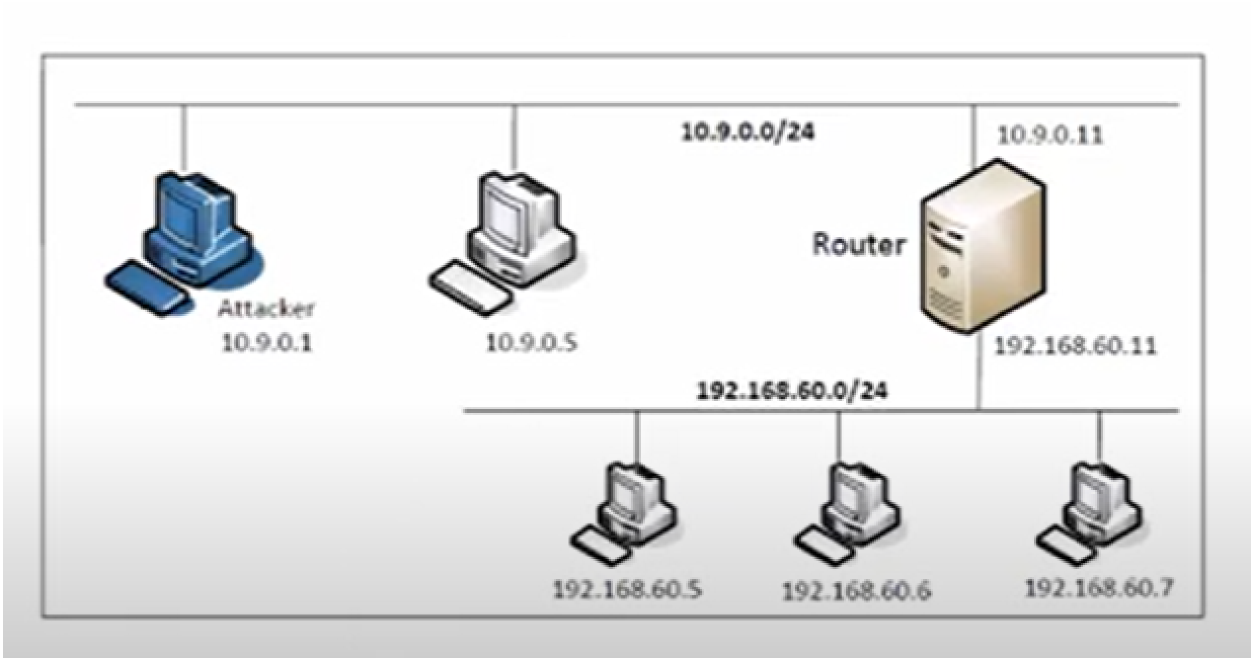


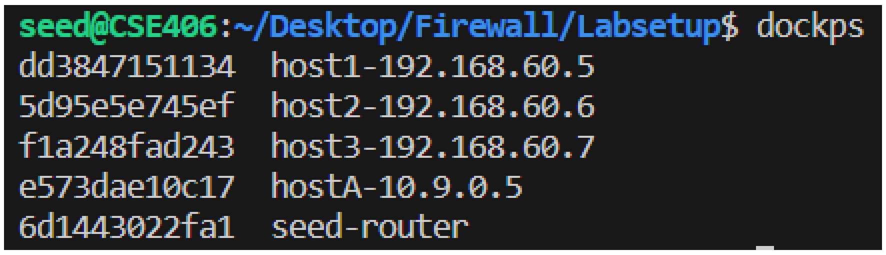
<u>Psst…to copy & paste into vs code terminal, copy from here then right click on the terminal</u>

Lab Setup

- Download the lab setup file
  ```
  wget
  https://seedsecuritylabs.org/Labs_20.04/Files/Firewall/Labsetup.z
  ip
  ```
- If the above one fails, try
  ```
  wget --no-check-certificate
  'https://docs.google.com/uc?export=download&id=1rCSKJp6CNQp6zT1L-
  09hjyDM-XYWou5Q' -O Labsetup.zip
  ```
- Unzip the lab setup file
  ```
  unzip Labsetup.zip
  ```
- Go to the unzipped folder
  ```
  cd Labsetup/
  ```
- Build the container image
  ```
  dcbuild
  ```
- Start the container
  ```
  dcup > ../output.log 2>&1 &
  ```
- Shut down the container
  ```
  dcdown
  ```

Docker Commands

- To see docker list: **`dockps`**
- To open terminal into a docker: **`docksh [docker_id_here]`**



```
seed@CSE406:~/Desktop/Firewall/Labsetup$ dockps
dd3847151134  host1-192.168.60.5
5d95e5e745ef  host2-192.168.60.6
f1a248fad243  host3-192.168.60.7
e573dae10c17  hostA-10.9.0.5
6d1443022fa1  seed-router
```

LKM (Loadable Kernel Module)

