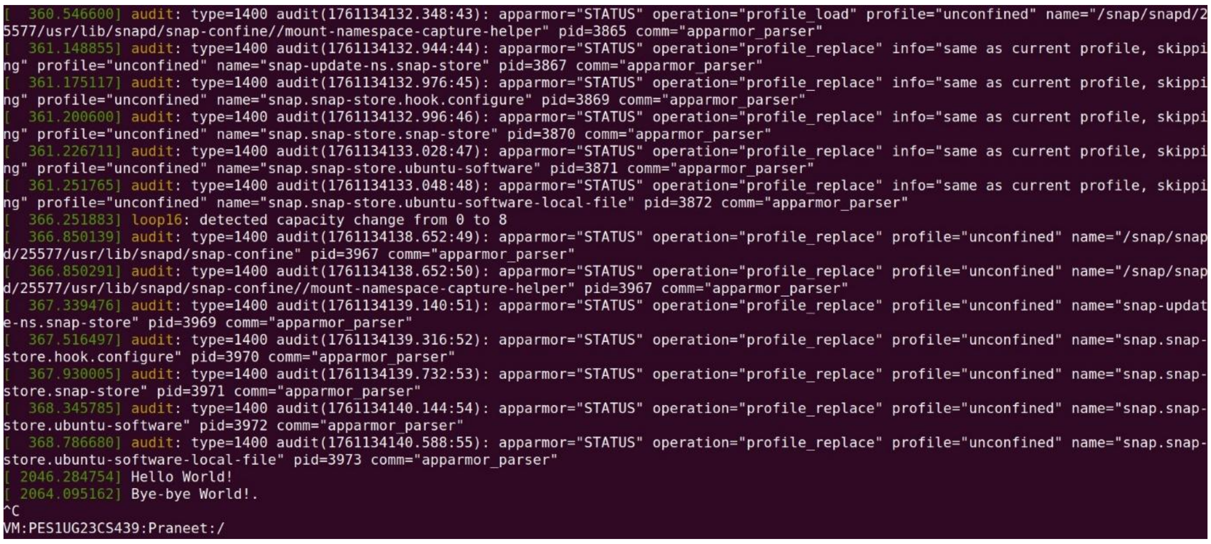
Name: Praneet T.H

Section: H

SRN:PES1UG23CS439

Task 1.A:

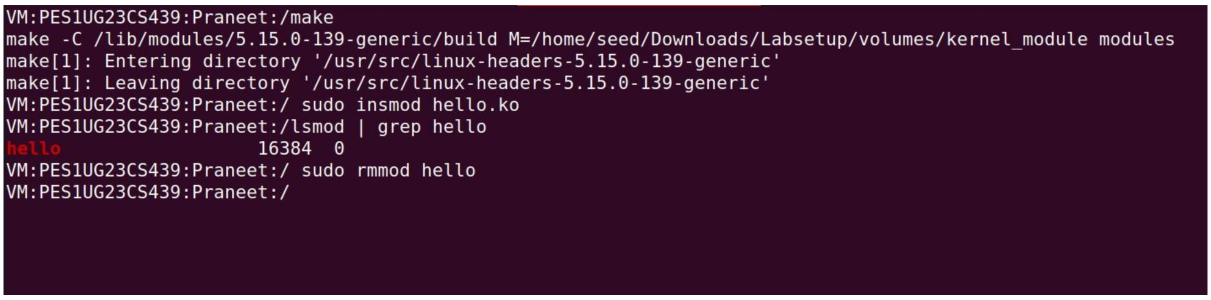
VM:



```
[  360.546600] audit: type=1400 audit(1761134132.348:43): apparmor="STATUS" operation="profile_load" profile="unconfined" name="/snap/snapd/2
5577/usr/lib/snapd/snap-confine//mount-namespace-capture-helper" pid=3865 comm="apparmor_parser"
[  361.148855] audit: type=1400 audit(1761134132.944:44): apparmor="STATUS" operation="profile_replace" info="same as current profile, skippi
ng" profile="unconfined" name="snap-update-ns.snap-store" pid=3867 comm="apparmor_parser"
[  361.175117] audit: type=1400 audit(1761134132.976:45): apparmor="STATUS" operation="profile_replace" info="same as current profile, skippi
ng" profile="unconfined" name="snap.snap-store.hook.configure" pid=3869 comm="apparmor_parser"
[  361.200600] audit: type=1400 audit(1761134132.996:46): apparmor="STATUS" operation="profile_replace" info="same as current profile, skippi
ng" profile="unconfined" name="snap.snap-store.snap-store" pid=3870 comm="apparmor_parser"
[  361.226711] audit: type=1400 audit(1761134133.028:47): apparmor="STATUS" operation="profile_replace" info="same as current profile, skippi
ng" profile="unconfined" name="snap.snap-store.ubuntu-software" pid=3871 comm="apparmor_parser"
[  361.251765] audit: type=1400 audit(1761134133.048:48): apparmor="STATUS" operation="profile_replace" info="same as current profile, skippi
ng" profile="unconfined" name="snap.snap-store.ubuntu-software-local-file" pid=3872 comm="apparmor_parser"
[  366.251883] loop16: detected capacity change from 0 to 8
[  366.850139] audit: type=1400 audit(1761134138.652:49): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="/snap/snap
d/25577/usr/lib/snapd/snap-confine" pid=3967 comm="apparmor_parser"
[  366.850291] audit: type=1400 audit(1761134138.652:50): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="/snap/snap
d/25577/usr/lib/snapd/snap-confine//mount-namespace-capture-helper" pid=3967 comm="apparmor_parser"
[  367.339476] audit: type=1400 audit(1761134139.140:51): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="snap-updat
e-ns.snap-store" pid=3969 comm="apparmor_parser"
[  367.516497] audit: type=1400 audit(1761134139.316:52): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="snap.snap-
store.hook.configure" pid=3970 comm="apparmor_parser"
[  367.930005] audit: type=1400 audit(1761134139.732:53): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="snap.snap-
store.snap-store" pid=3971 comm="apparmor_parser"
[  368.345785] audit: type=1400 audit(1761134140.144:54): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="snap.snap-
store.ubuntu-software" pid=3972 comm="apparmor_parser"
[  368.786680] audit: type=1400 audit(1761134140.588:55): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="snap.snap-
store.ubuntu-software-local-file" pid=3973 comm="apparmor_parser"
[ 2046.284754] Hello World!
[ 2064.095162] Bye-bye World!.
^C
VM:PES1UG23CS439:Praneet:/
```

Here we run monitoring using sudo dmesg -k -w and the respected output is achieved when the module is loaded and removed.

