Lab 4

	IP Address	MAC Address
A (74494086821b)	10.9.0.5	02:42:0a:09:00:05
B (ddab801d6da7)	10.9.0.6	02:42:0a:09:00:06
M (a6669b729782)	10.9.0.105	02:42:0a:09:00:69

```
root@74494086821b:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
        RX packets 31  bytes 3655 (3.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 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
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
root@ddab801d6da7:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.6  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:06  txqueuelen 0  (Ethernet)
        RX packets 32  bytes 3762 (3.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 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
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
root@a6669b729782:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.105  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:69  txqueuelen 0  (Ethernet)
        RX packets 31  bytes 3715 (3.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 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
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Task 1: Implementing a Simple Firewall

Task 1A:

Below, I generated the LKM, hello.ko by downloading the lab zip file from the website, going into files, then using the command ‘make’.

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup# ls
Labsetup  Labsetup.zip  docker-compose.yml  volumes
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup# cd Labsetup
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup# ls
Files  docker-compose.yml  router  volumes
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup# cd Files
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files# ls
kernel_module  packet_filter
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files# cd
root@ubuntu-s-1vcpu-2gb-nyc1-01:~# cd Labsetup
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup# ls
Labsetup  Labsetup.zip  docker-compose.yml  volumes
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup# cd Labsetup
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup# make
make: *** No targets specified and no makefile found.  Stop.
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup# cd Files
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files# ls
kernel_module  packet_filter
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files# cd kernel_module
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/kernel_module# ls
Makefile  hello.c
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/kernel_module# make
make -C /lib/modules/5.4.0-122-generic/build M=/root/Labsetup/Labsetup/Files/kernel_module modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-122-generic'
  CC [M]  /root/Labsetup/Labsetup/Files/kernel_module/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /root/Labsetup/Labsetup/Files/kernel_module/hello.mod.o
  LD [M]  /root/Labsetup/Labsetup/Files/kernel_module/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-122-generic'
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/kernel_module# ls
Makefile  Module.symvers  hello.c  hello.ko  hello.mod  hello.mod.c  hello.mod.o  hello.o  modules.order
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/kernel_module#
```

Loading hello.ko into the kernel and using modinfo on it:

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/kernel_module# sudo insmod hello.ko
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/kernel_module# modinfo hello.ko
filename:       /root/Labsetup/Labsetup/Files/kernel_module/hello.ko
license:        GPL
srcversion:     717A72281ACFAA8385B33A8
depends:
retpoline:      Y
name:           hello
vermagic:       5.4.0-122-generic SMP mod_unload modversions
```

Task 1B:

1. Below, seedFilter.c is compiled using make in the packet_filter file.

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/kernel_module# make
make -C /lib/modules/5.4.0-122-generic/build M=/root/Labsetup/Labsetup/Files/kernel_module modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-122-generic'
  CC [M]  /root/Labsetup/Labsetup/Files/kernel_module/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /root/Labsetup/Labsetup/Files/kernel_module/hello.mod.o
  LD [M]  /root/Labsetup/Labsetup/Files/kernel_module/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-122-generic'
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/kernel_module# ls
Makefile  Module.symvers  hello.c  hello.ko  hello.mod  hello.mod.c  hello.mod.o  hello.o  modules.order
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/kernel_module# modinfo hello.ko
filename:       /root/Labsetup/Labsetup/Files/kernel_module/hello.ko
license:        GPL
srcversion:     717A72281ACFAA8385B33A8
depends:
retpoline:      Y
name:           hello
vermagic:       5.4.0-122-generic SMP mod_unload modversions
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/kernel_module# ls
Makefile  Module.symvers  hello.c  hello.ko  hello.mod  hello.mod.c  hello.mod.o  hello.o  modules.order
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/kernel_module# cd ..
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files# ls
kernel_module  packet_filter
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files# cd packet_filter
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/packet_filter# ls
Makefile  seedFilter.c
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/packet_filter# make
make -C /lib/modules/5.4.0-122-generic/build M=/root/Labsetup/Labsetup/Files/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-122-generic'
  CC [M]  /root/Labsetup/Labsetup/Files/packet_filter/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /root/Labsetup/Labsetup/Files/packet_filter/seedFilter.mod.o
  LD [M]  /root/Labsetup/Labsetup/Files/packet_filter/seedFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-122-generic'
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/packet_filter# ls
Makefile        modules.order  seedFilter.ko   seedFilter.mod.c  seedFilter.o
Module.symvers  seedFilter.c   seedFilter.mod  seedFilter.mod.o
```

