Maya Humston
Professor Mo Satt
CS-UY 3933 Network Security
19 December 2023

Lab 5 – Firewalls

TASK 1: Implementing a Simple Firewall

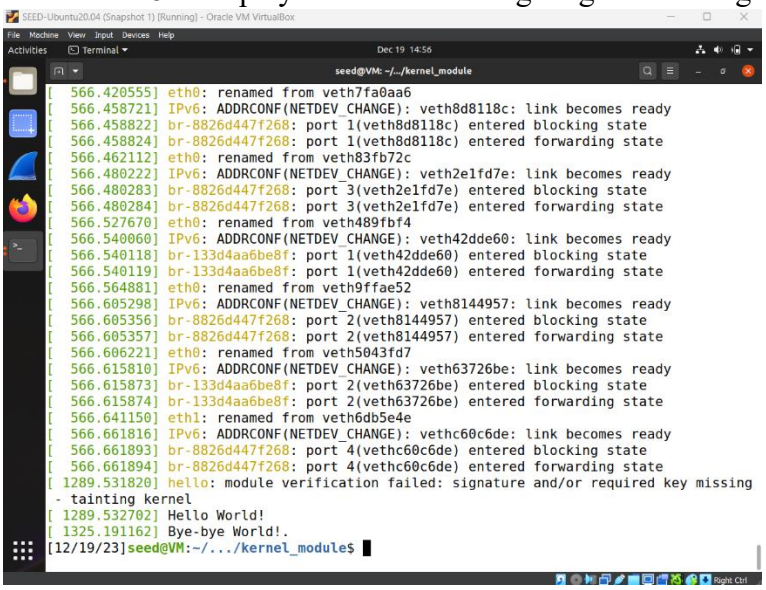
- Implement a simple kernel module:
  - Cd into the given kernel_module example folder. All code is already given.
  - Compile the program with "make"

```
[12/19/23]seed@VM:~/.../kernel_module$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Desktop/Labsetup/Files/kernel_
module modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Desktop/Labsetup/Files/kernel_module/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Desktop/Labsetup/Files/kernel_module/hello.mod.o
  LD [M]  /home/seed/Desktop/Labsetup/Files/kernel_module/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[12/19/23]seed@VM:~/.../kernel_module$
```

  - Mount the kernel module to the kernel, confirm that the module is there by listing all kernel modules with lsmod, and remove/unmount the module

```
[12/19/23]seed@VM:~/.../kernel_module$ sudo insmod hello.ko
[12/19/23]seed@VM:~/.../kernel_module$ lsmod | grep hello
hello                  16384  0
[12/19/23]seed@VM:~/.../kernel_module$ sudo rnmod hello
sudo: rnmod: command not found
[12/19/23]seed@VM:~/.../kernel_module$ sudo rmmod hello
[12/19/23]seed@VM:~/.../kernel_module$
```

  - Display the kernel message log with dmesg



- Implement a simple firewall using net filter:
    - Ran the sample code already given in seedFilter.c (compile, mount, and attempt a UDP packet query) – since the packet did not return the content/information at www.example.com, it was blocked by the firewall.

```
[12/19/23]seed@VM:~/.../kernel_module$ cd ../packet_filter
[12/19/23]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/5.4.0-54-generic/build M=/home/seed/Desktop/Labsetup/Files/packet_
filter modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-54-generic'
  CC [M]  /home/seed/Desktop/Labsetup/Files/packet_filter/seedFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/seed/Desktop/Labsetup/Files/packet_filter/seedFilter.mod.o
  LD [M]  /home/seed/Desktop/Labsetup/Files/packet_filter/seedFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-54-generic'
[12/19/23]seed@VM:~/.../packet_filter$ sudo insmod seedFilter.ko
[12/19/23]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

[12/19/23]seed@VM:~/.../packet_filter$
```

    - Changed the code in seedFilter.c
        - Added hooks to global vars

```
10
11
12 static struct nf_hook_ops hook1a, hook1b, hook1c, hook1d, hook1e, hook2;
13
```

        - Added hooks for printInfo

```
76
77    // HOOKS FOR PRINTINFO
78
79    hook1a.hook = printInfo;
80    hook1a.hooknum = NF_INET_PRE_ROUTING;
81    hook1a.pf = PF_INET;
82    hook1a.priority = NF_IP_PRI_FIRST;
83    nf_register_net_hook(&init_net, &hook1a);
84
85    hook1b.hook = printInfo;
86    hook1b.hooknum = NF_INET_LOCAL_IN;
87    hook1b.pf = PF_INET;
88    hook1b.priority = NF_IP_PRI_FIRST;
89    nf_register_net_hook(&init_net, &hook1b);
90
91    hook1c.hook = printInfo;
92    hook1c.hooknum = NF_INET_FORWARD;
93    hook1c.pf = PF_INET;
94    hook1c.priority = NF_IP_PRI_FIRST;
95    nf_register_net_hook(&init_net, &hook1c);
96
```

```
    hook1d.hook = printInfo;
    hook1d.hooknum = NF_INET_LOCAL_OUT;
    hook1d.pf = PF_INET;
    hook1d.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook1d);

    hook1e.hook = printInfo;
    hook1e.hooknum = NF_INET_POST_ROUTING;
    hook1e.pf = PF_INET;
    hook1e.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook1e);
```

- Deregistered hooks upon exit

```
void removeFilter(void) {
    printk(KERN_INFO "The filters are being removed.\n");
    nf_unregister_net_hook(&init_net, &hook1a);
    nf_unregister_net_hook(&init_net, &hook1b);
    nf_unregister_net_hook(&init_net, &hook1c);
    nf_unregister_net_hook(&init_net, &hook1d);
    nf_unregister_net_hook(&init_net, &hook1e);

    nf_unregister_net_hook(&init_net, &hook2);
}

module_init(registerFilter);
module_exit(removeFilter);
```