VM:



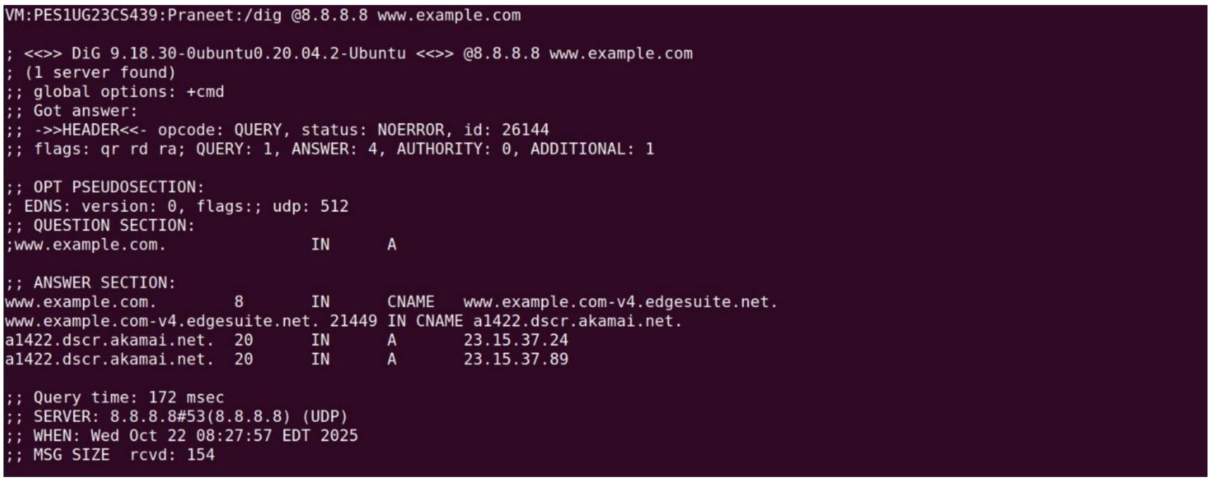
```
VM:PES1UG23CS439:Praneet:/make
make -C /lib/modules/5.15.0-139-generic/build M=/home/seed/Downloads/Labsetup/volumes/kernel_module modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-139-generic'
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-139-generic'
VM:PES1UG23CS439:Praneet:/ sudo insmod hello.ko
VM:PES1UG23CS439:Praneet:/lsmod | grep hello
hello                  16384  0
VM:PES1UG23CS439:Praneet:/ sudo rmmod hello
VM:PES1UG23CS439:Praneet:/
```

In this task, we created and tested a simple **Loadable Kernel Module (LKM)** to understand how modules can be dynamically added to or removed from the Linux kernel without rebooting. The provided hello.c program prints “Hello World!” when the module is loaded and “Bye-bye World!” when it is removed.

These messages are logged in the kernel log, which can be viewed using the dmesg command. To run the module, we open two terminal windows where one is to monitor the kernel log using sudo dmesg -k -w, and another to compile and manage the module with commands like make, sudo insmod hello.ko (to insert), lsmod | grep hello (to verify), and sudo rmmod hello (to remove)with commands like make, sudo insmod hello.ko (to insert), lsmod | grep hello (to verify), and sudo rmmod hello (to remove).

Task 1.B:

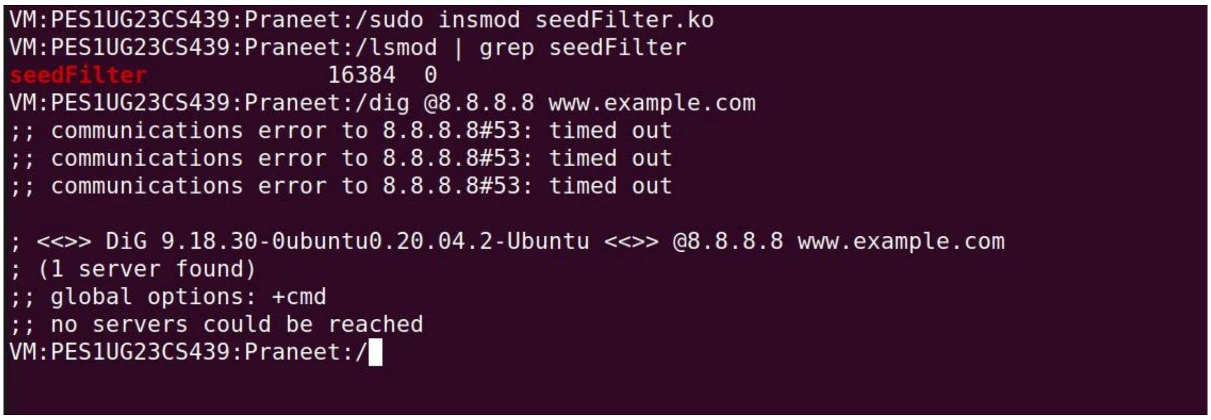
VM:



```
VM:PES1UG23CS439:Praneet:/dig @8.8.8.8 www.example.com

; <<>> DiG 9.18.30-0ubuntu0.20.04.2-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26144
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.example.com.        8       IN      CNAME   www.example.com-v4.edgesuite.net.
www.example.com-v4.edgesuite.net. 21449 IN CNAME a1422.dscr.akamai.net.
a1422.dscr.akamai.net.  20      IN      A       23.15.37.24
a1422.dscr.akamai.net.  20      IN      A       23.15.37.89

;; Query time: 172 msec
;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)
;; WHEN: Wed Oct 22 08:27:57 EDT 2025
;; MSG SIZE  rcvd: 154
```

This shows a normal, successful dig command. You send a DNS query to Google's server (8.8.8.8) and it correctly replied with the IP addresses for www.example.com. This proves that networking is working fine before your firewall is loaded.