Check the modules of kernel: **`lsmod`**

- Prepare the module **<module_name>.c**
- Prepare a **Makefile**

```
obj-m += <module_name>.o
all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD)
modules
clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

- Run the **Makefile**
  `make`
- Clear the message buffer
  `sudo dmesg -C`
- Insert the module
  `sudo insmod <module_name>.ko`
- Check if the module is inserted
  `lsmod | grep <module_name>`
- The module should be working now
- Remove the module
  `sudo rmmod <module_name>`

***MakeFile is in Files folder inside the Labsetup/kernel_module folder**

Simple Firewall using NetFilter

This Firewall prints info about all packets and blocks UDP requests

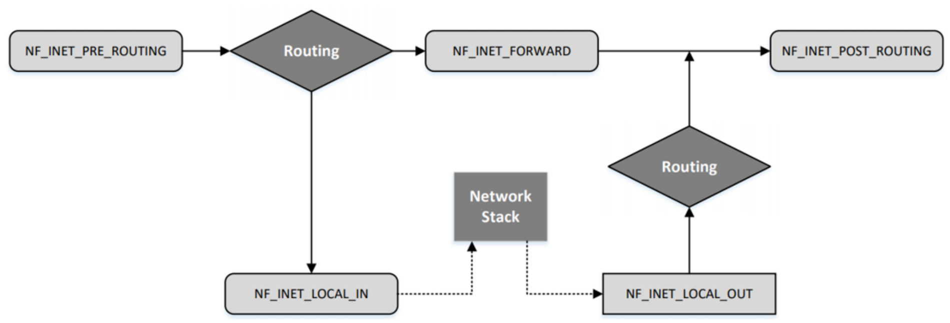
***Go to Labsetup/packet_filter**

- `make` - To make the seedFilter file
- `sudo insmod seedFilter.ko` - Inject the kernel object file
- `lsmod | grep seedFilter` - To check if the file's been properly injected
- `sudo dmesg -C` - Clear all d message
- `dig @8.8.8.8 www.example.com` – Sets up a dummy UDP request
- `sudo rmmod seedFilter` - Remove kernel object file (module)

```
struct iphdr   *iph   = ip_hdr(skb)    // (need to include <linux/ip.h>)
struct tcphdr  *tcph  = tcp_hdr(skb)   // (need to include <linux/tcp.h>)
struct udphdr  *udph  = udp_hdr(skb)   // (need to include <linux/udp.h>)
struct icmphdr *icmph = icmp_hdr(skb)  // (need to include <linux/icmp.h>)
```

Stateless Firewall using iptables

Netfilter Hooks



Verdict on Packets (Return Values)

- `NF_ACCEPT`: Let the packet flow through the stack
- `NF_DROP`: Discard the packet

- `NF_QUEUE`: Pass the packet to the user space
- `NF_STOLEN`: Tell netfilter to forget about this packet
- `NF_REPEAT`: Request the netfilter to call this module again.

Tables and Chains

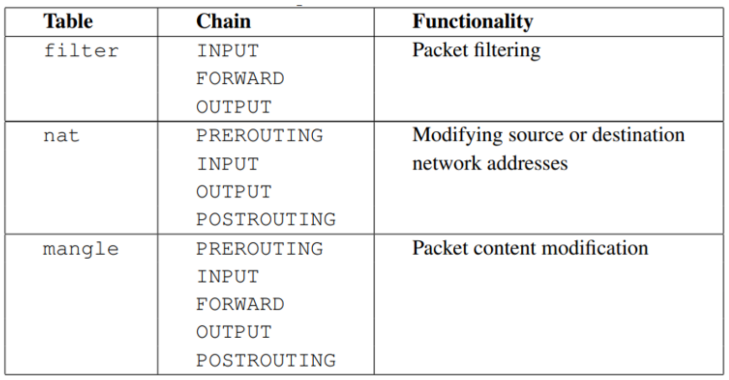


Table	Chain	Functionality

`filter`	`INPUT`
	`FORWARD`
	`OUTPUT`
		Packet filtering

`nat`	`PREROUTING`
	`INPUT`
	`OUTPUT`
	`POSTROUTING`
		Modifying source or destination network addresses

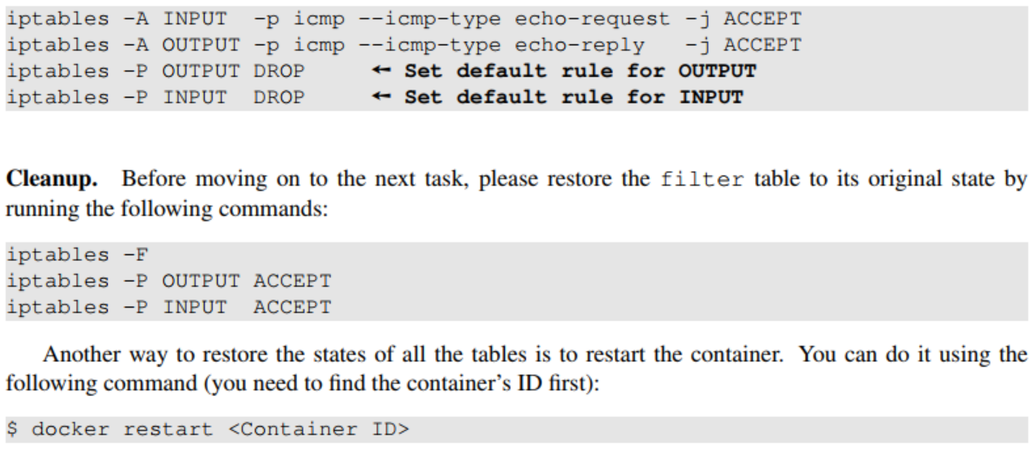
`mangle`	`PREROUTING`
	`INPUT`
	`FORWARD`
	`OUTPUT`
	`POSTROUTING`
		Packet content modification