- Compiled the code, mounted the kernel module, attempted to send UDP packet to test firewall

```
[ 9075.450819] Hello World!
[ 9076.353329] Bye-bye World!.
[ 9118.142540] Registering filters.
[ 9146.213076] *** LOCAL_OUT
[ 9146.213081]     127.0.0.1  --> 127.0.0.1 (UDP)
[ 9146.213104] *** POST_ROUTING
[ 9146.213106]     127.0.0.1  --> 127.0.0.1 (UDP)
[ 9146.213170] *** PRE_ROUTING
[ 9146.213172]     127.0.0.1  --> 127.0.0.1 (UDP)
[ 9146.213175] *** LOCAL_IN
[ 9146.213176]     127.0.0.1  --> 127.0.0.1 (UDP)
[ 9146.214089] *** LOCAL_OUT
[ 9146.214091]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 9146.214100] *** Dropping 8.8.8.8 (UDP), port 53
[ 9151.215832] *** LOCAL_OUT
[ 9151.215840]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 9151.215995] *** Dropping 8.8.8.8 (UDP), port 53
[ 9156.221467] *** LOCAL_OUT
[ 9156.221473]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 9156.221500] *** Dropping 8.8.8.8 (UDP), port 53
[12/19/23]seed@VM:~/.../packet_filter$
```

- It appears that packets go through local_out, post_routing, pre_routing, and local_in in that order. From the kernel logs, the UDP packet goes through the local_out hook but gets caught by the post_routing hook. This is confirmed by checking the c code provided.

- Make two more hooks

- Telnet: to block all telnet requests into 10.9.0.1, I identified packets using TCP on port 20 with destination IP = 10.9.0.1. I hooked this function to LOCAL_IN, so that all incoming packets would be filtered. The code for this function is as follows:

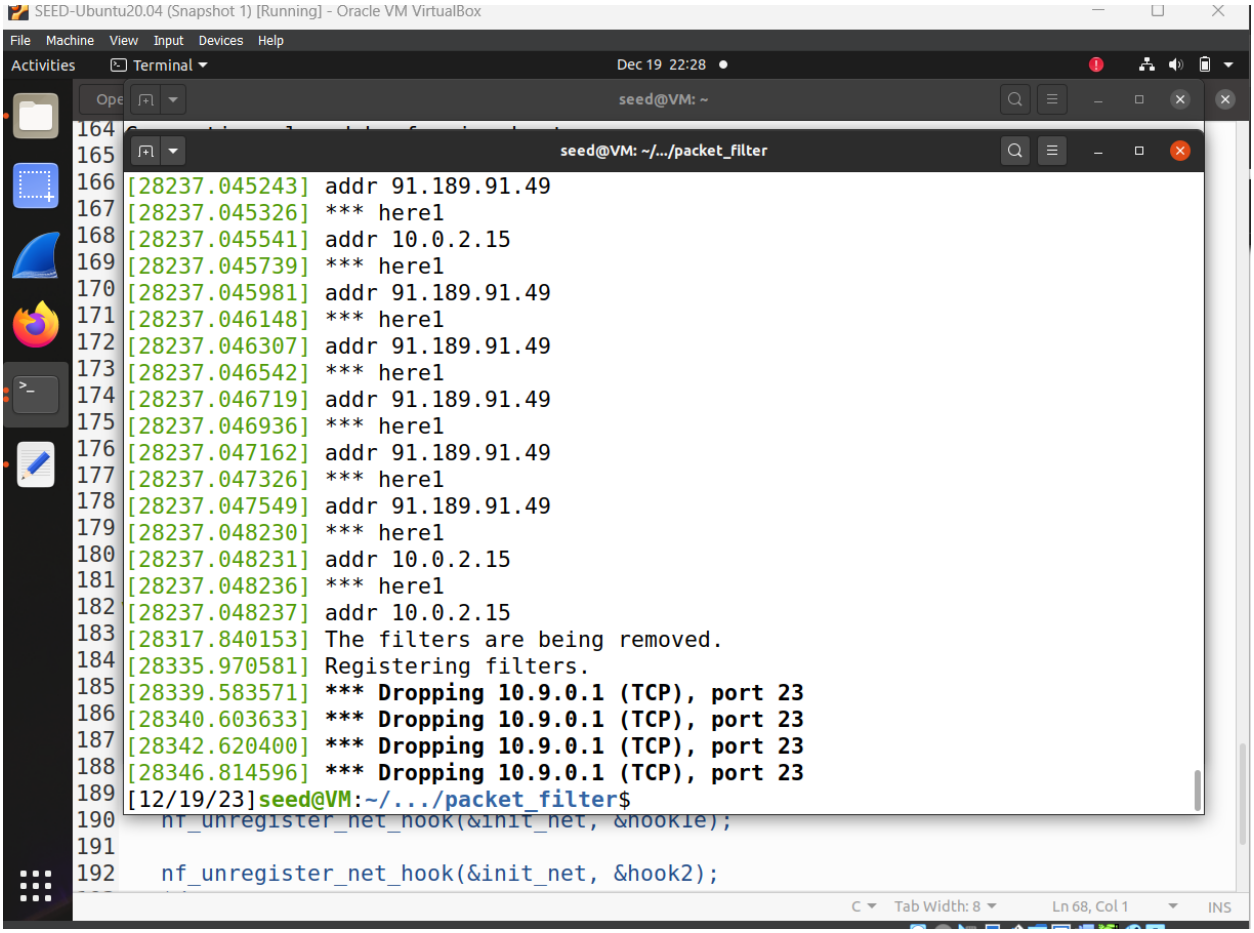
```
unsigned int blockTelnet(void *priv, struct sk_buff *skb,
                         const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct tcphdr *tcph;

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

```
   telnet_hook3.hook = blockTelnet;
   telnet_hook3.hooknum = NF_INET_LOCAL_IN;
   telnet_hook3.pf = PF_INET;
   telnet_hook3.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &telnet_hook3);
```

**Not shown: I added telnet_hook3 to the global variables and deregistered the hook upon kernel module unmount.

Here's the attempt to telnet into 10.9.0.1 from 10.9.0.5, and the corresponding kernel logs:

```
root@4683e1dc93df:/# telnet 10.9.0.1
Trying 10.9.0.1...
^C
root@4683e1dc93df:/# telnet 10.9.0.1
Trying 10.9.0.1...
^C
root@4683e1dc93df:/#
```



- Ping: ping is an ICMP packet, not TCP or UDP. You can identify ICMP packets with icmp.h, using struct icmphdr and code = ICMP_ECHO (the macro for 8) but this was not apparently the preferred method for this firewall implementation. Instead, I filtered all packets using the ICMP protocol that went to 10.9.0.1. I suppose this filters ping and all related queries as well, instead of just echo requests. The code for the function is as follows: (note that I reused code, so please ignore that variable names contain “telnet”, as the code is correct regardless. Additionally, I left the “port” variable so it will still get printed, but this does not affect anything and the code is still correct.)

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

- The ping attempt and the kernel logs:

```
^C
root@4683e1dc93df:/# ping 10.9.0.1
^[[6~PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
```

```
seed@VM: ~/.../packet_filter
[29501.411736] *** Dropping 10.9.0.1 (ICMP), port 23
[29502.432816] *** Dropping 10.9.0.1 (ICMP), port 23
[29503.461446] *** Dropping 10.9.0.1 (ICMP), port 23
[29504.482197] *** Dropping 10.9.0.1 (ICMP), port 23
[29505.506374] *** Dropping 10.9.0.1 (ICMP), port 23
[29506.531693] *** Dropping 10.9.0.1 (ICMP), port 23
[29507.555358] *** Dropping 10.9.0.1 (ICMP), port 23
[29508.579712] *** Dropping 10.9.0.1 (ICMP), port 23
[29509.604292] *** Dropping 10.9.0.1 (ICMP), port 23
[29510.629373] *** Dropping 10.9.0.1 (ICMP), port 23
[29511.652817] *** Dropping 10.9.0.1 (ICMP), port 23
[29512.678140] *** Dropping 10.9.0.1 (ICMP), port 23
[29513.702193] *** Dropping 10.9.0.1 (ICMP), port 23
[29514.727848] *** Dropping 10.9.0.1 (ICMP), port 23
[29515.751824] *** Dropping 10.9.0.1 (ICMP), port 23
[29516.775845] *** Dropping 10.9.0.1 (ICMP), port 23
[29517.801799] *** Dropping 10.9.0.1 (ICMP), port 23
[29518.824429] *** Dropping 10.9.0.1 (ICMP), port 23
[29519.848612] *** Dropping 10.9.0.1 (ICMP), port 23
[29520.874812] *** Dropping 10.9.0.1 (ICMP), port 23
[29521.901743] *** Dropping 10.9.0.1 (ICMP), port 23
[29522.923230] *** Dropping 10.9.0.1 (ICMP), port 23
[29523.946878] *** Dropping 10.9.0.1 (ICMP), port 23
[12/19/23]seed@VM:~/.../packet_filter$
```

TASK 2: Experimenting with Stateless Firewall Rules

- Protecting the router:
    - Run the commands as shown; Pings are allowed through manually, while all other incoming and outgoing packets are dropped by default.

```
[12/20/23]seed@VM:~$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j
ACCEPT
[12/20/23]seed@VM:~$ sudo iptables -A OUTPUT -p icmp --icmp-type echo-reply -j A
CCEPT
[12/20/23]seed@VM:~$ sudo iptables -P OUTPUT DROP
[12/20/23]seed@VM:~$ sudo iptables -P INPUT DROP
[12/20/23]seed@VM:~$ sudo 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 DROP)
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

Chain DOCKER (0 references)
target     prot opt source               destination

Chain DOCKER-ISOLATION-STAGE-1 (0 references)
target     prot opt source               destination

Chain DOCKER-ISOLATION-STAGE-2 (0 references)
target     prot opt source               destination

Chain DOCKER-USER (0 references)
target     prot opt source               destination
[12/20/23]seed@VM:~$
```

```
[12/20/23]seed@VM:~$ docker ps
CONTAINER ID        IMAGE                                  COMMAND
 CREATED             STATUS              PORTS               NAMES
9152baf789d7        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 41 minutes ago      Up 41 minutes                           host1-192.168.60.5
046d7f64e2a8        seed-router-image                    "bash -c ' ip route …"
 41 minutes ago      Up About a minute                       seed-router
59409a19e47d        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 41 minutes ago      Up 52 seconds                           hostA-10.9.0.5
3557183f89d1        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 41 minutes ago      Up 41 minutes                           host2-192.168.60.6
20629e2348f6        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 41 minutes ago      Up 41 minutes                           host3-192.168.60.7
[12/20/23]seed@VM:~$ docksh 59
root@59409a19e47d:/# telnet 10.9.0.1
Trying 10.9.0.1...
^C
root@59409a19e47d:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
64 bytes from 10.9.0.1: icmp_seq=1 ttl=64 time=0.103 ms
64 bytes from 10.9.0.1: icmp_seq=2 ttl=64 time=0.124 ms
64 bytes from 10.9.0.1: icmp_seq=3 ttl=64 time=0.072 ms
64 bytes from 10.9.0.1: icmp_seq=4 ttl=64 time=0.147 ms
64 bytes from 10.9.0.1: icmp_seq=5 ttl=64 time=0.096 ms
^C
--- 10.9.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4085ms
rtt min/avg/max/mdev = 0.072/0.108/0.147/0.025 ms
root@59409a19e47d:/#
```

- Protecting the internal network:
  - First, enter the docker container shell for the router and list the network interfaces

```
[12/20/23]seed@VM:~$ docker ps
CONTAINER ID        IMAGE                                COMMAND
 CREATED             STATUS              PORTS               NAMES
d31295963673        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 44 seconds ago      Up 41 seconds                           hostA-10.9.0.5
daee5dc0840d        seed-router-image                    "bash -c ' ip route …"
 44 seconds ago      Up 41 seconds                           seed-router
309ddb1b837a        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 44 seconds ago      Up 41 seconds                           host3-192.168.60.7
7093f826d829        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 44 seconds ago      Up 41 seconds                           host2-192.168.60.6
6e36aa4be9cb        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 44 seconds ago      Up 41 seconds                           host1-192.168.60.5
[12/20/23]seed@VM:~$ docksh da
root@daee5dc0840d:/# 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
42: eth0@if43: <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
46: eth1@if47: <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
```

- Then set the iptables rules. Drop all forward packets coming from the outside interface eth0 with icmp-type echo-request; Accept all forward packets coming from the outside eth0 with echo-reply; Accept forward packets from the inside eth1 with echo-request; Drop all other forward packets.

