Name: Dhanashree Reddy Srinivasa Reddy

SUID: 393473169

Course: Internet Security

Task 1: Implementing a Simple Firewall

Task 1.a: Implement a Simple Kernel Module

Initially, we navigate to the kernel module folder and compile the files into a loadable kernel module and then we use the make command to compile all the files.

```
[03/28/23]seed@VM:~/.../kernel_module$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Downloads/Labsetup/File
s/kernel_module modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Downloads/Labsetup/Files/kernel_module/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Downloads/Labsetup/Files/kernel_module/hello.mod.o
  LD [M]  /home/seed/Downloads/Labsetup/Files/kernel_module/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[03/28/23]seed@VM:~/.../kernel_module$ ls
hello.c   hello.mod    hello.mod.o  Makefile       Module.symvers
hello.ko  hello.mod.c  hello.o      modules.order
[03/28/23]seed@VM:~/.../kernel_module$ dmesg
[    0.000000] Linux version 5.4.0-54-generic (buildd@lcy01-amd64-024) (gcc vers
ion 9.3.0 (Ubuntu 9.3.0-17ubuntu1~20.04)) #60-Ubuntu SMP Fri Nov 6 10:37:59 UTC
2020 (Ubuntu 5.4.0-54.60-generic 5.4.65)
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-5.4.0-54-generic root=UUID
=a91f1a43-2770-4684-9fc3-b7abfd786c1d ro quiet splash
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Hygon HygonGenuine
[    0.000000]   Centaur CentaurHauls
```

```
[03/28/23]seed@VM:~/.../kernel_module$ lsmod | grep hello
[03/28/23]seed@VM:~/.../kernel_module$ lsmod
Module                  Size  Used by
xt_nat                 16384  20
xt_tcpudp              20480  20
veth                   28672  0
xt_conntrack           16384  4
xt_MASQUERADE          20480  4
nf_conntrack_netlink    45056  0
nfnetlink              16384  2 nf_conntrack_netlink
xfrm_user              36864  1
xfrm_algo              16384  1 xfrm_user
xt_addrtype            16384  2
iptable_filter         16384  1
iptable_nat            16384  6
nf_nat                 40960  3 xt_nat,iptable_nat,xt_MASQUERADE
nf_conntrack          139264  5 xt_conntrack,nf_nat,xt_nat,nf_conntrack_netlink,
xt_MASQUERADE
nf_defrag_ipv6         24576  1 nf_conntrack
nf_defrag_ipv4         16384  1 nf_conntrack
libcrc32c              16384  2 nf_conntrack,nf_nat
```

We use the dmesg command to display the messages.

```
[18719.131303] br-b9be54b991ea: port 2(vethb0041e0) entered blocking state
[18719.131307] br-b9be54b991ea: port 2(vethb0041e0) entered forwarding state
[19213.377107] hello: module verification failed: signature and/or required key
missing - tainting kernel
[19213.491622] Hello World!
```

Now we remove the hello.ko module and then use the dmesg to check the message.

We get the output as Bye-bye World!

```
[03/28/23]seed@VM:~/.../kernel_module$ sudo rmmod hello.ko
[03/28/23]seed@VM:~/.../kernel_module$ dmesg -k -w
[    0.000000] Linux version 5.4.0-54-generic (buildd@lcy01-amd64-024) (gcc vers
ion 9.3.0 (Ubuntu 9.3.0-17ubuntu1~20.04)) #60-Ubuntu SMP Fri Nov 6 10:37:59 UTC
2020 (Ubuntu 5.4.0-54.60-generic 5.4.65)
[18719.131250] IPv6: ADDRCONF(NETDEV_CHANGE): vethb0041e0: link becomes ready
[18719.131303] br-b9be54b991ea: port 2(vethb0041e0) entered blocking state
[18719.131307] br-b9be54b991ea: port 2(vethb0041e0) entered forwarding state
[19213.377107] hello: module verification failed: signature and/or required key
missing - tainting kernel
[19213.491622] Hello World!
[19486.995187] Bye-bye World!.
```

Task 1.b: Implement a Simple Firewall Using NetFilter

1. Navigate to the packet_filter folder and compile the seedFilter .file using the make command

```
[03/28/23]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Downloads/Labsetup/File
s/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedFilter.mod.o
  LD [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[03/28/23]seed@VM:~/.../packet_filter$ ls
Makefile       Module.symvers  seedFilter.ko   seedFilter.mod.c  seedFilter.o
modules.order  seedFilter.c    seedFilter.mod  seedFilter.mod.o
```

We use 'dig @8.8.8.8 www.example.com' command to generate UDP packets to 8.8.8.8 which is DNS google server.

```
[03/28/23]seed@VM:~/.../packet_filter$ dig @8.8.8.8 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31082
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        20252   IN      A       93.184.216.34

;; Query time: 56 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue Mar 28 17:47:03 EDT 2023
;; MSG SIZE  rcvd: 60
```

We use the 'insmod' which is the insert mode and then display the message using 'dmesg'

```
[03/28/23]seed@VM:~/.../packet_filter$ ls
Makefile       Module.symvers  seedFilter.ko   seedFilter.mod.c  seedFilter.o
modules.order  seedFilter.c    seedFilter.mod  seedFilter.mod.o
[03/28/23]seed@VM:~/.../packet_filter$ sudo insmod seedFilter.ko
[03/28/23]seed@VM:~/.../packet_filter$ lsmod | grep seedFilter
seedFilter             16384  0
[03/28/23]seed@VM:~/.../packet_filter$ dmesg -k -w
```

```
[19213.491622] Hello World!
[19486.995187] Bye-bye World!.
[20659.270814] Registering filters.
[20673.872276] *** LOCAL_OUT
[20673.872516]     10.0.2.7  --> 54.191.227.137 (TCP)
[20673.873258] *** LOCAL_OUT
[20673.873261]     10.0.2.7  --> 54.191.227.137 (TCP)
[20681.267457] *** LOCAL_OUT
[20681.267465]     10.0.2.7  --> 185.125.190.56 (UDP)
[20713.272124] *** LOCAL_OUT
[20713.272127]     10.0.2.7  --> 185.125.190.56 (UDP)
[20745.261832] *** LOCAL_OUT
[20745.261835]     10.0.2.7  --> 185.125.190.56 (UDP)
```

Now Google's DNS server is unreachable, which shows the firewall is working as expected.

```
[03/28/23]seed@VM:~/.../packet_filter$ dig @8.8.8.8 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

The packets are tried multiple times before being dropped.