Next, seedFilter.ko is loaded into the kernel and UDP packets are sent to Google's DNS server. The packets were blocked as shown by the message "connection timed out; no servers could be reached".

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/packet_filter# sudo insmod seedFilter.ko
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/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
```

Note: I had forgotten to load the lkm to the kernel initially and my packets were sent successfully, so I figured I did something wrong, below is my output that prompted me to realize my mistake

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/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: 18663
;; 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.        19612   IN      A       93.184.216.34

;; Query time: 4 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Nov 28 00:50:15 UTC 2022
;; MSG SIZE  rcvd: 60
```

2. Below, three more hook objects added

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

Following is the filter information for each hook condition

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

   //NF_INET_LOCAL_OUT
   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);

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

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

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

   //NF_INET_FORWARD
   hook5.hook = printInfo;
   hook5.hooknum = NF_INET_FORWARD;
   hook5.pf = PF_INET;
   hook5.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook5);
   return 0;
}
```

Updated hook de-registrations at the end

```
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);
}
```

3. To implement the two new hook functions, I first created two additional hook objects, hook3 and hook4 below.

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

Shown below is my function to prevent ICMP pinging to IP address 10.9.0.1. I used the same function from blockUDP, but changed the IP address and made sure to check that the icmp message was a reply.

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

   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;
}
```

To prevent Telnet communications, I set port to port 23, and blocked that destination port in the if statement checking the tcp packet header.

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

   u16  port   = 23; //telnet
   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 (TCP), port %d\n", &(iph->daddr), port);
            return NF_DROP;
        }
   }
   return NF_ACCEPT;
}
```

Below I implemented my hooks and unregistered them as well.

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

   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);

   return 0;
}

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);
}
```

After preventing ICMP pings to 10.9.0.1, the result from trying to ping is:

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/packet_filter# insmod seedFilter1B3.ko
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/packet_filter# 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 ---
58 packets transmitted, 0 received, 100% packet loss, time 58375ms
```

The result from trying to Telnet after blocking Telnet communications from port 23 is:

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:~/Labsetup/Labsetup/Files/packet_filter# telnet 10.9.0.1
Trying 10.9.0.1...
```

My droplet restarted…
246380d2481e host2-192.168.60.6
5fe863423438 hostA-10.9.0.5

848960a47faa host3-192.168.60.7
335fb99ce82b host1-192.168.60.5
28598c9f9891 seed-router

Task 2: Experimenting with Stateless Firewall Rules

Task 2A:

Below, I implemented the following rules:

```
root@28598c9f9891:/# iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
root@28598c9f9891:/# iptables -t filter -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 8

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@28598c9f9891:/# iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
root@28598c9f9891:/# iptables -t filter -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 8

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0            icmptype 0
root@28598c9f9891:/# iptables -P OUTPUT DROP
root@28598c9f9891:/# iptables -P INPUT DROP
root@28598c9f9891:/# 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

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

I first allow all input and output from icmp, then I give rules to drop both of them. The telnet could no longer connect after that.

```
root@5fe863423438:/# telnet 10.9.0.11
Trying 10.9.0.11...
^C
```

Task 2B:

1. I implemented the following rule to block outside hosts from pinging internal hosts.

```
root@28598c9f9891:/# iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-request -j DROP
root@28598c9f9891:/# iptables -t filter -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
```

   As shown in the screenshot, the outside host can ping the serve but cannot ping another internal host.