```
root@daee5dc0840d:/# iptables -P FORWARD -i eth0 -p icmp --icmp-type echo-reques
t -j DROP
iptables v1.8.4 (legacy): -P requires a chain and a policy
Try `iptables -h' or 'iptables --help' for more information.
root@daee5dc0840d:/# iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-reques
t -j DROP
root@daee5dc0840d:/# iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-reply
-j ACCEPT
root@daee5dc0840d:/# iptables -A FORWARD -i eth1 -p icmp --icmp-type echo-reques
t -j ACCEPT
root@daee5dc0840d:/# iptables -P DROP
iptables v1.8.4 (legacy): -P requires a chain and a policy
Try `iptables -h' or 'iptables --help' for more information.
root@daee5dc0840d:/# iptables -P FORWARD DROP
root@daee5dc0840d:/# iptables -t filter -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain FORWARD (policy DROP 3 packets, 180 bytes)
 pkts bytes target     prot opt in     out     source               destination

   11   924 DROP       icmp --  eth0   *       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
    0     0 ACCEPT     icmp --  eth1   *       0.0.0.0/0            0.0.0.0/0
         icmptype 8

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
```

- From the docker shell of the attacker, attempt to ping the internal network (fail), ping the router (success), and telnet into an internal host (fail).

```
[12/20/23]seed@VM:~$ docksh d3
root@d31295963673:/# telnet 192.168.60.5
Trying 192.168.60.5...
^C
root@d31295963673:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
64 bytes from 10.9.0.1: icmp_seq=1 ttl=64 time=0.067 ms
64 bytes from 10.9.0.1: icmp_seq=2 ttl=64 time=0.135 ms
^C
--- 10.9.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1017ms
rtt min/avg/max/mdev = 0.067/0.101/0.135/0.034 ms
root@d31295963673:/# 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 5110ms

root@d31295963673:/# exit
```

- From the docksh of an internal host, attempt to ping the external network (success) and telnet into an external host (fail).

```
[12/20/23]seed@VM:~$ docksh 30
root@309ddb1b837a:/# 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=0.204 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.173 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.067 ms
^C
--- 10.9.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2056ms
rtt min/avg/max/mdev = 0.067/0.148/0.204/0.058 ms
root@309ddb1b837a:/# telnet 10.9.0.5
Trying 10.9.0.5...
^C
root@309ddb1b837a:/#
```

- Protecting internal servers:
  - Set the rules on the router's iptables:
  - Accept forward packets from the outside using TCP with destination IP = 192.168.60.5 and destination port = 23
  - Accept forward packets from the inside using TCP with source IP = 192.168.60.5 and source port = 23
  - Accept forward packets from the inside to the inside using TCP with destination port = 23
  - Accept forward packets from the inside to the inside using TCP with source port = 23

```
root@daee5dc0840d:/# iptables -A FORWARD -i eth0 -p tcp -d 192.168.60.5 --dport
23 -j ACCEPT
root@daee5dc0840d:/# iptables -A FORWARD -i eth1 -p tcp -s 192.168.60.5 --sport
23 -j ACCEPT
root@daee5dc0840d:/# iptables -A FORWARD -i eth1 -o eth1 -p tcp -d --dport 23 -j
 ACCEPT
Bad argument `23'
Try `iptables -h' or 'iptables --help' for more information.
root@daee5dc0840d:/# iptables -A FORWARD -i eth1 -o eth1 -p tcp --dport 23 -j AC
CEPT
root@daee5dc0840d:/# iptables -P FORWARD DROP
root@daee5dc0840d:/# iptables -A FORWARD -i eth1 -o eth1 -p tcp --sport 23 -j AC
CEPT
root@daee5dc0840d:/#
```

  - Test telnet from an external host. telnet into 192.168.60.5 (success) and 192.168.60.6 (fail)

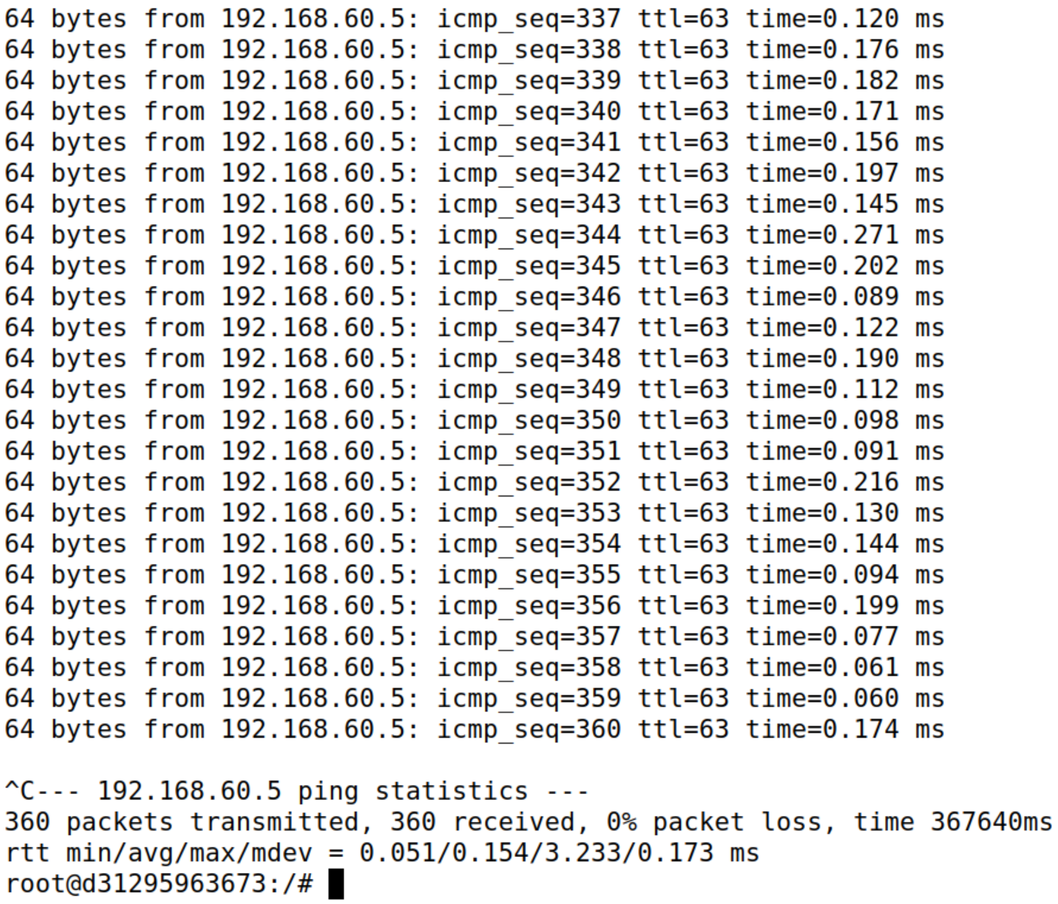
```
[12/20/23]seed@VM:~$ docksh d3
root@d31295963673:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
^C^C^CUbuntu 20.04.1 LTS
6e36aa4be9cb login: ^CConnection closed by foreign host.
root@d31295963673:/# telnet 192.168.60.6
Trying 192.168.60.6...
^C
root@d31295963673:/#
```

- Test telnet from an internal host. telnet into 10.9.0.5 (fail) 192.168.60.5 (success) and 192.168.60.6 (success)

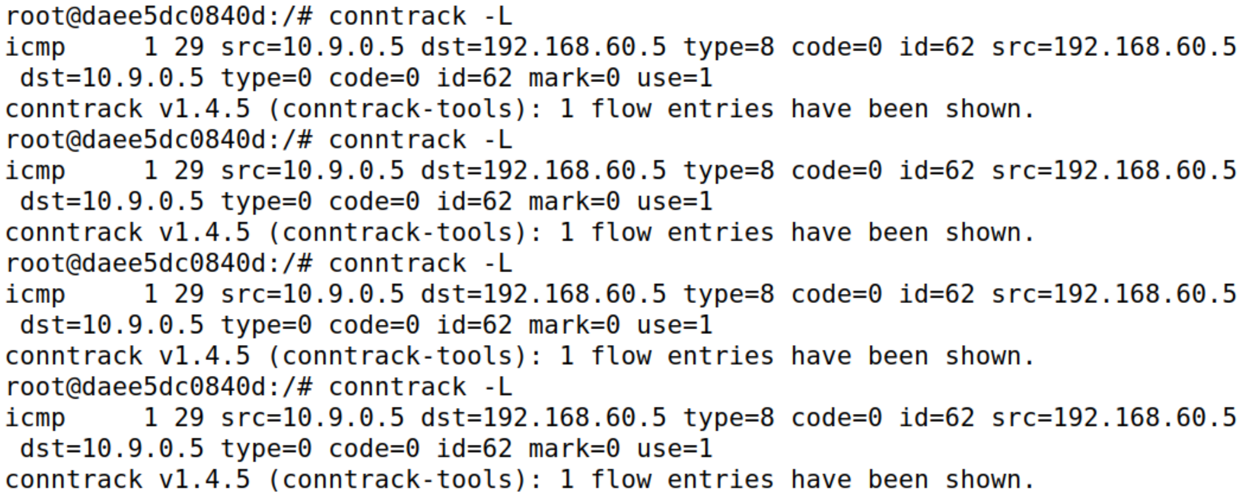
```
root@309ddb1b837a:/# telnet 10.9.0.5
Trying 10.9.0.5...
^C
root@309ddb1b837a:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6e36aa4be9cb login: ^CConnection closed by foreign host.
root@309ddb1b837a:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
7093f826d829 login: ^CConnection closed by foreign host.
root@309ddb1b837a:/# exit
```

TASK 3: Connection Tracking and Stateful Firewall

- Experiment with the connection tracking:
  - Ping: I set up a continuous ping in the background from the external host to the internal host (first pic), and ran a few conntrack -L commands on the seed-router (second pic). This was hard to interpret because the table doesn't have headers and doesn't seem to have a verbose version, but this source https://www.frozentux.net/iptables-tutorial/chunkyhtml/x1582.html says the default connection timeout is 30 second for ICMP packets, which appears to be confirmed by the "29" in my conntrack entries.