```
[20771.266647]     127.0.0.1  --> 127.0.0.1 (UDP)
[20771.267786] *** LOCAL_OUT
[20771.267788]     10.0.2.7  --> 8.8.8.8 (UDP)
[20771.267804] *** Dropping 8.8.8.8 (UDP), port 53
[20776.283070] *** LOCAL_OUT
[20776.283073]     10.0.2.7  --> 8.8.8.8 (UDP)
[20776.283090] *** Dropping 8.8.8.8 (UDP), port 53
[20777.259255] *** LOCAL_OUT
[20777.259257]     10.0.2.7  --> 185.125.190.56 (UDP)
```

Task 1b 2:

We use the makefile to make the module

```
  GNU nano 4.8                                        Makefile
obj-m += seedFilter.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

ins:
        sudo dmesg -C
        sudo insmod seedFilter.ko

rm:
        sudo rmmod seedFilter
```

Add more hooks to the code- seedFilter.c file

```
static struct nf_hook_ops hook1, hook2, hook3, hook4, hook5;
```

Add the hooks in the following hook functions to register the hooks

```c
int registerFilter(void) {
   printk(KERN_INFO "Registering filters.\n");

   //NF_INET_PRE_ROUTING
   hook1.hook = printInfo;
   hook1.hooknum = NF_INET_PRE_ROUTING;
   hook1.pf = PF_INET;
   hook1.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook1);

   //NF_INET_POST_ROUTING
   hook2.hook = printInfo;
   hook2.hooknum = NF_INET_LOCAL_IN;
   hook2.pf = PF_INET;
   hook2.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook2);

   //NF_INET_FORWARD
   hook3.hook = printInfo;
   hook3.hooknum = NF_INET_FORWARD;
   hook3.pf = PF_INET;
   hook3.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook3);
```

```
//NF_INET_LOCAL_OUT
hook4.hook = printInfo;
hook4.hooknum = NF_INET_LOCAL_OUT;
hook4.pf = PF_INET;
hook4.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook4);

//NF_INET_POST_ROUTING
hook5.hook = printInfo;
hook5.hooknum = NF_INET_POST_ROUTING;
hook5.pf = PF_INET;
hook5.priority = NF_IP_PRI_FIRST;
nf_register_net_hook(&init_net, &hook5);
```

Also unregister the hooks in the removeFilter function

```
void removeFilter(void) {
    printk(KERN_INFO "The filters are being removed.\n");
    nf_unregister_net_hook(&init_net, &hook1);
    nf_unregister_net_hook(&init_net, &hook2);
    nf_unregister_net_hook(&init_net, &hook3);
    nf_unregister_net_hook(&init_net, &hook4);
    nf_unregister_net_hook(&init_net, &hook5);

}
```

Compile the file using the make command and insert the seedFilter.ko file using the insmod command.

```
[03/30/23]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Downloads/Labsetup/File
s/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  LD [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[03/30/23]seed@VM:~/.../packet_filter$ sudo insmod seedFilter.ko
[03/30/23]seed@VM:~/.../packet_filter$ dmesg
```

```
[  190.321542] seedFilter: module verification failed: signature and/or required
 key missing - tainting kernel
[  190.322277] Registering filters.
```

To generate the UDP packets use the dig @8.8.8.8 www.example.com and the corresponding functions are executed.

```
[03/30/23]seed@VM:~/.../packet_filter$ dig @8.8.8.8 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39369
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        20219   IN      A       93.184.216.34

;; Query time: 23 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Mar 30 23:58:21 EDT 2023
;; MSG SIZE  rcvd: 60
```

On using the dmesg command we get the following messages. We can see that LOCAL_OUT, LOCAL_IN, POST_ROUTING and PRE_ROUTING functions are invoked except the FORWARD function

```
[  233.958096] *** LOCAL_OUT
[  233.958098]     10.0.2.7  --> 8.8.8.8 (UDP)
[  233.958108] *** POST_ROUTING
[  233.958110]     10.0.2.7  --> 8.8.8.8 (UDP)
[  233.983711] *** PRE_ROUTING
[  233.983852]     8.8.8.8  --> 10.0.2.7 (UDP)
[  233.984095] *** LOCAL_IN
[  233.984185]     8.8.8.8  --> 10.0.2.7 (UDP)
```

Then the filters are removed and unregistered.

```
[  438.766679] *** PRE_ROUTING
[  438.766886]     185.125.190.57  --> 10.0.2.7 (UDP)
[  438.767100] *** LOCAL_IN
[  438.767206]     185.125.190.57  --> 10.0.2.7 (UDP)
[  475.448507] The filters are being removed.
```

Task 1b.3

For this task, create a new file called seedBlock.c

Inthis file we must include two main tasks

1.Preventing other computers to ping the VM, and

2.Preventing other computers to telnet into the VM.

The below code is implemented as follows:

Blocking ping to 8.8.8.8:53

```c
unsigned int blockUDP(void *priv, struct sk_buff *skb,
                      const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct udphdr *udph;

   u16  port   = 53;
   char ip[16] = "10.9.0.1";
   u32  ip_addr;

   if (!skb) return NF_ACCEPT;

   iph = ip_hdr(skb);
   // Convert the IPv4 address from dotted decimal to 32-bit binary
   in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);

   if (iph->protocol == IPPROTO_UDP) {
       udph = udp_hdr(skb);
       if (iph->daddr == ip_addr && ntohs(udph->dest) == port){
           printk(KERN_WARNING "*** Dropping %pI4 (UDP), port %d\n", &(iph->daddr),port);
           return NF_DROP;
        }
   }
   return NF_ACCEPT;
```

Blocking ping to VM:

```c
unsigned int blockICMP(void *priv, struct sk_buff *skb,
                       const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct icmphdr *icmph;

   //u16  port    = 53;
   char ip[16] = "10.9.0.1";
   u32  ip_addr;

   if (!skb) return NF_ACCEPT;

   iph = ip_hdr(skb);
   // Convert the IPv4 address from dotted decimal to 32-bit binary
   in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);

   if (iph->protocol == IPPROTO_ICMP) {
       icmph = icmp_hdr(skb);
       if (iph->daddr == ip_addr && icmph->type == ICMP_ECHO){
            printk(KERN_WARNING "*** Dropping %pI4 (ICMP)\n", &(iph->daddr));
            return NF_DROP;
        }
   }
   return NF_ACCEPT;
}
```

Blocking ping to Telnet to VM

```c
unsigned int blockTelnet(void *priv, struct sk_buff *skb,
                      const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct tcphdr *tcph;

   u16  port   = 23;
   char ip[16] = "10.9.0.1";
   u32  ip_addr;

   if (!skb) return NF_ACCEPT;

   iph = ip_hdr(skb);
   // Convert the IPv4 address from dotted decimal to 32-bit binary
   in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);

   if (iph->protocol == IPPROTO_TCP) {
       tcph = tcp_hdr(skb);
       if (iph->daddr == ip_addr && ntohs(tcph->dest) == port){
            printk(KERN_WARNING "*** Dropping %pI4 (TELNET), port %d\n", &(iph->daddr),port);
            return NF_DROP;
        }
   }
   return NF_ACCEPT;
```

We try to register the hooks to the functions.