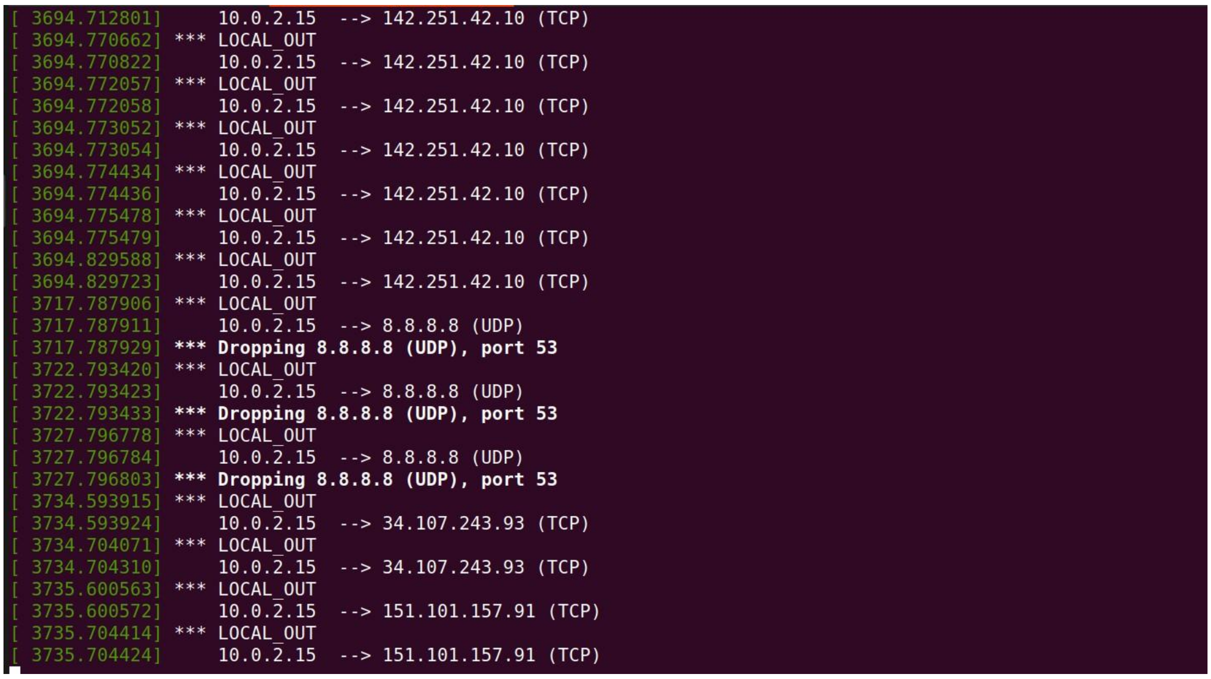
VM:



```
VM:PES1UG23CS439:Praneet:/sudo insmod seedFilter.ko
VM:PES1UG23CS439:Praneet:/lsmod | grep seedFilter
seedFilter             16384  0
VM:PES1UG23CS439:Praneet:/dig @8.8.8.8 www.example.com
;; communications error to 8.8.8.8#53: timed out
;; communications error to 8.8.8.8#53: timed out
;; communications error to 8.8.8.8#53: timed out

; <<>> DiG 9.18.30-0ubuntu0.20.04.2-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; no servers could be reached
VM:PES1UG23CS439:Praneet:/
```

Here, i've successfully loaded the seedFilter.ko module using insmod (and also confirmed it with lsmod). When you immediately try the same dig command it now fails with a "timed out" error. This is an indicator that the firewall is working.

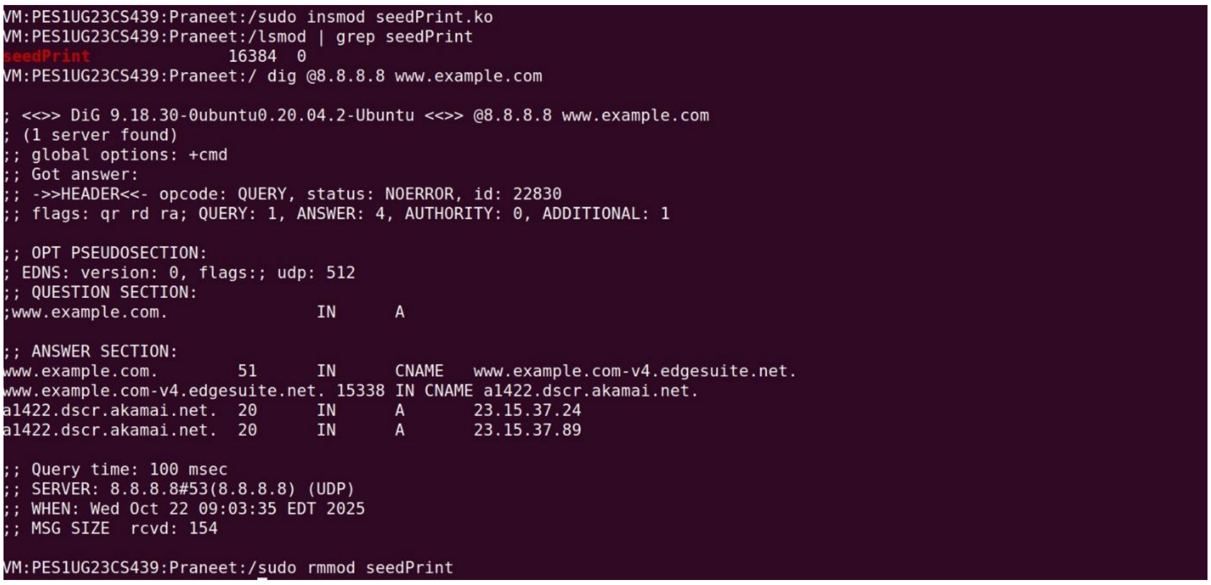
VM:



```
[ 3694.712801]     10.0.2.15  --> 142.251.42.10 (TCP)
[ 3694.770662] *** LOCAL_OUT
[ 3694.770822]     10.0.2.15  --> 142.251.42.10 (TCP)
[ 3694.772057] *** LOCAL_OUT
[ 3694.772058]     10.0.2.15  --> 142.251.42.10 (TCP)
[ 3694.773052] *** LOCAL_OUT
[ 3694.773054]     10.0.2.15  --> 142.251.42.10 (TCP)
[ 3694.774434] *** LOCAL_OUT
[ 3694.774436]     10.0.2.15  --> 142.251.42.10 (TCP)
[ 3694.775478] *** LOCAL_OUT
[ 3694.775479]     10.0.2.15  --> 142.251.42.10 (TCP)
[ 3694.829588] *** LOCAL_OUT
[ 3694.829723]     10.0.2.15  --> 142.251.42.10 (TCP)
[ 3717.787906] *** LOCAL_OUT
[ 3717.787911]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 3717.787929] *** Dropping 8.8.8.8 (UDP), port 53
[ 3722.793420] *** LOCAL_OUT
[ 3722.793423]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 3722.793433] *** Dropping 8.8.8.8 (UDP), port 53
[ 3727.796778] *** LOCAL_OUT
[ 3727.796784]     10.0.2.15  --> 8.8.8.8 (UDP)
[ 3727.796803] *** Dropping 8.8.8.8 (UDP), port 53
[ 3734.593915] *** LOCAL_OUT
[ 3734.593924]     10.0.2.15  --> 34.107.243.93 (TCP)
[ 3734.704071] *** LOCAL_OUT
[ 3734.704310]     10.0.2.15  --> 34.107.243.93 (TCP)
[ 3735.600563] *** LOCAL_OUT
[ 3735.600572]     10.0.2.15  --> 151.101.157.91 (TCP)
[ 3735.704414] *** LOCAL_OUT
[ 3735.704424]     10.0.2.15  --> 151.101.157.91 (TCP)
```