```
64 bytes from 192.168.60.5: icmp_seq=337 ttl=63 time=0.120 ms
64 bytes from 192.168.60.5: icmp_seq=338 ttl=63 time=0.176 ms
64 bytes from 192.168.60.5: icmp_seq=339 ttl=63 time=0.182 ms
64 bytes from 192.168.60.5: icmp_seq=340 ttl=63 time=0.171 ms
64 bytes from 192.168.60.5: icmp_seq=341 ttl=63 time=0.156 ms
64 bytes from 192.168.60.5: icmp_seq=342 ttl=63 time=0.197 ms
64 bytes from 192.168.60.5: icmp_seq=343 ttl=63 time=0.145 ms
64 bytes from 192.168.60.5: icmp_seq=344 ttl=63 time=0.271 ms
64 bytes from 192.168.60.5: icmp_seq=345 ttl=63 time=0.202 ms
64 bytes from 192.168.60.5: icmp_seq=346 ttl=63 time=0.089 ms
64 bytes from 192.168.60.5: icmp_seq=347 ttl=63 time=0.122 ms
64 bytes from 192.168.60.5: icmp_seq=348 ttl=63 time=0.190 ms
64 bytes from 192.168.60.5: icmp_seq=349 ttl=63 time=0.112 ms
64 bytes from 192.168.60.5: icmp_seq=350 ttl=63 time=0.098 ms
64 bytes from 192.168.60.5: icmp_seq=351 ttl=63 time=0.091 ms
64 bytes from 192.168.60.5: icmp_seq=352 ttl=63 time=0.216 ms
64 bytes from 192.168.60.5: icmp_seq=353 ttl=63 time=0.130 ms
64 bytes from 192.168.60.5: icmp_seq=354 ttl=63 time=0.144 ms
64 bytes from 192.168.60.5: icmp_seq=355 ttl=63 time=0.094 ms
64 bytes from 192.168.60.5: icmp_seq=356 ttl=63 time=0.199 ms
64 bytes from 192.168.60.5: icmp_seq=357 ttl=63 time=0.077 ms
64 bytes from 192.168.60.5: icmp_seq=358 ttl=63 time=0.061 ms
64 bytes from 192.168.60.5: icmp_seq=359 ttl=63 time=0.060 ms
64 bytes from 192.168.60.5: icmp_seq=360 ttl=63 time=0.174 ms

^C--- 192.168.60.5 ping statistics ---
360 packets transmitted, 360 received, 0% packet loss, time 367640ms
rtt min/avg/max/mdev = 0.051/0.154/3.233/0.173 ms
root@d31295963673:/# 
```



```
root@daee5dc0840d:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=62 src=192.168.60.5
 dst=10.9.0.5 type=0 code=0 id=62 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@daee5dc0840d:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=62 src=192.168.60.5
 dst=10.9.0.5 type=0 code=0 id=62 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@daee5dc0840d:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=62 src=192.168.60.5
 dst=10.9.0.5 type=0 code=0 id=62 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@daee5dc0840d:/# conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=62 src=192.168.60.5
 dst=10.9.0.5 type=0 code=0 id=62 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
```

- UDP: Set up a UDP server on 192.168.60.5 (pic 1), then connect to it and send a message from 10.9.0.5 (pic 2), then QUICKLY switch to the router shell and run conntrack -L (pic 3). The first time I showed the conntrack it showed 21 remaining seconds, and the second attempt I was slightly faster and showed 26 remaining seconds, so I am assuming a 30 second connection from this information.

