```

Allow SSH from lab2 and to lab2

```
sudo nft insert rule ip filter FORWARD position 5 iifname "enp0s8" tcp dport
22 counter accept
sudo nft insert rule ip filter FORWARD position 5 oifname "enp0s8" tcp sport
22 counter accept

sudo nft insert rule ip filter FORWARD position 5 iifname "enp0s9" tcp dport
22 counter accept
```

```
sudo nft insert rule ip filter FORWARD position 5 oifname "enp0s9" tcp sport
22 counter accept

checking solution:
ssh vagrant@lab3  [from lab2]
ssh vagrant@lab2 [from lab3]
```

Allowing http transfer

```
sudo nft insert rule ip filter FORWARD position 5 iifname "enp0s8" tcp dport
{80, 443} counter accept
sudo nft insert rule ip filter FORWARD position 5 oifname "enp0s8" tcp sport
{80, 443} counter accept

checking solution:
curl http://lab3
```

Allowing ftp transfer

```
sudo nft insert rule ip filter FORWARD position 5 iifname "enp0s8" tcp dport
{20, 21} counter accept
sudo nft insert rule ip filter FORWARD position 5 oifname "enp0s8" tcp sport
{20, 21} counter accept

checking solution:
curl -o . ftp://vagrant:1234@lab3/home/vagrant/newfile
```

Test cases to test the rules:

1. checking reachability of a machine by ping
2. SSH can be restricted and limited through firewall.
3. Limiting web access by specifying which port can be used. For example we can block http (80) traffic and can allow only https for our server.
4. File transfer via FTP.
5. With ping, we can also specify which way traffic is allowed. For example, lab2 can ping lab3 but not the other way unless we set a rule for that.

```
# all the ruleset
table ip filter {
	chain FORWARD {
		type filter hook forward priority filter; policy accept;
		iifname "enp0s8" oifname "enp0s9" icmp type echo-request accept
		iifname "enp0s9" oifname "enp0s8" icmp type echo-reply accept
		iifname "enp0s8" tcp dport 22 counter packets 126 bytes 19212
accept
		oifname "enp0s8" tcp sport 22 counter packets 145 bytes 24902
accept
		iifname "enp0s9" tcp dport 22 counter packets 60 bytes 11411
accept
		oifname "enp0s9" tcp sport 22 counter packets 81 bytes 14755
accept
		iifname "enp0s8" tcp dport { 80, 443 } counter packets 218 bytes
16782 accept
		oifname "enp0s8" tcp sport { 80, 443 } counter packets 181 bytes
115268 accept
```

```
            iifname "enp0s8" tcp dport { 20, 21 } counter packets 109 bytes
6141 accept
            oifname "enp0s8" tcp sport { 20, 21 } counter packets 95 bytes
7107 accept
            counter packets 6343 bytes 344588 drop
      }
}
```

4. Implement a web proxy

Capture HTTP Response Headers from lab3:

curl -I http://lab3

squid installation and configuration

sudo apt update

sudo apt install squid

sudo systemctl status squid

sudo nano “/etc/squid/squid.conf”

acl localnet src 192.168.0.11/24 # add this line 1333

http_access allow localnet # uncomment this line 1552

http_port 3128 transparent # 2107

sudo iptables -t nat -A PREROUTING -i enp0s8 -p tcp --dport 80 -j REDIRECT --to-port 3128

sudo iptables -t nat -A PREROUTING -i enp0s8 -s 192.168.1.0/24 -p tcp --dport 80 -j REDIRECT --to-port 3128

adding firewall rule and forwarding port to 3128

nft add table ip nat

nft -- add chain ip nat prerouting { type nat hook prerouting priority -100 \; }

sudo nft add rule ip nat prerouting iifname "enp0s8" ip saddr 192.168.1.0/24 tcp dport 80 redirect to :3128

denying access to load lab3 webpage

acl lab3_web dst 192.168.2.11

http_access deny lab3_web

5. Implement a DMZ

```
# create a suitable table/chain setup for all further examples
add table nat
add chain nat prerouting { type nat hook prerouting priority dstnat; }
add chain nat postrouting { type nat hook postrouting priority srcnat; }

# redirect all traffic entering via eth0 to destination address 192.168.1.120
add rule nat prerouting iif eth0 dnat to 192.168.1.120

# translate source addresses of all packets leaving via eth0 to whatever locally
generated packets would use as source to reach the same destination

add rule nat postrouting oif eth0 masquerade

# redirect incoming TCP traffic for port 22 to port 2222

add rule nat prerouting tcp dport 8080 redirect to :80

sudo nft add rule nat prerouting tcp dport 8080 redirect to :80
```