## Lab Setup

* Download the lab setup file

**wget https://seedsecuritylabs.org/Labs\_20.04/Files/Firewall/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 &**

**[Eikhanei ashol jinis korbo]**

* Shut down the container

**dcdown**

**lsmod [eita likhle module gular nam dekhte parba]**

**kernel\_module folder e**

**make**

**Packet Filter folder e jeye**

**make**

**sudo insmod seedFilter.ko**

**lsmod | grep seedFilter**

**sudo dmesg –C**

**ping 1.1.1.1**

**sudo dmesg**

**sudo rmmod seedFilter**

**make**

**sudo insmod helloFilter.ko [jodi nam change kore dei, kothay change korbo don’t know]**

**lsmod | grep helloFilter**

**sudo dmesg –C**

**ping 1.1.1.1**

**sudo dmesg**

**sudo rmmod helloFilter**

**dig @8.8.8.8** [**www.example.com**](http://www.example.com)

**task 1 done**

**telnet 10.9.0.11**

**iptables –A FORWARD –p tcp –m conntrack –-ctstate ESTABLISHED, RELATED –j ACCEPT**

**iptables –A FORWARD –p tcp –I eth0 --dport 9090 –sym –m conntrack –ctstate NEW –j ACCEPT**

**iptables –P FORWARD DROP**

**iptables –F**

**iptables –P FORWARD ACCEPT**

**Task 4**

**iptables –A FORWARD –s 10.9.0.5 –m limit --limit 10/minute –limit-burst 5 –j ACCEPT**

**iptables –A FORWARD –s 10.9.0.5 –j DROP**

## LKM (Loadable Kernel Module)

* 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>**

## Useful Resources

* <https://www.digitalocean.com/community/tutorials/a-deep-dive-into-iptables-and-netfilter-architecture>
* <https://www.cloudsigma.com/the-architecture-of-iptables-and-netfilter/>
* <https://levelup.gitconnected.com/write-a-linux-firewall-from-scratch-based-on-netfilter-462013202686>
* <https://www3.cs.stonybrook.edu/~ezk/cse506-s19/handouts/kdk-Netfilter.pdf>
* <https://infosecwriteups.com/linux-kernel-communication-part-1-netfilter-hooks-15c07a5a5c4e>
* <https://www.opensourceforu.com/2022/08/building-a-stateless-firewall-using-netfilter-in-linux/>
* iptables manual
  + <https://man.cx/?page=iptables(8)>
  + <https://linux.die.net/man/8/iptables>
* header files
  + <https://github.com/torvalds/linux/blob/master/include/linux/netfilter.h>
  + <https://github.com/torvalds/linux/blob/master/include/linux/netfilter_ipv4.h>
  + <https://github.com/torvalds/linux/blob/master/include/linux/ip.h>
  + <https://github.com/torvalds/linux/blob/master/include/linux/tcp.h>
  + <https://github.com/torvalds/linux/blob/master/include/linux/udp.h>
  + <https://github.com/torvalds/linux/blob/master/include/linux/if_ether.h>
  + <https://github.com/torvalds/linux/blob/master/include/linux/inet.h>

**2 B**

**Outside hosts cannot ping internal hosts:**

iptables -A FORWARD -p icmp --icmp-type echo-request -i external\_interface -o internal\_interface -j DROP

interface er eikahne address dite hobe

**Outside hosts can ping the router:**

iptables -A FORWARD -p icmp --icmp-type echo-request -i external\_interface -d router\_ip -j ACCEPT

**Internal hosts can ping outside hosts:**

iptables -A FORWARD -p icmp --icmp-type echo-request -i internal\_interface -o external\_interface -j ACCEPT

**Block all other packets between the internal and external networks:**

iptables -A FORWARD -i internal\_interface -o external\_interface -j DROP

iptables -A FORWARD -i external\_interface -o internal\_interface -j DROP

After adding these rules, you can list the rules in the FORWARD chain to ensure they are correctly set up:

iptables -L FORWARD

Task 2C

To achieve the objectives outlined for protecting the TCP servers inside the internal network using iptables, follow these steps. For the purpose of this example, let's assume the interface names are eth0 for the external interface and eth1 for the internal interface.

1. **Allow Outside Hosts to Access Specific Telnet Server (192.168.60.5):**

bash

 sudo iptables -A FORWARD -i eth0 -o eth1 -p tcp -d 192.168.60.5 --dport 23 -j ACCEPT

 **Deny Outside Hosts Access to Other Internal Servers:**

bash

 sudo iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 23 -j DROP

 **Allow Internal Hosts to Access All Internal Servers:**

bash

 sudo iptables -A FORWARD -i eth1 -o eth1 -p tcp -m state --state NEW,ESTABLISHED -j ACCEPT

sudo iptables -A FORWARD -i eth1 -o eth1 -p tcp -m state --state ESTABLISHED -j ACCEPT

 **Deny Internal Hosts Access to External Servers:**

bash

 sudo iptables -A FORWARD -i eth1 -o eth0 -p tcp -j DROP

 **Clean Up:**

To clean up the rules, you can either flush all rules or specifically delete the added rules. In this example, we'll delete the added rules:

bash

 sudo iptables -D FORWARD -i eth0 -o eth1 -p tcp -d 192.168.60.5 --dport 23 -j ACCEPT

sudo iptables -D FORWARD -i eth0 -o eth1 -p tcp --dport 23 -j DROP

sudo iptables -D FORWARD -i eth1 -o eth1 -p tcp -m state --state NEW,ESTABLISHED -j ACCEPT

sudo iptables -D FORWARD -i eth1 -o eth1 -p tcp -m state --state ESTABLISHED -j ACCEPT

sudo iptables -D FORWARD -i eth1 -o eth0 -p tcp -j DROP

Repeat these commands for each rule you added.

 **Ensure Default Policies:**

Set the default policies for FORWARD to DROP, as specified in the task:

bash

1. sudo iptables -P FORWARD DROP

Remember to replace eth0 and eth1 with the actual names of your external and internal network interfaces. Also, keep in mind that these rules will be cleared upon a system restart. If you need to persist them across reboots, you'll need to set up a script or tool to apply them automatically during system startup.