```
[12/20/23]seed@VM:~$ docker ps
CONTAINER ID        IMAGE                                COMMAND
 CREATED             STATUS              PORTS               NAMES
d31295963673        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 About an hour ago   Up About an hour                      hostA-10.9.0.5
daee5dc0840d        seed-router-image                    "bash -c ' ip route …"
 About an hour ago   Up About an hour                      seed-router
309ddb1b837a        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 About an hour ago   Up About an hour                      host3-192.168.60.7
7093f826d829        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 About an hour ago   Up About an hour                      host2-192.168.60.6
6e36aa4be9cb        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 About an hour ago   Up About an hour                      host1-192.168.60.5
[12/20/23]seed@VM:~$ docksh 6e
root@6e36aa4be9cb:/# nc -lu 9090
hello
hello again
one more
```

```
root@d31295963673:/#
root@d31295963673:/# nc -u 192.168.60.5 9090
hello
hello again
one more
```

```
root@daee5dc0840d:/# conntrack -L
udp      17 26 src=10.9.0.5 dst=192.168.60.5 sport=40568 dport=9090 [UNREPLIED]
src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=40568 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@daee5dc0840d:/#
```

- TCP: start a server for tcp connection on host 192.168.60.5 (pic1), connect to it on host 10.9.0.5 and send a tcp message (pic 2), then QUICKLY check the conntrack table on the router (pic 3). As shown, the remaining time for this tcp connection is 431996 seconds, so I guess the connection state timeout is 432,000 seconds.

```
^C
root@6e36aa4be9cb:/# nc -l 9090
tcp connect
```

```
^C
root@d31295963673:/# nc 192.168.60.5 9090
tcp connect
```

```
root@daee5dc0840d:/# conntrack -L
tcp      6 431996 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=51738 dport=90
90 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=51738 [ASSURED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
root@daee5dc0840d:/#
```

- Setting up a stateful firewall: configure the iptables:
    - Accept tcp packets going internal -> internal
    - Accept tcp packets going from the outside to 192.168.60.5 port 23 with new connection state/syn bit
    - Accept tcp packets going from internal to external with new connection state
    - Accept tcp packets with established/related connection status
    - Drop other forwarding packets

```
root@daee5dc0840d:/# iptables -A FORWARD -i eth0 -p tcp -d 192.168.60.5 --dport
23 -j ACCEPT
root@daee5dc0840d:/# iptables -A FORWARD -i eth0 -p tcp -d 192.168.60.5 --dport
23 -j ACCEPT
root@daee5dc0840d:/# iptables -A FORWARD -i eth1 -p tcp --syn -m conntrack --cst
ate NEW -j ACCEPT
iptables v1.8.4 (legacy): unknown option "--cstate"
Try `iptables -h' or 'iptables --help' for more information.
root@daee5dc0840d:/# iptables -A FORWARD -i eth1 -p tcp --syn -m conntrack --cts
tate NEW -j ACCEPT
root@daee5dc0840d:/# iptables -A FORWARD -p tcp -m conntrack --ctstate RELATED,E
STABLISHED -j ACCEPT
root@daee5dc0840d:/# iptables -P FORWARD DROP
root@daee5dc0840d:/# iptables -A FORWARD -i eth1 -o eth1 -p tcp -j ACCEPT
```

    - Try telnet from external to 192.168.60.5 (success), then from external to 192.168.60.6 (fail)

```
root@d31295963673:/# telnet 192.168.60.6
Trying 192.168.60.6...
^C
root@d31295963673:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6e36aa4be9cb login: ^CConnection closed by foreign host.
root@d31295963673:/# exit
```

    - Try telnet from 192.168.60.7 to 192.168.60.6 (success) and from 192.168.60.7 to 10.9.0.5 (success)

```
[12/20/23]seed@VM:~$ docksh 30
root@309ddb1b837a:/# telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
7093f826d829 login: ^CConnection closed by foreign host.
root@309ddb1b837a:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
d31295963673 login: ^CConnection closed by foreign host.
root@309ddb1b837a:/#
```

TASK 4: Limiting Network Traffic

- Section 6:
  - Implement the new filter rule for the router iptables:

```
root@21e4c99bcb73:/# iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute
--limit-burst 5 -j ACCEPT
root@21e4c99bcb73:/#
```

  - Ping 192.168.60.5 from 10.9.0.5. As you can see, the rule we added does not successfully limit the number of incoming/outgoing ping packets.

```
 CREATED              STATUS              PORTS                NAMES
5a08a975b464        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 6 minutes ago       Up 6 minutes                         host3-192.168.60.7
21e4c99bcb73        seed-router-image                   "bash -c ' ip route …"
 6 minutes ago       Up 6 minutes                         seed-router
2fc6d50c2fc6        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 6 minutes ago       Up 6 minutes                         host2-192.168.60.6
b58f1e0572d3        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 6 minutes ago       Up 6 minutes                         hostA-10.9.0.5
4a5aaee2891f        handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"
 6 minutes ago       Up 6 minutes                         host1-192.168.60.5
[12/20/23]seed@VM:~$ docksh b5
root@b58f1e0572d3:/# 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=0.896 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.227 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.136 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.152 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.163 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.189 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.177 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.203 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.179 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.189 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.117 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.146 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.297 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.128 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.175 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.124 ms
```

  - Now add the second rule

```
root@21e4c99bcb73:/# iptables -A FORWARD -s 10.9.0.5 -j DROP
root@21e4c99bcb73:/#
```

  - Try pinging again. This time, after five pings, the rate of allowed pings slowed significantly. As you can see from the sequence numbers, only about 1 in 6 was getting through.