```
root@5fe863423438:/# 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=0.190 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.093 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.089 ms
^C
--- 10.9.0.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2049ms
rtt min/avg/max/mdev = 0.089/0.124/0.190/0.046 ms
root@5fe863423438:/# 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 ---
12 packets transmitted, 0 received, 100% packet loss, time 11243ms
```

2. Rule for allowing outside hosts to ping router, done by accepting replies on eth0:

```
root@28598c9f9891:/# iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-reply -j ACCEPT
```

   Works:

```
root@335fb99ce82b:/# 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=0.128 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.123 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.108 ms
^C
--- 10.9.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.108/0.119/0.128/0.008 ms
```

3. Rule for allowing inside hosts to ping outside hosts, done by accepting requests on eth1:

```
root@28598c9f9891:/# iptables -A FORWARD -i eth1 -p icmp --icmp-type echo-request -j ACCEPT
root@28598c9f9891:/# iptables -t filter -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
```

4. All packets between internal and external hosts blocked, done by setting default rule to drop forward packets:

```
root@28598c9f9891:/# iptables -P FORWARD DROP
```

Works:

```
root@5fe863423438:/# telnet 192.168.60.5
Trying 192.168.60.5...
^C
```

```
root@335fb99ce82b:/# telnet 10.9.0.5
Trying 10.9.0.5...
^C
```

IP table results after all rules:

```
root@28598c9f9891:/# iptables -t filter -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
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
```

Task 2C:

1. Outside hosts can only access 192.168.60.5

```
root@28598c9f9891:/# iptables -A FORWARD -i eth0 -p tcp -d 192.168.60.5 --dport 23 -j ACCEPT
```

2. Internal host can accept other internal

```
root@28598c9f9891:/# iptables -A FORWARD -i eth1 -s 192.168.60.5 -p tcp --sport 23 -j ACCEPT
```

3. Internal cannot access external

```
root@28598c9f9891:/# iptables -P FORWARD DROP
```

4. Results of all:

```
root@28598c9f9891:/# iptables -t filter -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            192.168.60.5         tcp dpt:23
ACCEPT     tcp  --  192.168.60.5         0.0.0.0/0            tcp spt:23

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

Task 3: Connection Tracking and Stateful Firewall

Task 3A:

ICMP connection state is only kept for a couple of seconds (about 5).

```
root@28598c9f9891:/# ping 192.168.60.5 &> /dev/null &
[1] 94
root@28598c9f9891:/# conntrack -L
icmp     1 29 src=192.168.60.11 dst=192.168.60.5 type=8 code=0 id=94 src=192.168.60.5 dst=192.168.60.11 type=0 co
de=0 id=94 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@28598c9f9891:/# kill %1
root@28598c9f9891:/# conntrack -L
icmp     1 23 src=192.168.60.11 dst=192.168.60.5 type=8 code=0 id=94 src=192.168.60.5 dst=192.168.60.11 type=0 co
de=0 id=94 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
[1]+  Terminated              ping 192.168.60.5 &> /dev/null
root@28598c9f9891:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
```

Running UDP experiment:

```
root@335fb99ce82b:/# nc -lu 9090
hello
^C
```

```
root@5fe863423438:/# nc -u 192.168.60.5 9090
hello
```

```
root@28598c9f9891:/# conntrack -L
udp      17 2 src=10.9.0.5 dst=192.168.60.5 sport=35269 dport=9090 [UNREPLIED] src=192.168.60.5 dst=10.9.0.5 spor
t=9090 dport=35269 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@28598c9f9891:/# conntrack -L
conntrack v1.4.5 (conntrack-tools): 0 flow entries have been shown.
```