```
    hook1.hook = printInfo;
    hook1.hooknum = NF_INET_LOCAL_OUT;
    hook1.pf = PF_INET;
    hook1.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook1);

    hook2.hook = blockUDP;
    hook2.hooknum = NF_INET_POST_ROUTING;
    hook2.pf = PF_INET;
    hook2.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook2);

    hook3.hook = blockICMP;
    hook3.hooknum = NF_INET_PRE_ROUTING;
    hook3.pf = PF_INET;
    hook3.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook3);

    hook4.hook = blockTelnet;
    hook4.hooknum = NF_INET_PRE_ROUTING;
    hook4.pf = PF_INET;
    hook4.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook4);
```

Unregister the hooks:

```
void removeFilter(void) {
   printk(KERN_INFO "The filters are being removed.\n");
   nf_unregister_net_hook(&init_net, &hook1);
   nf_unregister_net_hook(&init_net, &hook2);
   nf_unregister_net_hook(&init_net, &hook3);
   nf_unregister_net_hook(&init_net, &hook4);
}
```

Next modify the makefile and add the kernel module seedblock.o object to the file created.

```
 GNU nano 4.8                          Makefile
#obj-m += seedFilter.o
obj-m += seedBlock.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

ins:
        sudo dmesg -C
        sudo insmod seedFilter.ko

rm:
        sudo rmmod seedFilter
```

Now we use the command ‘make’ and make the kernel module.

```
[03/31/23]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Downloads/Labsetup/Files/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedBlock.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedBlock.mod.o
  LD [M]  /home/seed/Downloads/Labsetup/Files/packet_filter/seedBlock.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[03/31/23]seed@VM:~/.../packet_filter$ ls
Makefile       Module.symvers  seedBlock.ko   seedBlock.mod.c  seedBlock.o
modules.order  seedBlock.c     seedBlock.mod  seedBlock.mod.o  seedFilter.c
[03/31/23]seed@VM:~/.../packet_filter$ sudo insmod seedBlock.ko
[03/31/23]seed@VM:~/.../packet_filter$ dmesg
```

By using the command dmesg, we can see that the filters have been registered.

```
[  241.567051] Registering filters.
[  244.424888] *** LOCAL_OUT
[  244.424892]     10.0.2.7   --> 91.189.91.157 (UDP)
```

Now ping from the hostA, the ping is unsuccessfull.

```
[03/31/23]seed@VM:~/.../Labsetup$ dockps
c24d04d3f008  seed-router
d9a92071c671  host1-192.168.60.5
cea44110251b  host2-192.168.60.6
0862c0e8591d  hostA-10.9.0.5
482117ddabcb  host3-192.168.60.7
[03/31/23]seed@VM:~/.../Labsetup$ docksh 0862
root@0862c0e8591d:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
^C
--- 10.9.0.1 ping statistics ---
19 packets transmitted, 0 received, 100% packet loss, time 18430ms
```

we can see that the UDP packets are dropped

```
[ 1392.717162] *** Dropping 10.9.0.1 (ICMP)
[ 1393.726912] *** Dropping 10.9.0.1 (ICMP)
[ 1394.749596] *** Dropping 10.9.0.1 (ICMP)
[ 1395.772448] *** Dropping 10.9.0.1 (ICMP)
[ 1396.795217] *** Dropping 10.9.0.1 (ICMP)
[ 1397.832772] *** Dropping 10.9.0.1 (ICMP)
[ 1398.941895] *** Dropping 10.9.0.1 (ICMP)
[ 1399.961202] *** Dropping 10.9.0.1 (ICMP)
[ 1400.999884] *** Dropping 10.9.0.1 (ICMP)
[ 1402.007584] *** Dropping 10.9.0.1 (ICMP)
[ 1402.306190] *** LOCAL_OUT
[ 1402.306193]     10.0.2.7  --> 91.189.91.157 (UDP)
[ 1403.023875] *** Dropping 10.9.0.1 (ICMP)
```

When we try to telnet into the VM, we can see that it is unsuccessful.

```
root@0862c0e8591d:/# telnet 10.9.0.1
Trying 10.9.0.1...
^C
```

The TELNET packets have been dropped.

```
[ 1548.071463] *** Dropping 10.9.0.1 (TELNET), port 23
[ 1549.091912] *** Dropping 10.9.0.1 (TELNET), port 23
[ 1551.111468] *** Dropping 10.9.0.1 (TELNET), port 23
[ 1555.225258] *** Dropping 10.9.0.1 (TELNET), port 23
[ 1562.464832] *** LOCAL_OUT
[ 1562.464835]     10.0.2.7  --> 91.189.91.157 (UDP)
[ 1563.391262] *** Dropping 10.9.0.1 (TELNET), port 23
[ 1574.239383] *** Dropping 10.9.0.1 (TELNET), port 23
[ 1594.488779] *** LOCAL_OUT
```

Task 2: Experimenting with Stateless Firewall Rules

Task 2a: Protecting the router

Initially, we have to display the router container. By the command ‘ip addr’ the container information is as follows.