This dmesg log shows why the dig command timed out. The kernel is printing messages from your module, showing it's dropping the outgoing UDP packets destined for 8.8.8.8 on port 53 (the DNS port). The filter is correctly identifying and blocking the target traffic.

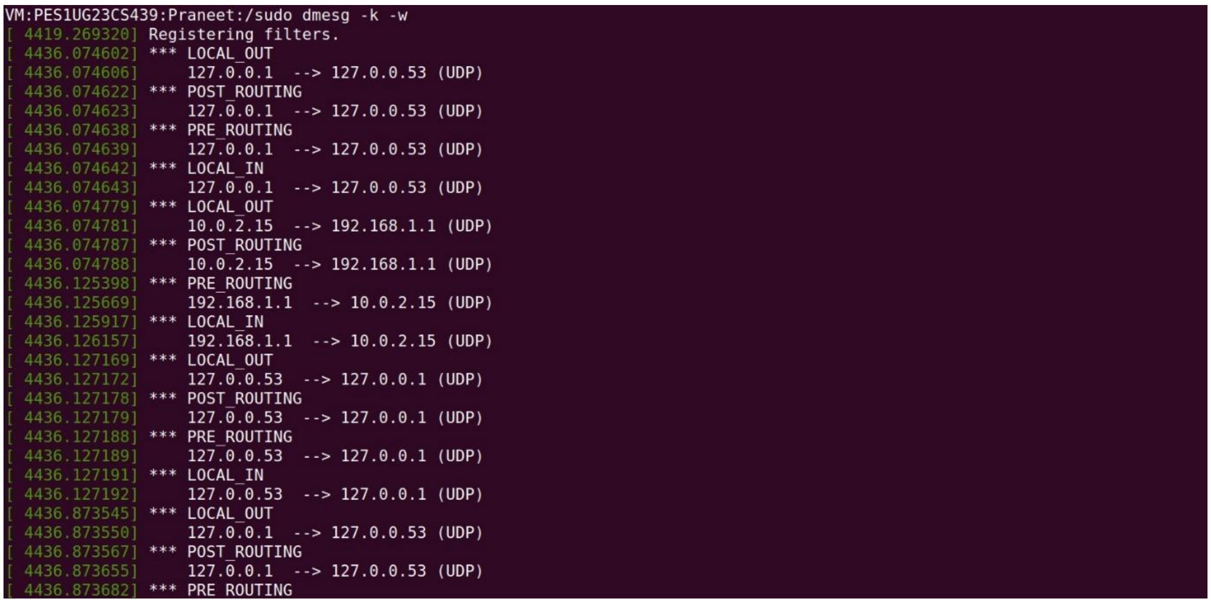
VM:



```
VM:PES1UG23CS439:Praneet:/sudo insmod seedPrint.ko
VM:PES1UG23CS439:Praneet:/lsmod | grep seedPrint
seedPrint              16384  0
VM:PES1UG23CS439:Praneet:/ dig @8.8.8.8 www.example.com

; <<>> DiG 9.18.30-0ubuntu0.20.04.2-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22830
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.example.com.        51      IN      CNAME   www.example.com-v4.edgesuite.net.
www.example.com-v4.edgesuite.net. 15338 IN CNAME a1422.dscr.akamai.net.
a1422.dscr.akamai.net.  20      IN      A       23.15.37.24
a1422.dscr.akamai.net.  20      IN      A       23.15.37.89

;; Query time: 100 msec
;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)
;; WHEN: Wed Oct 22 09:03:35 EDT 2025
;; MSG SIZE  rcvd: 154

VM:PES1UG23CS439:Praneet:/sudo rmmod seedPrint
```

Loading the seedPrint.ko module. This time, when I run the dig command, it works perfectly. This is expected as this module's function is to print and not to block.