For TCP, waited about 120 sec.

```
root@5fe863423438:/# nc 192.168.60.5 9090
hi
```

```
root@335fb99ce82b:/# nc -l 9090
hi
```

```
root@28598c9f9891:/# conntrack -L
tcp      6 431989 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=56222 dport=9090 src=192.168.60.5 dst=10.9.0.5
sport=9090 dport=56222 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@28598c9f9891:/# conntrack -L
tcp      6 116 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=56222 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport
=9090 dport=56222 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@28598c9f9891:/# conntrack -L
tcp      6 114 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=56222 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport
=9090 dport=56222 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@28598c9f9891:/# conntrack -L
tcp      6 113 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=56222 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport
=9090 dport=56222 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@28598c9f9891:/# conntrack -L
tcp      6 112 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=56222 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport
=9090 dport=56222 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@28598c9f9891:/# conntrack -L
tcp      6 111 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=56222 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport
=9090 dport=56222 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@28598c9f9891:/# conntrack -L
tcp      6 110 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=56222 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport
=9090 dport=56222 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@28598c9f9891:/# conntrack -L
tcp      6 108 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=56222 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport
=9090 dport=56222 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@28598c9f9891:/# conntrack -L
tcp      6 107 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=56222 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport
=9090 dport=56222 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@28598c9f9891:/# conntrack -L
tcp      6 106 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=56222 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport
=9090 dport=56222 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@28598c9f9891:/# conntrack -L
tcp      6 105 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=56222 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport
=9090 dport=56222 [ASSURED] mark=0 use=1
```

Task 3B:

Task 4: Limiting Network Traffic

Without the second rule, there is no limiting of packets because all packets are still being accepted, thus 0% lost. With the second rule, the limit is set and we are now dropping those packets, thus a 67% loss when pinging 10.9.0.5.

```
root@28598c9f9891:/# iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute --limit-burst 5 -j ACCEPT
root@28598c9f9891:/# iptables -A FORWARD -s 10.9.0.5 -j DROP
```

```
root@5fe863423438:/# ping 192.168.60.5
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=63 time=0.160 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.101 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.090 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.104 ms
^C
--- 192.168.60.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3078ms
rtt min/avg/max/mdev = 0.090/0.113/0.160/0.027 ms
root@5fe863423438:/# ping 192.168.60.5
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=63 time=0.095 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.113 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.121 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.116 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.111 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.128 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.113 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.121 ms
64 bytes from 192.168.60.5: icmp_seq=25 ttl=63 time=0.146 ms
64 bytes from 192.168.60.5: icmp_seq=31 ttl=63 time=0.124 ms
^C
--- 192.168.60.5 ping statistics ---
31 packets transmitted, 10 received, 67.7419% packet loss, time 30703ms
rtt min/avg/max/mdev = 0.095/0.118/0.146/0.012 ms
```

Task 5:

I implemented the following rules to send a packet to different destinations every nth packet. The first packet was sent to host1, second to host2, and third to host3.

```
root@28598c9f9891:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 3 --packet
 0 -j DNAT --to-destination 192.168.60.5:8080
root@28598c9f9891:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 2 --packet
 0 -j DNAT --to-destination 192.168.60.6:8080
root@28598c9f9891:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 1 --packet
 0 -j DNAT --to-destination 192.168.60.7:8080
root@28598c9f9891:/# 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
```

The echos from hostA:

```
root@5fe863423438:/# echo hello | nc -u 10.9.0.11 8080
^C
root@5fe863423438:/# echo hello2 | nc -u 10.9.0.11 8080
^C
root@5fe863423438:/# echo hello3 | nc -u 10.9.0.11 8080
^C
```

Host1 results:

```
root@335fb99ce82b:/# nc -luk 8080
hello
hello
```

Host2 results:

```
root@246380d2481e:/# nc -luk 8080
hello2
```

Host 3 results:

```
root@848960a47faa:/# nc -luk 8080
hello3
```

Write-Up

I learned that firewalls are much easier to implement than expected. Ultimately, any tool that can block a packet, we looked at as a firewall, whether that be an LKM or directly through iptables. Specifically, with the iptable rules, there were many components to a rule that had to be specified, but the table provided to us and the context behind whether we were looking to block or accept input, output, or forwarding guided our tests. Iptables is especially powerful because its rules are explicitly laid out and it is built into Linux for ease of use.