```
[03/31/23]seed@VM:~/.../Labsetup$ dockps
c24d04d3f008  seed-router
d9a92071c671  host1-192.168.60.5
cea44110251b  host2-192.168.60.6
0862c0e8591d  hostA-10.9.0.5
482117ddabcb  host3-192.168.60.7
[03/31/23]seed@VM:~/.../Labsetup$ docksh c2
root@c24d04d3f008:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
15: eth0@if16: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:0a:09:00:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.11/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
17: eth1@if18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:c0:a8:3c:0b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.60.11/24 brd 192.168.60.255 scope global eth1
       valid_lft forever preferred_lft forever
```

We can see that there are two interfaces with two Ips. Eth0 has ip 10.9.0.11 and Eth1 has ip 192.168.60.11

When we try to ping to interfaces eth0 and eth1 which is successful

```
root@0862c0e8591d:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.333 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.071 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.074 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.070 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.068 ms
c64 bytes from 10.9.0.11: icmp_seq=6 ttl=64 time=0.074 ms
64 bytes from 10.9.0.11: icmp_seq=7 ttl=64 time=0.071 ms
^C
--- 10.9.0.11 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6147ms
rtt min/avg/max/mdev = 0.068/0.108/0.333/0.091 ms
root@0862c0e8591d:/# ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.087 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.081 ms
64 bytes from 192.168.60.11: icmp_seq=3 ttl=64 time=0.072 ms
64 bytes from 192.168.60.11: icmp_seq=4 ttl=64 time=13.4 ms
^C
--- 192.168.60.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3073ms
rtt min/avg/max/mdev = 0.072/3.420/13.441/5.785 ms
```

We can telnet 10.9.0.11 which is successful.

```
root@0862c0e8591d:/# telnet 10.9.0.11
Trying 10.9.0.11...
Connected to 10.9.0.11.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
c24d04d3f008 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
```

Check the status of iptables.

```
root@c24d04d3f008:/# iptables -t filter -L -n
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
```

We set rules on the router to prevent external machines from accessing the router host. We check the status of the iptables.

The below rules is as follows:

1.iptables –A INPUT –p icmp –icmp-type echo-request –j ACCEPT

The above command accepts the echo requests when ping is triggered and blocks other requests

2. iptables –A INPUT –p icmp –icmp-type echo-reply –j ACCEPT

The above command accepts the reply to the user.

3.iptables -P OUTPUT DROP

4. iptables –P INPUT DROP

The above two commands are set to default which does not let other protocols have access to the router.

```
root@c24d04d3f008:/# iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
root@c24d04d3f008:/# iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
root@c24d04d3f008:/# -P OUTPUT DROP
bash: -P: command not found
root@c24d04d3f008:/# iptables -P OUTPUT DROP
root@c24d04d3f008:/# iptables -P INPUT DROP
root@c24d04d3f008:/# iptables -t filter -L -n
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 8
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 0

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

Ping to 192.168.60.11 is denied.

Telnet to 10.9.0.11 is denied too.

```
root@0862c0e8591d:/# ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
^C
--- 192.168.60.11 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4092ms

root@0862c0e8591d:/# telnet 10.9.0.11
Trying 10.9.0.11...
^C
```

For the next task, the filter iptables are reset.

```
root@c24d04d3f008:/# iptables -F
root@c24d04d3f008:/# iptables -P OUTPUT ACCEPT
root@c24d04d3f008:/# iptables -P INPUT ACCEPT
root@c24d04d3f008:/# iptables -t filter -L -n
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
```

Task 2.b: Protecting the Internal Network

Initially we check the status of the iptables before setting any rule for dropping ICMP echo requests

```
root@c24d04d3f008:/# iptables -L -n
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
root@c24d04d3f008:/# iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-request -j DROP
root@c24d04d3f008:/# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 8

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

From the rule set, we can see that the external hosts will not be able to ping the internal host because of the drop in packets.

The external host can ping the router.

```
root@0862c0e8591d:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.146 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.073 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.067 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.104 ms
^C
--- 10.9.0.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3095ms
```

The external host cannot ping the internal hosts.

```
root@0862c0e8591d:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5114ms
```

The internal host can ping the external hosts.

```
[03/31/23]seed@VM:~/.../Labsetup$ dockps
c24d04d3f008  seed-router
d9a92071c671  host1-192.168.60.5
cea44110251b  host2-192.168.60.6
0862c0e8591d  hostA-10.9.0.5
482117ddabcb  host3-192.168.60.7
[03/31/23]seed@VM:~/.../Labsetup$ docksh d9
root@d9a92071c671:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.180 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.101 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.085 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.107 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.138 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=63 time=0.084 ms
64 bytes from 10.9.0.5: icmp_seq=7 ttl=63 time=0.359 ms
64 bytes from 10.9.0.5: icmp_seq=8 ttl=63 time=0.084 ms
64 bytes from 10.9.0.5: icmp_seq=9 ttl=63 time=0.081 ms
^C
--- 10.9.0.5 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8171ms
rtt min/avg/max/mdev = 0.081/0.135/0.359/0.084 ms
```

All the other packets between the internal and external networks should be blocked.

We have to set a rule which accepts packets sent between internal host and external host. We have to set it for both eth0 and eth1.

```
root@c24d04d3f008:/# iptables -A FORWARD -i eth1 -p icmp --icmp-type echo-request -j ACCEPT
root@c24d04d3f008:/# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 8
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 8

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

```
root@c24d04d3f008:/# iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-reply -j ACCEPT
root@c24d04d3f008:/# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 8
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 8
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 0

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

The iptables as follows:

```
root@c24d04d3f008:/# iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    6   504 DROP       icmp --  eth0   *       0.0.0.0/0            0.0.0.0/0           icmptype 8
    0     0 ACCEPT     icmp --  eth1   *       0.0.0.0/0            0.0.0.0/0           icmptype 8
    0     0 ACCEPT     icmp --  eth0   *       0.0.0.0/0            0.0.0.0/0           icmptype 0

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

We try to ping between external and internal hosts:

It does not work between external host to internal host.

```
root@0862c0e8591d:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3076ms
```

It works from internal host to external host.

```
root@d9a92071c671:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.218 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.087 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.089 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.090 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.079 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=63 time=0.090 ms
^C
--- 10.9.0.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5115ms
rtt min/avg/max/mdev = 0.079/0.108/0.218/0.048 ms
```

For other connections requests we use telnet from internal host to external host.

Telnet does not work either way from external to internal host and vice versa.