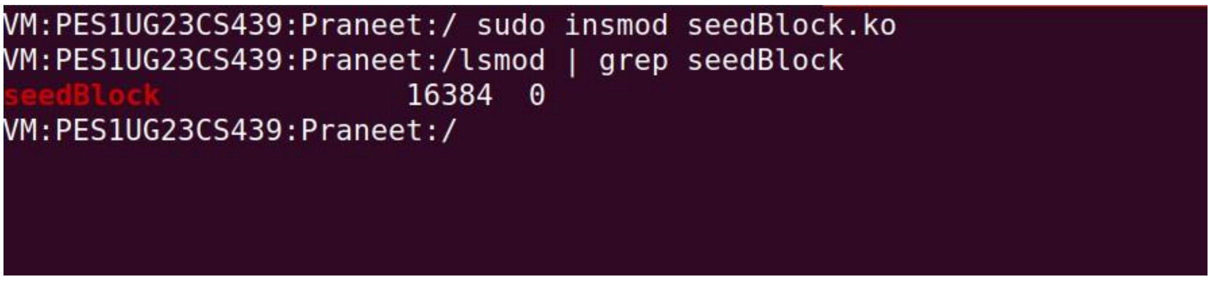
VM:


```
VM:PES1UG23CS439:Praneet:/sudo dmesg -k -w
[ 4419.269320] Registering filters.
[ 4436.074602] *** LOCAL_OUT
[ 4436.074606]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 4436.074622] *** POST_ROUTING
[ 4436.074623]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 4436.074638] *** PRE_ROUTING
[ 4436.074639]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 4436.074642] *** LOCAL_IN
[ 4436.074643]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 4436.074779] *** LOCAL_OUT
[ 4436.074781]     10.0.2.15  --> 192.168.1.1 (UDP)
[ 4436.074787] *** POST_ROUTING
[ 4436.074788]     10.0.2.15  --> 192.168.1.1 (UDP)
[ 4436.125398] *** PRE_ROUTING
[ 4436.125669]     192.168.1.1  --> 10.0.2.15 (UDP)
[ 4436.125917] *** LOCAL_IN
[ 4436.126157]     192.168.1.1  --> 10.0.2.15 (UDP)
[ 4436.127169] *** LOCAL_OUT
[ 4436.127172]     127.0.0.53  --> 127.0.0.1 (UDP)
[ 4436.127178] *** POST_ROUTING
[ 4436.127179]     127.0.0.53  --> 127.0.0.1 (UDP)
[ 4436.127188] *** PRE_ROUTING
[ 4436.127189]     127.0.0.53  --> 127.0.0.1 (UDP)
[ 4436.127191] *** LOCAL_IN
[ 4436.127192]     127.0.0.53  --> 127.0.0.1 (UDP)
[ 4436.873545] *** LOCAL_OUT
[ 4436.873550]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 4436.873567] *** POST_ROUTING
[ 4436.873655]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 4436.873682] *** PRE_ROUTING
```

It shows the path the dig packets took through the kernel's network stack.

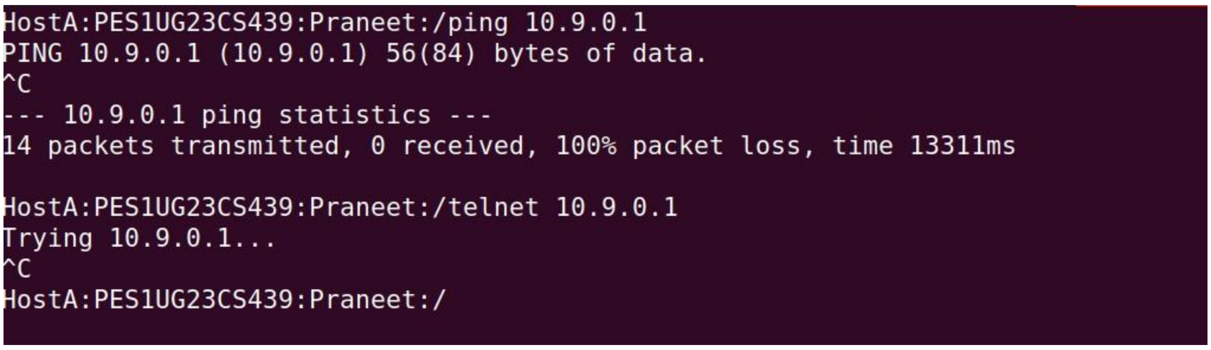
- **LOCAL_OUT:** VM's dig process creating a packet.
- **POST_ROUTING:** The packet leaving your VM (going to 127.0.0.53 which is a local DNS resolver and 192.168.1.1 which is the router).
- **PRE_ROUTING:** The DNS reply packet arriving back at the VM from the network.
- **LOCAL_IN:** The reply packet being delivered to the local dig process.
- I most probably didn't see NF_INET_FORWARD because the VM wasn't acting as a router it was the origin or final destination of the packets.

Vm:


```
VM:PES1UG23CS439:Praneet:/ sudo insmod seedBlock.ko
VM:PES1UG23CS439:Praneet:/lsmod | grep seedBlock
seedBlock              16384  0
VM:PES1UG23CS439:Praneet:/
```

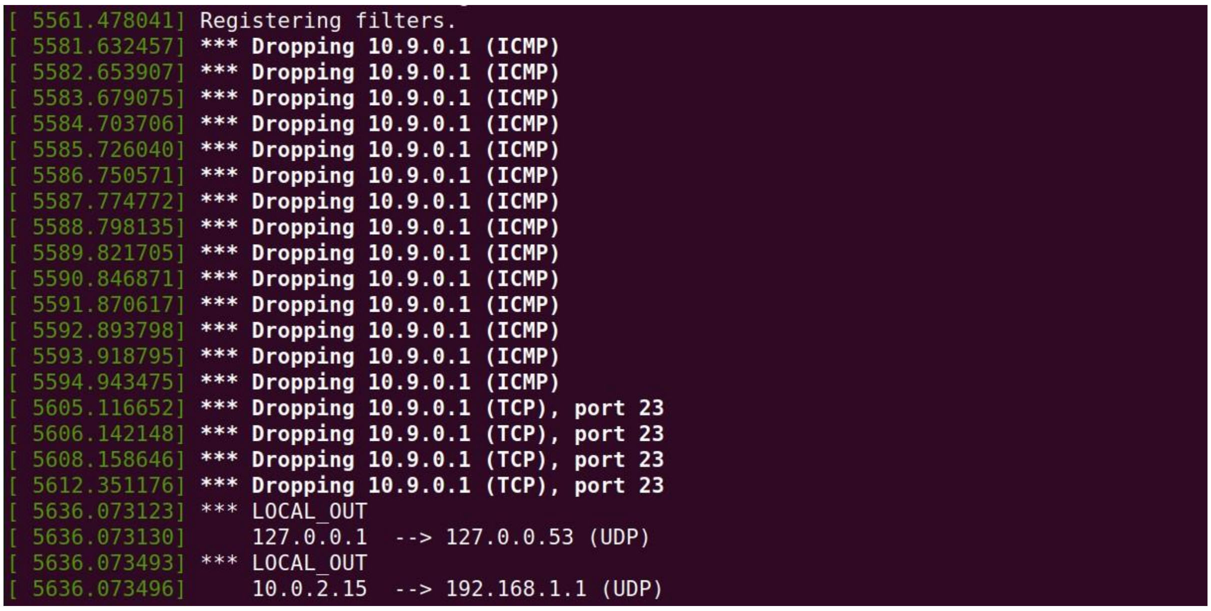
This simply shows that it successfully loaded the seedBlock.ko module on the VM getting it ready to block incoming traffic.