```
iptables -t <table> -<operation> <chain>  <rule>   -j <target>
         ---------- -------------------- -------   -----------
            Table           Chain           Rule       Action
```

- **`sudo iptables -t nat -L -n`** - List all the rules in a table (without line number)
- **`sudo iptables -t filter -L -n --line-numbers`** – Same with line number
- **`sudo iptables -t filter -D INPUT 2`** – Delete rule No. 2 in the INPUT chain of the filter table
- **`sudo iptables -t filter -A INPUT <rule> -j DROP`** – Drop all the incoming packets that satisfy the <rule>

Firewall using iptable - None can access router except ping

- **`docksh <docker_id>`** - Open terminal in outside pc
- **`telnet 10.9.0.11`** - Telnet into router from outside pc [uid: **`seed;`** pass: **`dees`**]



```
iptables -A INPUT  -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type echo-reply   -j ACCEPT
iptables -P OUTPUT DROP      ← Set default rule for OUTPUT
iptables -P INPUT  DROP      ← Set default rule for INPUT
```

Cleanup. Before moving on to the next task, please restore the `filter` table to its original state by running the following commands:

```
iptables -F
iptables -P OUTPUT ACCEPT
iptables -P INPUT  ACCEPT
```

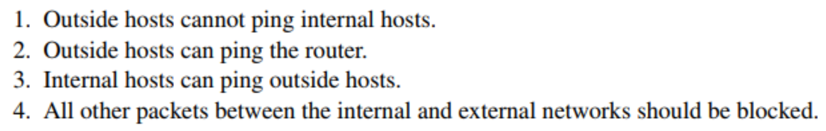
Another way to restore the states of all the tables is to restart the container. You can do it using the following command (you need to find the container's ID first):

```
$ docker restart <Container ID>
```

●

[Write these rules in router]

Firewall using iptable - Protecting internal network



1. Outside hosts cannot ping internal hosts.
2. Outside hosts can ping the router.
3. Internal hosts can ping outside hosts.
4. All other packets between the internal and external networks should be blocked.

- **`iptables -A FORWARD -p icmp --icmp-type echo-request -j DROP`** - Drops ICMP echo request (ping)
- **`iptables -A FORWARD -i eth0 -p tcp --sport 5000 -j ACCEPT`** - Allows the TCP packets coming from the interface eth0 if their source port is 5000

[Write these rules in router]

Stateful Firewall using iptables

- **`conntrack -L`** - Tracks all packets of router

- **`nc -lu 9090`** - Start a netcat UDP server on 192.168.60.5
- **`nc -u 192.168.60.5 9090`** - From 10.9.0.5, send out UDP packets [then write something and hit enter]

- **`nc -l 9090`** - Start a netcat TCP server on 192.168.60.5
- **`nc 192.168.60.5 9090`** - From 10.9.0.5, send out TCP packets [then write something and hit enter]
- **`iptables -A FORWARD -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT`** - Allows TCP packets belonging to an existing connection to pass through.
- **`iptables -A FORWARD -p tcp -i eth0 --dport 9090 --syn -m conntrack --ctstate NEW -j ACCEPT`** - The rule above does not cover the SYN packets, which do not belong to any established connection. Without it, we will not be able to create a connection in the first place. Therefore, we need to add a rule to accept incoming SYN packet
- **`iptables -P FORWARD DROP`** - Drop everything else

[Write these rules in router] \

- **`iptables -f`** - iptable Flush
- **`iptables -p FORWARD ACCEPT`** - Accept all

[These are used to reset router, that is, delete all rules]

Limiting Network Traffic

```
$ iptables -m limit -h
limit match options:
--limit avg              max average match rate: default 3/hour
                         [Packets per second unless followed by
                         /sec /minute /hour /day postfixes]
--limit-burst number     number to match in a burst, default 5
```

- **`iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute --limit-burst 5 -j ACCEPT`** - Per minute, 10 packets are allowed from 10.9.0.5 to the internal network, and at a time 5 packets are allowed
- **`iptables -A FORWARD -s 10.9.0.5 -j DROP`** - Drop everything else

[Ping to test]

iptable info
iptable info (2)
linux/netfilter.h
linux/netfilter_ipv4.h
linux/ip.h
linux/tcp.h

linux/udp.h
linux/if_ether.h
linux/inet.h
iptable & netfilter overview
Build Stateless Firewall
Netfilter example - Only accept TCP packets

Zarif Vai :3