This shows that the firewall is working as expected.

```
root@0862c0e8591d:/# telnet 192.168.60.5
Trying 192.168.60.5...
```

```
seed@d9a92071c671:~$ telnet 10.9.0.5
Trying 10.9.0.5...
```

For the task, we restore the filter iptables is restored.

```
root@c24d04d3f008:/# iptables -F
root@c24d04d3f008:/# iptables -P FORWARD ACCEPT
root@c24d04d3f008:/# iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Task 2.c: Protecting Internal Servers:

For this task, we write the following rules on the server router

```
root@c24d04d3f008:/# iptables -A FORWARD -i eth0 -p tcp -d 192.168.60.5 --dport 23 -j ACCEPT
root@c24d04d3f008:/# iptables -A FORWARD -i eth1 -p tcp -s 192.168.60.5 --sport 23 -j ACCEPT
root@c24d04d3f008:/# iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     tcp  --  eth0   *       0.0.0.0/0            192.168.60.5        tcp dpt:23
    0     0 ACCEPT     tcp  --  eth1   *       192.168.60.5         0.0.0.0/0           tcp spt:23

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Eth0: accepts requests from external to internal host on the destination Ip 192.168.60.5 and dport 23

Eth1: accepts requests from internal to external host on the destination Ip 192.168.60.5 and dport 23

The third rule mentioned above drops any other packets not intended to the IP or port specified.

The telnet connection to 192.168.60.7 and 192.168.60.6 does not work. But to 192.168.60.5 is successful as mentioned in the rules.

```
root@0862c0e8591d:/# telnet 192.168.60.6
Trying 192.168.60.6...
```

```
root@0862c0e8591d:/# telnet 192.168.60.7
Trying 192.168.60.7...
```

```
root@0862c0e8591d:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
d9a92071c671 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Fri Mar 31 15:06:35 UTC 2023 on pts/2
```

When we try to connect any internal host to internal server on port number 23, we see that it connects successfully.

```
root@cea44110251b:/# nc -u 192.168.60.5 9090
hello
Hello
hello
hello
Hello Dhanashree
Hello D
```

```
root@d9a92071c671:/# nc -lt 9090
^C
root@d9a92071c671:/# nc -lu 9090
Hello
hello
hello
hello
Hello Dhanashree
Hello D
```

Now, the internal host tries to connect to the external server which is not possible.

```
root@0862c0e8591d:/# nc 192.168.60.5 9090
root@0862c0e8591d:/# nc -ltv 9090
Listening on 0.0.0.0 9090
^C
```

Before moving to next task, we reset to original filter iptables:

```
root@c24d04d3f008:/# iptables -F
root@c24d04d3f008:/# iptables -P FORWARD ACCEPT
root@c24d04d3f008:/# iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Task 3: Connection Tracking and Stateful Firewall

Task 3a: Experiment with the Connection Tracking

ICMP Experiment:

For the icmp experiment we ping the 192.168.60.5 in the background to run continuously as follows:

```
root@c24d04d3f008:/# ping 192.168.60.5 &
[1] 105
root@c24d04d3f008:/# PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=64 time=0.090 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.086 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=0.093 ms
^C
root@c24d04d3f008:/# 64 bytes from 192.168.60.5: icmp_seq=5 ttl=64 time=0.068 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=64 time=0.074 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=64 time=0.070 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=64 time=0.124 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=64 time=0.097 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=64 time=0.069 ms
```

On the server router, we can check connection tracking information using the ‘conntrack –L’ command

For ICMP, each packet to be sent takes around 5 to 6 seconds.

```
root@c24d04d3f008:/# ping 192.168.60.5 &>/dev/null &
[1] 106
root@c24d04d3f008:/# conntrack -L
icmp     1 29 src=192.168.60.11 dst=192.168.60.5 type=8 code=0 id=86 src=192.168
.60.5 dst=192.168.60.11 type=0 code=0 id=86 mark=0 use=1
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=133 src=192.168.60.
5 dst=10.9.0.5 type=0 code=0 id=133 mark=0 use=1
tcp      6 431546 ESTABLISHED src=10.9.0.5 dst=10.9.0.11 sport=59944 dport=23 sr
c=10.9.0.11 dst=10.9.0.5 sport=23 dport=59944 [ASSURED] mark=0 use=1
icmp     1 29 src=192.168.60.11 dst=192.168.60.5 type=8 code=0 id=106 src=192.16
8.60.5 dst=192.168.60.11 type=0 code=0 id=106 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 4 flow entries have been shown.
root@c24d04d3f008:/# conntrack -L
icmp     1 29 src=192.168.60.11 dst=192.168.60.5 type=8 code=0 id=86 src=192.168
.60.5 dst=192.168.60.11 type=0 code=0 id=86 mark=0 use=1
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=133 src=192.168.60.
5 dst=10.9.0.5 type=0 code=0 id=133 mark=0 use=1
tcp      6 431140 ESTABLISHED src=10.9.0.5 dst=10.9.0.11 sport=59944 dport=23 sr
c=10.9.0.11 dst=10.9.0.5 sport=23 dport=59944 [ASSURED] mark=0 use=1
udp      17 0 src=10.9.0.5 dst=192.168.60.5 sport=47878 dport=9090 src=192.168.6
0.5 dst=10.9.0.5 sport=9090 dport=47878 mark=0 use=1
icmp     1 29 src=192.168.60.11 dst=192.168.60.5 type=8 code=0 id=106 src=192.16
8.60.5 dst=192.168.60.11 type=0 code=0 id=106 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 5 flow entries have been shown.
root@c24d04d3f008:/#
root@c24d04d3f008:/# conntrack -L
icmp     1 29 src=192.168.60.11 dst=192.168.60.5 type=8 code=0 id=86 src=192.168
.60.5 dst=192.168.60.11 type=0 code=0 id=86 mark=0 use=1
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=133 src=192.168.60.
5 dst=10.9.0.5 type=0 code=0 id=133 mark=0 use=1
tcp      6 431138 ESTABLISHED src=10.9.0.5 dst=10.9.0.11 sport=59944 dport=23 sr
c=10.9.0.11 dst=10.9.0.5 sport=23 dport=59944 [ASSURED] mark=0 use=1
icmp     1 29 src=192.168.60.11 dst=192.168.60.5 type=8 code=0 id=106 src=192.16
8.60.5 dst=192.168.60.11 type=0 code=0 id=106 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 4 flow entries have been shown.
```

UDP Experiment

```
[03/31/23]seed@VM:~/.../Labsetup$ docksh hostA-10.9.0.5
root@0862c0e8591d:/# nc -u 192.168.60.5 9090
hello
Hello
Hello Dhanashree
hello
^C
```

```
[03/31/23]seed@VM:~/.../Labsetup$ docksh host1-192.168.60.5
root@d9a92071c671:/# nc -lu 9090
hello
Hello
Hello Dhanashree
hello
^C
```

By using the 'conntrack –L' to track the packet connection we can see that the UDP connection stays for approx. 5 to 6 seconds.