VM:


```
HostA:PES1UG23CS439:Praneet:/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 ---
14 packets transmitted, 0 received, 100% packet loss, time 13311ms

HostA:PES1UG23CS439:Praneet:/telnet 10.9.0.1
Trying 10.9.0.1...
^C
HostA:PES1UG23CS439:Praneet:/
```

This screenshot is from the other machine (Host A). From here I tried to ping VM (10.9.0.1) and got "100% packet loss." I then tried to telnet to it and the connection just hung at "Trying..." until I cancelled it. This shows that from an external perspective, VM is successfully deflecting these two types of traffic.

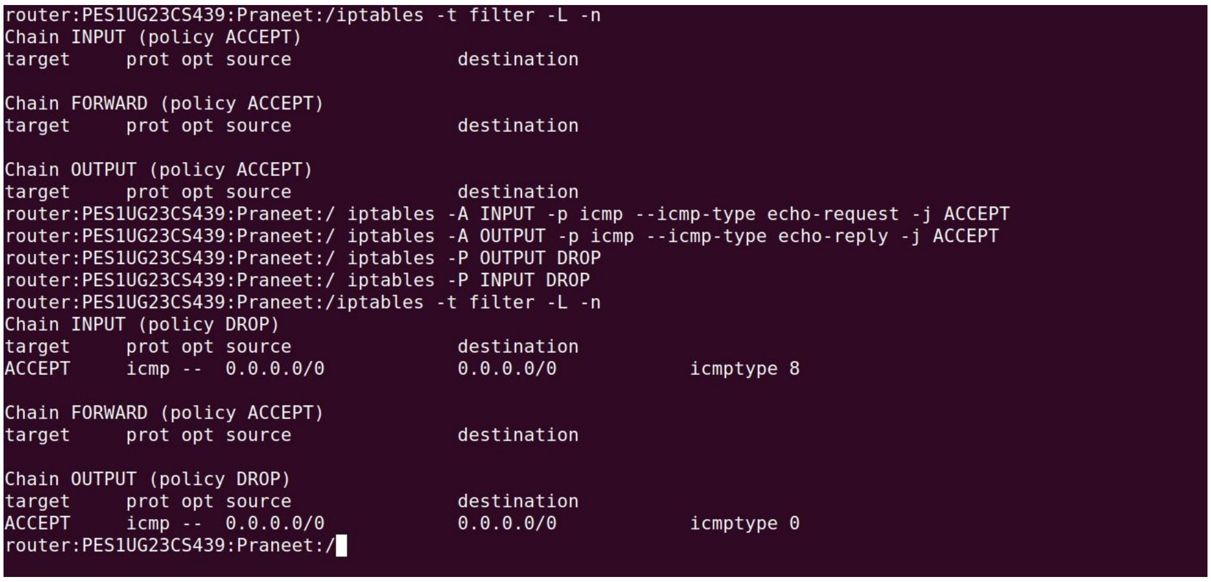
VM:


```
[ 5561.478041] Registering filters.
[ 5581.632457] *** Dropping 10.9.0.1 (ICMP)
[ 5582.653907] *** Dropping 10.9.0.1 (ICMP)
[ 5583.679075] *** Dropping 10.9.0.1 (ICMP)
[ 5584.703706] *** Dropping 10.9.0.1 (ICMP)
[ 5585.726040] *** Dropping 10.9.0.1 (ICMP)
[ 5586.750571] *** Dropping 10.9.0.1 (ICMP)
[ 5587.774772] *** Dropping 10.9.0.1 (ICMP)
[ 5588.798135] *** Dropping 10.9.0.1 (ICMP)
[ 5589.821705] *** Dropping 10.9.0.1 (ICMP)
[ 5590.846871] *** Dropping 10.9.0.1 (ICMP)
[ 5591.870617] *** Dropping 10.9.0.1 (ICMP)
[ 5592.893798] *** Dropping 10.9.0.1 (ICMP)
[ 5593.918795] *** Dropping 10.9.0.1 (ICMP)
[ 5594.943475] *** Dropping 10.9.0.1 (ICMP)
[ 5605.116652] *** Dropping 10.9.0.1 (TCP), port 23
[ 5606.142148] *** Dropping 10.9.0.1 (TCP), port 23
[ 5608.158646] *** Dropping 10.9.0.1 (TCP), port 23
[ 5612.351176] *** Dropping 10.9.0.1 (TCP), port 23
[ 5636.073123] *** LOCAL_OUT
[ 5636.073130]     127.0.0.1  --> 127.0.0.53 (UDP)
[ 5636.073493] *** LOCAL_OUT
[ 5636.073496]     10.0.2.15  --> 192.168.1.1 (UDP)
```

This dmesg log from shows the firewall. It's logging every time it drops an incoming packet. You can clearly see it blocking **ICMP** packets (the pings) and **TCP** packets destined for **port 23** (the telnet attempts). This confirms that the module is the reason the connections in the previous image failed.

Task 2.A:

Router:



```
router:PES1UG23CS439:Praneet:/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
router:PES1UG23CS439:Praneet:/ iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
router:PES1UG23CS439:Praneet:/ iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
router:PES1UG23CS439:Praneet:/ iptables -P OUTPUT DROP
router:PES1UG23CS439:Praneet:/ iptables -P INPUT DROP
router:PES1UG23CS439:Praneet:/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
router:PES1UG23CS439:Praneet:/
```

This screenshot shows the config of the iptables firewall on the seed-router machine.