```
root@b58f1e0572d3:/# 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=0.212 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.064 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.103 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.103 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.106 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.097 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.133 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.176 ms
64 bytes from 192.168.60.5: icmp_seq=25 ttl=63 time=0.103 ms
64 bytes from 192.168.60.5: icmp_seq=31 ttl=63 time=0.761 ms
^C
--- 192.168.60.5 ping statistics ---
36 packets transmitted, 10 received, 72.2222% packet loss, time 35826ms
rtt min/avg/max/mdev = 0.064/0.185/0.761/0.195 ms
root@b58f1e0572d3:/#
```

TASK 5: Load Balancing

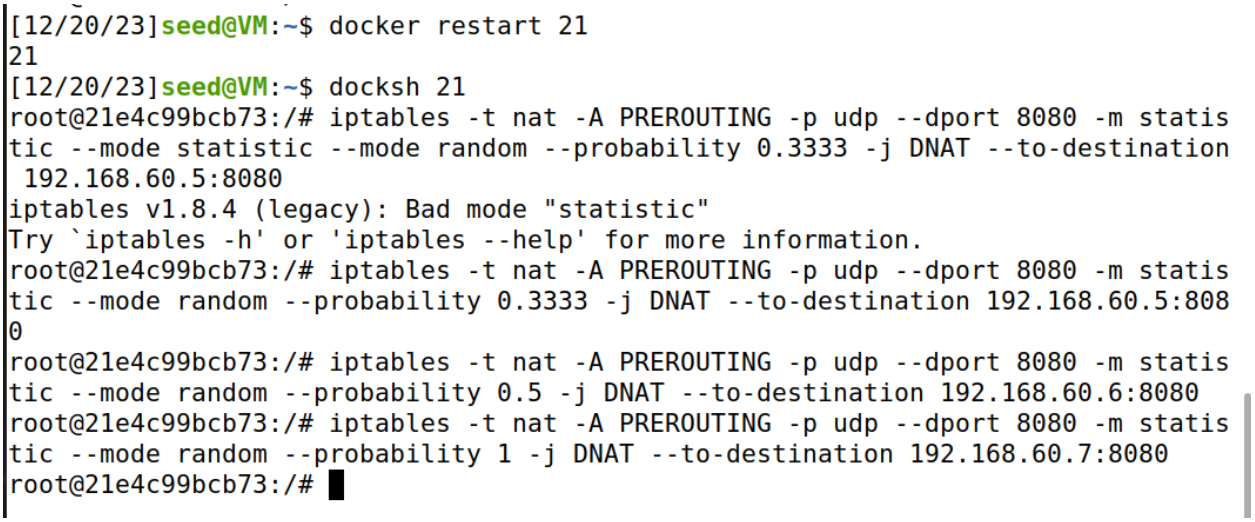
- Section 7:
  - Nth packet: add two rules to the given one.
    - Given: iptables … -every 3 … 192.168.60.5:8080
    - Add: iptables … -every 2 … 192.168.60.6:8080
    - Add: iptables … -every 1 … 192.168.60.7:8080
    - The logic is that the two packets that bypass the first rule will get caught by the second rule, and every other packet will get redirected by that rule and the other will get passed on to the last rule.