```
icmp     1 29 src=192.168.60.11 dst=192.168.60.5 type=8 code=0 id=86 src=192.168
.60.5 dst=192.168.60.11 type=0 code=0 id=86 mark=0 use=1
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=133 src=192.168.60.
5 dst=10.9.0.5 type=0 code=0 id=133 mark=0 use=1
tcp      6 429260 ESTABLISHED src=10.9.0.5 dst=10.9.0.11 sport=59944 dport=23 sr
c=10.9.0.11 dst=10.9.0.5 sport=23 dport=59944 [ASSURED] mark=0 use=1
tcp      6 431060 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=39756 dport=90
90 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=39756 [ASSURED] mark=0 use=1
icmp     1 29 src=192.168.60.11 dst=192.168.60.5 type=8 code=0 id=106 src=192.16
8.60.5 dst=192.168.60.11 type=0 code=0 id=106 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 5 flow entries have been shown.
root@c24d04d3f008:/# conntrack -L
icmp     1 30 src=192.168.60.11 dst=192.168.60.5 type=8 code=0 id=86 src=192.168
.60.5 dst=192.168.60.11 type=0 code=0 id=86 mark=0 use=1
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=133 src=192.168.60.
5 dst=10.9.0.5 type=0 code=0 id=133 mark=0 use=1
tcp      6 429258 ESTABLISHED src=10.9.0.5 dst=10.9.0.11 sport=59944 dport=23 sr
c=10.9.0.11 dst=10.9.0.5 sport=23 dport=59944 [ASSURED] mark=0 use=1
tcp      6 431058 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=39756 dport=90
90 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=39756 [ASSURED] mark=0 use=1
icmp     1 29 src=192.168.60.11 dst=192.168.60.5 type=8 code=0 id=106 src=192.16
8.60.5 dst=192.168.60.11 type=0 code=0 id=106 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 5 flow entries have been shown.
```

Task 3.B: Setting up a stateful firewall

We set the following rules:

```
root@c24d04d3f008:/# iptables -A FORWARD -p tcp -i eth0 -d 192.168.60.5 --dport
23 --syn -m conntrack --ctstate NEW -j ACCEPT
root@c24d04d3f008:/# iptables -A FORWARD -i eth1 -p tcp --syn -m conntrack --cts
tate NEW -j ACCEPT
root@c24d04d3f008:/# iptables -A FORWARD -p tcp -m conntrack --ctstate RELATED,E
STABLISHED -j ACCEPT
root@c24d04d3f008:/# iptables -A FORWARD -p tcp -j DROP
root@c24d04d3f008:/# iptables -p FORWARD ACCEPT
iptables v1.8.4 (legacy): unknown protocol "forward" specified
Try `iptables -h' or 'iptables --help' for more information.
root@c24d04d3f008:/# iptables -p FORWARD ACCEPT
iptables v1.8.4 (legacy): unknown protocol "forward" specified
Try `iptables -h' or 'iptables --help' for more information.
root@c24d04d3f008:/# iptables -P FORWARD ACCEPT
```

First to establish a TCP connection we will have to accept the new sync packet. Any TCP packet which agrees with the above established and related rule will be accepted. Any other connection will be dropped.

To check the state of the iptables:

```
root@c24d04d3f008:/# iptables -L -n -v
Chain INPUT (policy ACCEPT 120 packets, 10080 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain FORWARD (policy ACCEPT 120 packets, 10080 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
          ctstate RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  eth0   *       0.0.0.0/0            192.168.60.5
          tcp dpt:23 flags:0x17/0x02 ctstate NEW
    0     0 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            0.0.0.0/0
          tcp flags:0x17/0x02 ctstate NEW
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
          ctstate RELATED,ESTABLISHED
    0     0 DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0


Chain OUTPUT (policy ACCEPT 120 packets, 10080 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

When we try to telnet to 10.9.0.5 external host. The telnet connection is successful.

```
root@d9a92071c671:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
0862c0e8591d login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Fri Mar 31 15:06:58 UTC 2023 from 192.168.60.5 on pts/4
```

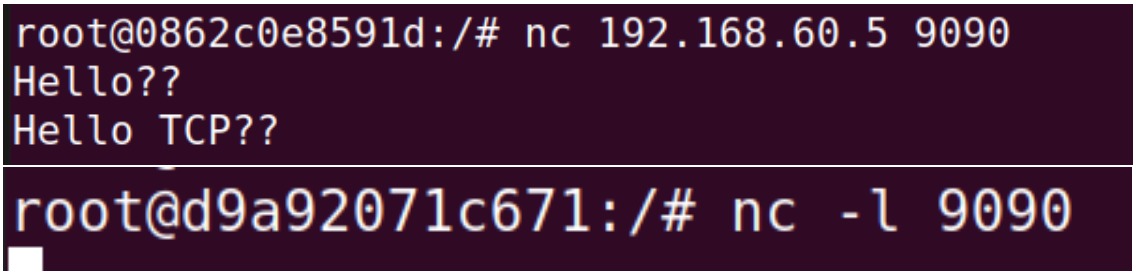
From the external host try to telnet the internal host 192.168.60.5. The telnet connection is successful.

```
root@0862c0e8591d:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
d9a92071c671 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Fri Mar 31 15:19:21 UTC 2023 on pts/2
```

As we have not mentioned the host 192.168.60.6 and 192.168.60.7 in the rule. The telnet connection is unsuccessful.

```
root@0862c0e8591d:/# telnet 192.168.60.6
Trying 192.168.60.6...
^C
root@0862c0e8591d:/# telnet 192.168.60.7
Trying 192.168.60.7...
^C
```

From the external host, we try to connect to the internal host through the netcap TCP server, which is not allowed.



For the next task, the filter iptables should be reset.

```
root@c24d04d3f008:/# iptables -F
root@c24d04d3f008:/# iptables -L -n -v
Chain INPUT (policy ACCEPT 42 packets, 3528 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain FORWARD (policy ACCEPT 42 packets, 3528 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 42 packets, 3528 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Task 4: Limiting Network Traffic

Set the below rule on server-router:

```
root@c24d04d3f008:/# iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute
--limit-burst 5 -j ACCEPT
```

From the host 192.168.60.5 ping to 10.9.0.5

From the screenshot below a connection is established but the number of packets that have been sent is not limited to the amount that can pass through the firewall.