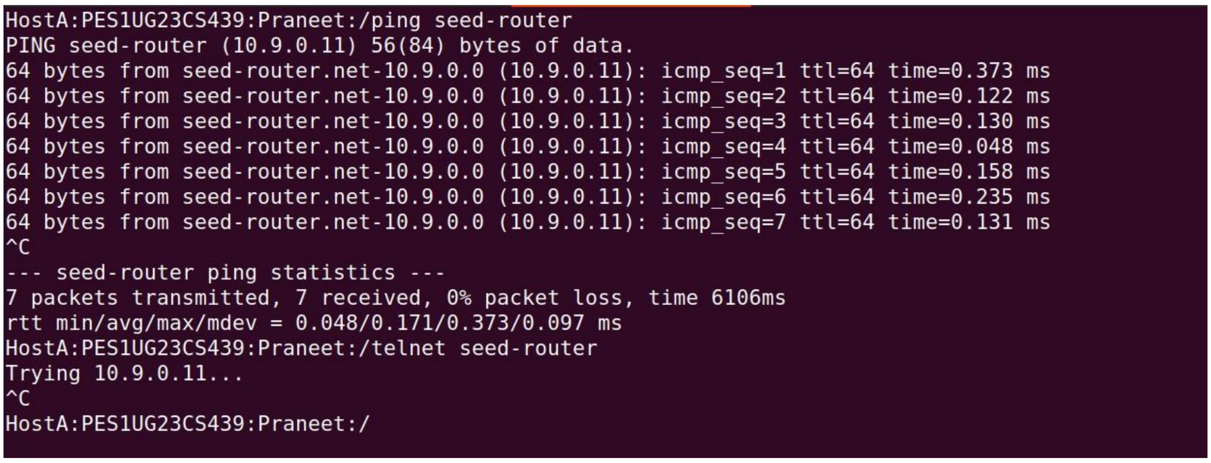
Initial State: The first command, iptables -t filter -L -n, shows that all chains (INPUT, FORWARD, OUTPUT) have a default policy of **ACCEPT**. This is a "wide-open" firewall where it allows all traffic by default.

Changing Policy: then execute four commands that change this behaviour:

1. iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT: This adds a rule to the **INPUT** chain. It specifically looks for ICMP packets of type 8 (which is an "echo-request," or a ping) and explicitly **ACCEPTS** them.
2. iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT: This adds a rule to the **OUTPUT** chain. It allows the router to send ICMP packets of type 0 (an "echo-reply," where it as an answer to a ping).
3. iptables -P OUTPUT DROP: This changes the default policy for the **OUTPUT** chain. It's a "default-deny" stance which means "Unless a rule explicitly says to **ACCEPT** it **DROP** all outgoing traffic."
4. iptables -P INPUT DROP: This does the same for the **INPUT** chain, setting the default to **DROP** all incoming traffic.

Final State: The last iptables -t filter -L -n command confirms the new setup. Both the INPUT and OUTPUT chains now show (policy DROP) and each chain has its single ACCEPT rule for ICMP traffic.

Host A:


```
HostA:PES1UG23CS439:Praneet:/ping seed-router
PING seed-router (10.9.0.11) 56(84) bytes of data.
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=1 ttl=64 time=0.373 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=2 ttl=64 time=0.122 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=3 ttl=64 time=0.130 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=4 ttl=64 time=0.048 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=5 ttl=64 time=0.158 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=6 ttl=64 time=0.235 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=7 ttl=64 time=0.131 ms
^C
--- seed-router ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6106ms
rtt min/avg/max/mdev = 0.048/0.171/0.373/0.097 ms
HostA:PES1UG23CS439:Praneet:/telnet seed-router
Trying 10.9.0.11...
^C
HostA:PES1UG23CS439:Praneet:/
```

This shows the results of the new firewall rules.

ping seed-router: This command is **successful**. sending 7 packets and received 7 packets back. This works precisely because of the two ACCEPT rules added in the previous step, which created a specific opening in the firewall to allow ping requests in and ping replies out.

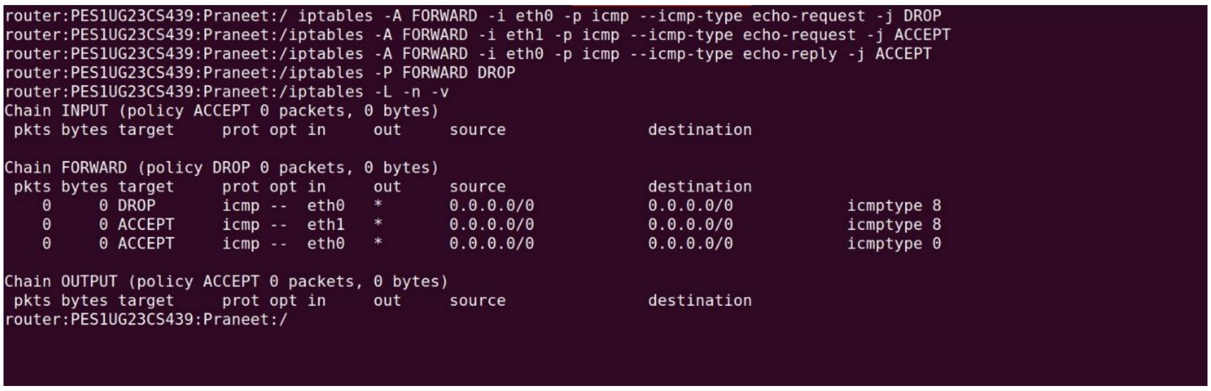
telnet seed-router: This command **fails**. It just hangs at "Trying 10.9.0.11..." until i manually cancelled it . This is because telnet uses the TCP protocol. When TCP packet arrived at the router, the INPUT chain was checked. It did not match the only ACCEPT rule (which was for ICMP), so it fell through to the default policy: **DROP**. The router discarded the packet, and HostA never received a response.

Q 1) the ping was successful with 0% packet loss. This is because we created a stateless rule exception. explicitly allowed incoming icmp echo-request packets on the INPUT chain and outgoing icmp echo-reply packets on the OUTPUT chain. Even though the default policy is to DROP everything, these specific rules are matched first, allowing the ping to succeed.