```
[12/20/23]seed@VM:~$ docksh 21
root@21e4c99bcb73:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode nth --every 3 --packet 0 -j DNAT --to-destination 192.168.60.5:8080
root@21e4c99bcb73:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode nth --every 2 --packet 0 -j DNAT --to-destination 192.168.60.6:8080
root@21e4c99bcb73:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode nth --every 1 --packet 0 -j DNAT --to-destination 192.168.60.7:8080
root@21e4c99bcb73:/#
```

    - Start the server on all three internal hosts, and then send a series of echo commands into the server from the outside host.

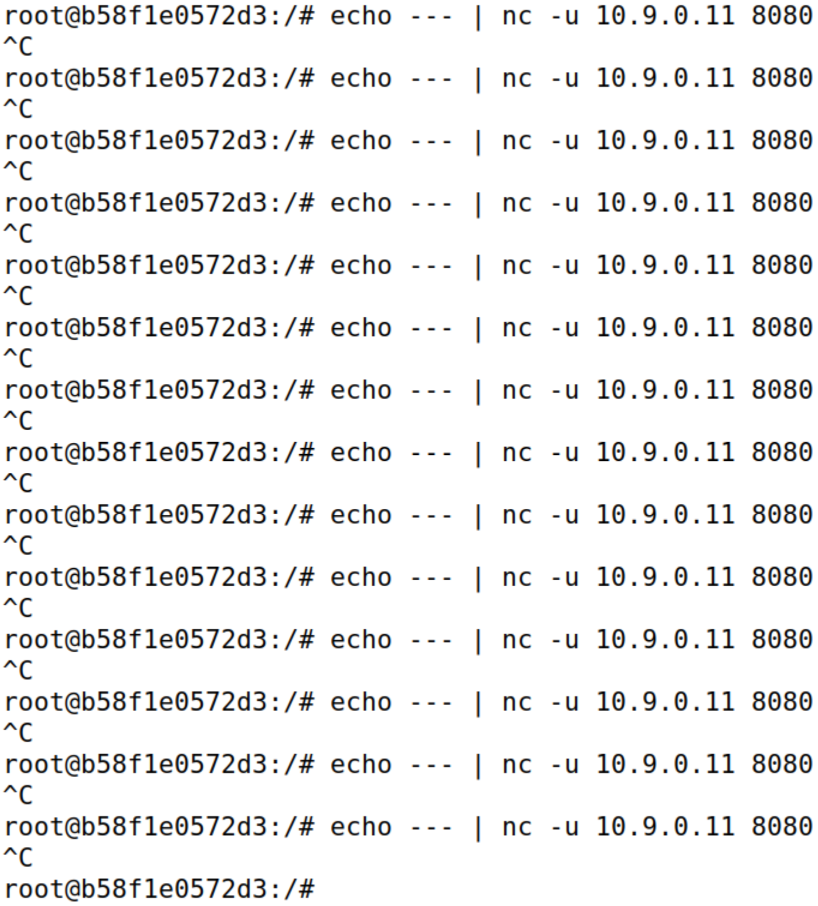
```
root@b58f1e0572d3:/# echo 1 | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo 2 | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo 3 | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo 4 | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo 5 | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo 6 | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo 7 | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo 8 | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo 9 | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo 10 | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/#
```

- Random probability:
- (referenced https://internetlifeforum.com/showthread.php?3007-How-to-install-kernel-IPTables-firewall-module-quot-statistic-quot to understand in what format probability values must be used)
- My logic for this one was very similar to the last. My theory is that the first rule in both cases has to take one of three packets. On this new one, this meant giving it a random probability value of 0.3333. The second rule must take half of the packets that get sent to it, so give it P 0.5. Lastly, the last IP address must be assigned to ALL remaining packets, so P 1.0.

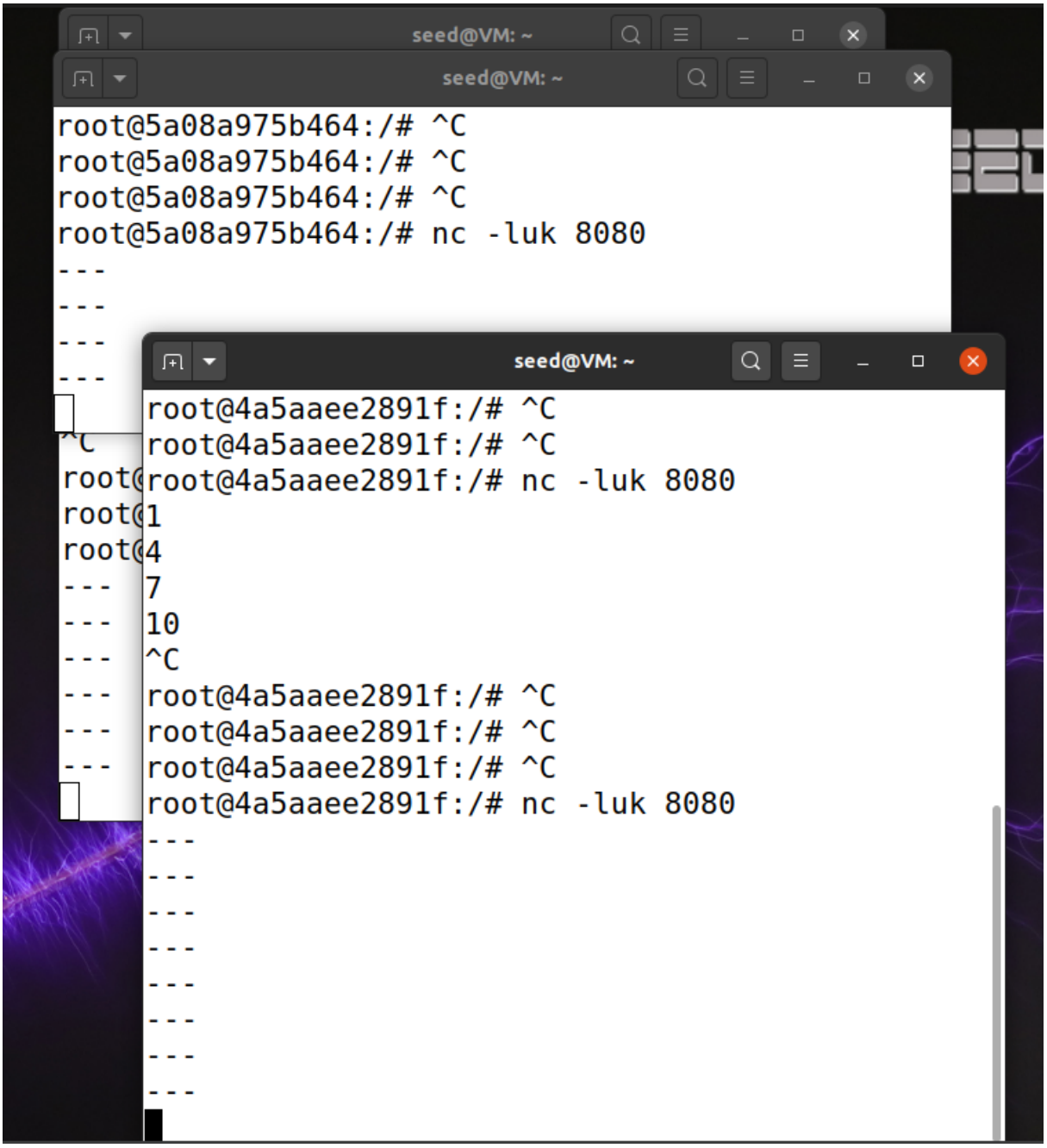


```
[12/20/23]seed@VM:~$ docker restart 21
21
[12/20/23]seed@VM:~$ docksh 21
root@21e4c99bcb73:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode statistic --mode random --probability 0.3333 -j DNAT --to-destination
 192.168.60.5:8080
iptables v1.8.4 (legacy): Bad mode "statistic"
Try `iptables -h' or 'iptables --help' for more information.
root@21e4c99bcb73:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode random --probability 0.3333 -j DNAT --to-destination 192.168.60.5:808
0
root@21e4c99bcb73:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode random --probability 0.5 -j DNAT --to-destination 192.168.60.6:8080
root@21e4c99bcb73:/# iptables -t nat -A PREROUTING -p udp --dport 8080 -m statis
tic --mode random --probability 1 -j DNAT --to-destination 192.168.60.7:8080
root@21e4c99bcb73:/# 
```

- Send echos to the internal network from the external host. I sent approx. 18 packets. Examine the distribution (slightly uneven, but I think my logic holds.)



```
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/# echo --- | nc -u 10.9.0.11 8080
^C
root@b58f1e0572d3:/#
```



<u>TASK 6: Write-Up</u>

This was neat way to explore firewalls and their capabilities. I noticed that when we set iptables rules the first few times, we set the default to DROP instead of FORWARD. I have to wonder – while this is probably more secure, I can't see how it is efficient or even possible with large firewalls or large systems. I think too many valid packets would get dropped. I suspect it's probably better practice to apply iptables drop rules based on known databases of attacks rather than a handful of discrete "accept" rules. However, for small or strict systems, this could be a

better practice. I liked how we used firewalls to redirect and change routing, not just to drop packets. This was an unfamiliar use case to me.

I found the load distribution very cool at the end. I was wondering when random probability would ever be useful compared to the nth if you know the number of hosts in the system, but I suppose random probability would be more useful for reactivity (taking a host out of the system frequently or turning hosts on and off depending on the work they're doing).

I had a lot of trouble with the beginning of this lab. The first few tasks took a very long, and they're not very well documented in the instructions or online in my opinion. This was pretty frustrating. For example, task 2 took me several hours where I kept restarting my router container and rerunning iptables rules. It turns out I was using the correct rules, but they only worked once I restarted all of my docker containers and the virtual machine entirely. This was obviously very frustrating and time consuming! The later tasks were much easier and more intuitive.