```
root@d9a92071c671:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=6 ttl=63 time=0.095 ms
64 bytes from 10.9.0.5: icmp_seq=12 ttl=63 time=0.121 ms
64 bytes from 10.9.0.5: icmp_seq=18 ttl=63 time=0.107 ms
64 bytes from 10.9.0.5: icmp_seq=35 ttl=63 time=0.107 ms
64 bytes from 10.9.0.5: icmp_seq=41 ttl=63 time=0.109 ms
64 bytes from 10.9.0.5: icmp_seq=47 ttl=63 time=0.096 ms
64 bytes from 10.9.0.5: icmp_seq=53 ttl=63 time=0.092 ms
64 bytes from 10.9.0.5: icmp_seq=76 ttl=63 time=0.134 ms
64 bytes from 10.9.0.5: icmp_seq=82 ttl=63 time=0.084 ms
64 bytes from 10.9.0.5: icmp_seq=88 ttl=63 time=0.149 ms
64 bytes from 10.9.0.5: icmp_seq=94 ttl=63 time=0.110 ms
64 bytes from 10.9.0.5: icmp_seq=100 ttl=63 time=0.087 ms
64 bytes from 10.9.0.5: icmp_seq=117 ttl=63 time=0.083 ms
64 bytes from 10.9.0.5: icmp_seq=123 ttl=63 time=0.163 ms
64 bytes from 10.9.0.5: icmp_seq=129 ttl=63 time=0.088 ms
64 bytes from 10.9.0.5: icmp_seq=135 ttl=63 time=0.118 ms
64 bytes from 10.9.0.5: icmp_seq=141 ttl=63 time=0.100 ms
64 bytes from 10.9.0.5: icmp_seq=164 ttl=63 time=0.084 ms
64 bytes from 10.9.0.5: icmp_seq=170 ttl=63 time=0.082 ms
```

We add another rule in the server router $ iptables –A FORWARD –s 10.9.0.5 -j DROP, in which the packets not dropped from the first rule will be dropped from the second rule which does not match the first rule.

Now we again ping the 10.9.0.5

```
root@d9a92071c671:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.174 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.156 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.091 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.449 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.121 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=63 time=0.098 ms
64 bytes from 10.9.0.5: icmp_seq=7 ttl=63 time=0.122 ms
64 bytes from 10.9.0.5: icmp_seq=8 ttl=63 time=0.115 ms
64 bytes from 10.9.0.5: icmp_seq=9 ttl=63 time=0.204 ms
64 bytes from 10.9.0.5: icmp_seq=10 ttl=63 time=0.084 ms
64 bytes from 10.9.0.5: icmp_seq=11 ttl=63 time=0.128 ms
64 bytes from 10.9.0.5: icmp_seq=12 ttl=63 time=0.089 ms
64 bytes from 10.9.0.5: icmp_seq=13 ttl=63 time=0.201 ms
^C
--- 10.9.0.5 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12225ms
rtt min/avg/max/mdev = 0.084/0.156/0.449/0.093 ms
```

Now there Is a time condition 10/minute in which a packet is sent every 6 seconds approx. For this the second rule set above is required.

For the next task, we restore the table to its original state.

```
root@c24d04d3f008:/# iptables -F
root@c24d04d3f008:/# iptables -L -n -v
Chain INPUT (policy ACCEPT 6 packets, 504 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 6 packets, 504 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 6 packets, 504 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Task 5: Load Balancing

On the seed-router set the following rule:

By setting the below rule it dispatches the packets coming to 8080 to the inner hosts.

```
[03/31/23]seed@VM:~/.../Labsetup$ docksh cb43
root@cb438ffbc5f3:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --ev
ery 3 --packet 0 -j DNAT --to-destination 192.168.60.5:8080
root@cb438ffbc5f3:/#
```

We should start the server on each of the hosts 192.168.60.5, 192.168.60.6, 192.168.60.7.

From the external host hostA-10.9.0.5 we run the below command to send an UDP packet to the router's 8080 port.

```
[03/31/23]seed@VM:~/.../Labsetup$ dockps
4a72a39cd4fa  host1-192.168.60.5
cb438ffbc5f3  seed-router
c07bbe19d229  host3-192.168.60.7
b9452ba1491c  host2-192.168.60.6
d72556944e39  hostA-10.9.0.5
[03/31/23]seed@VM:~/.../Labsetup$ docksh hostA-10.9.0.5
root@d72556944e39:/# echo hello | nc -u 10.9.0.11 8080
^C
```

One out of 3 packets reach the host 192.168.60.5

```
[03/31/23]seed@VM:~/.../Labsetup$ docksh host1-192.168.60.5
root@4a72a39cd4fa:/# nc -luk 8080
hello
^C
```

On the server router we had two more rules for the host 192.168.60.6 and 192.168.60.7, so that all the three internal hosts get equal number of packets.

```
root@cb438ffbc5f3:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --ev
ery 2 --packet 0 -j DNAT --to-destination 192.168.60.6:8080
root@cb438ffbc5f3:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --ev
ery 1 --packet 0 -j DNAT --to-destination 192.168.60.7:8080
```

```
root@cb438ffbc5f3:/# iptables -t nat -L -n -v
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    1    34 DNAT       udp  --  *      *       0.0.0.0/0            0.0.0.0/0            udp dpt:80
80 statistic mode nth every 3 to:192.168.60.5:8080
    0     0 DNAT       udp  --  *      *       0.0.0.0/0            0.0.0.0/0            udp dpt:80
80 statistic mode nth every 2 to:192.168.60.6:8080
    0     0 DNAT       udp  --  *      *       0.0.0.0/0            0.0.0.0/0            udp dpt:80
80 statistic mode nth every 1 to:192.168.60.7:8080

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DOCKER_OUTPUT  all  --  *      *       0.0.0.0/0            127.0.0.11

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DOCKER_POSTROUTING  all  --  *      *       0.0.0.0/0            127.0.0.11
```

```
Chain DOCKER_OUTPUT (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            127.0.0.11           tcp dpt:53
 to:127.0.0.11:46359
    0     0 DNAT       udp  --  *      *       0.0.0.0/0            127.0.0.11           udp dpt:53
 to:127.0.0.11:56118

Chain DOCKER_POSTROUTING (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 SNAT       tcp  --  *      *       127.0.0.11           0.0.0.0/0            tcp spt:46
359 to::53
    0     0 SNAT       udp  --  *      *       127.0.0.11           0.0.0.0/0            udp spt:56
118 to::53
```

When we send the message repeatedly while we can see that all the internal hosts get equal number of packets.

```
[03/31/23]seed@VM:~/.../Labsetup$ dockps
4a72a39cd4fa  host1-192.168.60.5
cb438ffbc5f3  seed-router
c07bbe19d229  host3-192.168.60.7
b9452ba1491c  host2-192.168.60.6
d72556944e39  hostA-10.9.0.5
[03/31/23]seed@VM:~/.../Labsetup$ docksh hostA-10.9.0.5
root@d72556944e39:/# echo hello | nc -u 10.9.0.11 8080
^C
root@d72556944e39:/# echo hello | nc -u 10.9.0.11 8080
^C
root@d72556944e39:/# echo "hello D hello D" | nc -u 10.9.0.11 8080
^C
root@d72556944e39:/# echo "hello D hello D" | nc -u 10.9.0.11 8080
^C
root@d72556944e39:/# echo "hello D hello D" | nc -u 10.9.0.11 8080
```

```
[03/31/23]seed@VM:~/.../Labsetup$ dockps
4a72a39cd4fa  host1-192.168.60.5
cb438ffbc5f3  seed-router
c07bbe19d229  host3-192.168.60.7
b9452ba1491c  host2-192.168.60.6
d72556944e39  hostA-10.9.0.5
[03/31/23]seed@VM:~/.../Labsetup$ docksh host1-192.168.60.5
root@4a72a39cd4fa:/# nc -luk 8080
hello
^C
root@4a72a39cd4fa:/# nc -luk 8080
hello D hello D
```

```
[03/31/23]seed@VM:~/.../Labsetup$ dockps
4a72a39cd4fa  host1-192.168.60.5
cb438ffbc5f3  seed-router
c07bbe19d229  host3-192.168.60.7
b9452ba1491c  host2-192.168.60.6
d72556944e39  hostA-10.9.0.5
[03/31/23]seed@VM:~/.../Labsetup$ docksh host2-192.168.60.6
root@b9452ba1491c:/# nc -luk 8080
hello D hello D
^C
```

```
[03/31/23]seed@VM:~/.../Labsetup$ dockps
4a72a39cd4fa  host1-192.168.60.5
cb438ffbc5f3  seed-router
c07bbe19d229  host3-192.168.60.7
b9452ba1491c  host2-192.168.60.6
d72556944e39  hostA-10.9.0.5
[03/31/23]seed@VM:~/.../Labsetup$ docksh host3-192.168.60.7
root@c07bbe19d229:/# nc -luk 8080
hello D hello D
```

Using the random mode:

The below rules can be modified based on the random probability mode with random probabilities assigned to the packets sent to different hosts.

```
root@cb438ffbc5f3:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.3333 -j DNAT --to-destination 192.168.60.5:8080
root@cb438ffbc5f3:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.5 -j DNAT --to-destination 192.168.60.6:8080
root@cb438ffbc5f3:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 1 -j DNAT --to-destination 192.168.60.7:8080
```

```
root@cb438ffbc5f3:/# iptables -t nat -L -n --line-numbers
Chain PREROUTING (policy ACCEPT)
num  target     prot opt source               destination
1    DNAT       udp  --  0.0.0.0/0            0.0.0.0/0            udp dpt:8080 statistic mode nth
every 3 to:192.168.60.5:8080
2    DNAT       udp  --  0.0.0.0/0            0.0.0.0/0            udp dpt:8080 statistic mode nth
every 2 to:192.168.60.6:8080
3    DNAT       udp  --  0.0.0.0/0            0.0.0.0/0            udp dpt:8080 statistic mode nth
every 1 to:192.168.60.7:8080
4    DNAT       udp  --  0.0.0.0/0            0.0.0.0/0            udp dpt:8080 statistic mode nth
every 2 to:192.168.60.6:8080
5    DNAT       udp  --  0.0.0.0/0            0.0.0.0/0            udp dpt:8080 statistic mode nth
every 1 to:192.168.60.7:8080
6    DNAT       udp  --  0.0.0.0/0            0.0.0.0/0            udp dpt:8080 statistic mode rand
om probability 0.33330000006 to:192.168.60.5:8080
7    DNAT       udp  --  0.0.0.0/0            0.0.0.0/0            udp dpt:8080 statistic mode rand
om probability 0.50000000000 to:192.168.60.6:8080
8    DNAT       udp  --  0.0.0.0/0            0.0.0.0/0            udp dpt:8080 statistic mode rand
om probability 1.00000000000 to:192.168.60.7:8080

Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
1    DOCKER_OUTPUT  all  --  0.0.0.0/0            127.0.0.11

Chain POSTROUTING (policy ACCEPT)
num  target     prot opt source               destination
1    DOCKER_POSTROUTING  all  --  0.0.0.0/0            127.0.0.11

Chain DOCKER_OUTPUT (1 references)
num  target     prot opt source               destination
1    DNAT       tcp  --  0.0.0.0/0            127.0.0.11           tcp dpt:53 to:127.0.0.11:46359
2    DNAT       udp  --  0.0.0.0/0            127.0.0.11           udp dpt:53 to:127.0.0.11:56118

Chain DOCKER_POSTROUTING (1 references)
num  target     prot opt source               destination
1    SNAT       tcp  --  127.0.0.11           0.0.0.0/0            tcp spt:46359 to::53
2    SNAT       udp  --  127.0.0.11           0.0.0.0/0            udp spt:56118 to::53
```

From the host A sends out packets to different internal hosts.

```
root@d72556944e39:/# echo "Hello 1" | nc -u 10.9.0.11 8080
^C
root@d72556944e39:/# echo "Hello 2" | nc -u 10.9.0.11 8080
^C
root@d72556944e39:/# echo "Hello 3" | nc -u 10.9.0.11 8080
^C
root@d72556944e39:/# echo "Hello 4" | nc -u 10.9.0.11 8080
^C
root@d72556944e39:/# echo "Hello 5" | nc -u 10.9.0.11 8080
^C
root@d72556944e39:/# echo "Hello 6" | nc -u 10.9.0.11 8080
^C
root@d72556944e39:/# echo "Hello 7" | nc -u 10.9.0.11 8080
^C
root@d72556944e39:/# echo "Hello 8" | nc -u 10.9.0.11 8080
^C
root@d72556944e39:/# echo "Hello 9" | nc -u 10.9.0.11 8080
^C
```

Now based on the probabilities, packets are sent to different hosts randomly.

```
root@4a72a39cd4fa:/# nc -luk 8080
Hello 3
Hello 6
Hello 9
```

```
root@b9452ba1491c:/# nc -luk 8080
Hello 1
Hello 4
Hello 7
```

```
root@c07bbe19d229:/# nc -luk 8080
Hello 2
Hello 5
Hello 8
```