Q 2) the telnet command failed to connect and timed out. This is the firewall working as intended. The default INPUT policy is DROP. Since I only added an ACCEPT rule for ICMP (ping), the incoming telnet packet (which uses TCP) did not match any ACCEPT rule. It therefore hit the default DROP policy and was discarded preventing the connection.

Task 2.B:

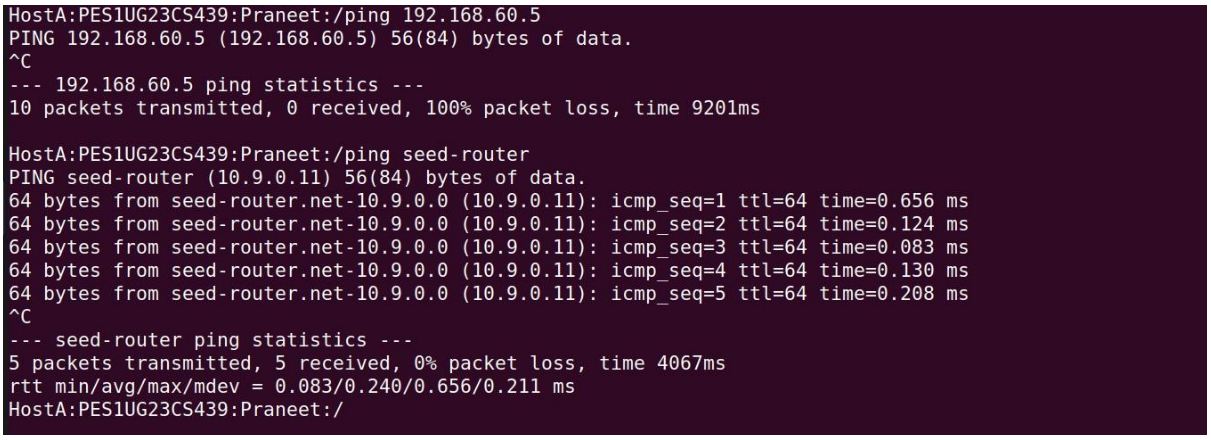
Router:



```
router:PES1UG23CS439:Praneet:/ iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-request -j DROP
router:PES1UG23CS439:Praneet:/iptables -A FORWARD -i eth1 -p icmp --icmp-type echo-request -j ACCEPT
router:PES1UG23CS439:Praneet:/iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-reply -j ACCEPT
router:PES1UG23CS439:Praneet:/iptables -P FORWARD DROP
router:PES1UG23CS439:Praneet:/iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 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
router:PES1UG23CS439:Praneet:/
```

we see the execution of the four iptables commands. The final iptables -L -n -v command confirms the new policy which is the **FORWARD** chain's default policy is now set to **DROP** and the three specific ACCEPT and DROP rules for ICMP traffic (pings) have been successfully added. This setup is designed to "forward" only very specific, approved traffic between the internal (eth1) and external (eth0) networks.

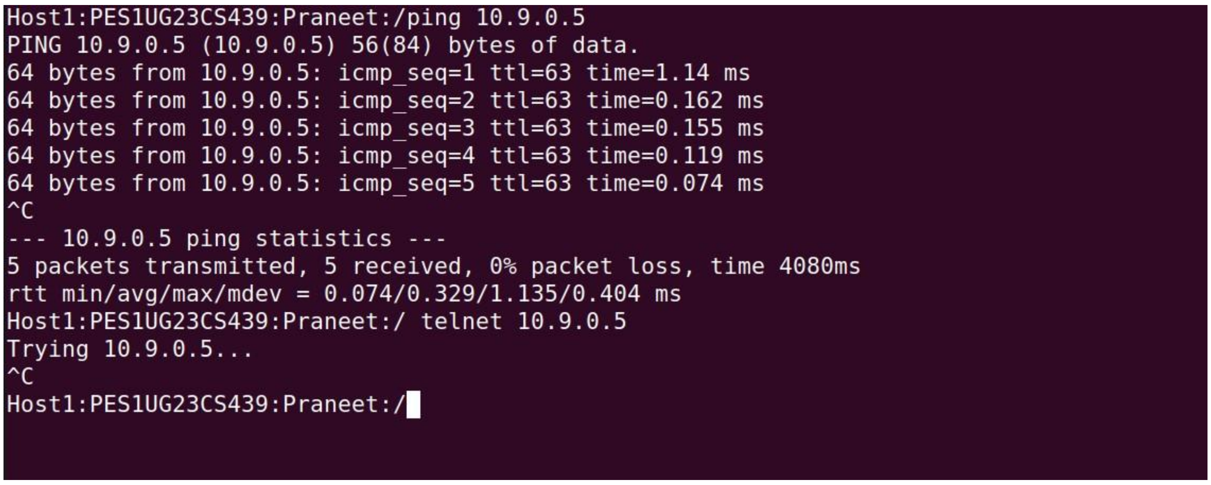
HostA:



```
HostA:PES1UG23CS439:Praneet:/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 ---
10 packets transmitted, 0 received, 100% packet loss, time 9201ms

HostA:PES1UG23CS439:Praneet:/ping seed-router
PING seed-router (10.9.0.11) 56(84) bytes of data.
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=1 ttl=64 time=0.656 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=2 ttl=64 time=0.124 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=3 ttl=64 time=0.083 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=4 ttl=64 time=0.130 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=5 ttl=64 time=0.208 ms
^C
--- seed-router ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4067ms
rtt min/avg/max/mdev = 0.083/0.240/0.656/0.211 ms
HostA:PES1UG23CS439:Praneet:/
```

Here testing from the outside network (HostA). Just as intended, the command **ping 192.168.60.5 (outside to inside) fails** with 100% packet loss. This is the firewall blocking the incoming ping. However, the command **ping seed-router (outside to router) succeeds**. This is a key distinction where this ping is handled by the router's INPUT chain not the FORWARD chain so it's allowed.

Host1:



```
Host1:PES1UG23CS439:Praneet:/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=1.14 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.162 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.155 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=63 time=0.119 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=63 time=0.074 ms
^C
--- 10.9.0.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4080ms
rtt min/avg/max/mdev = 0.074/0.329/1.135/0.404 ms
Host1:PES1UG23CS439:Praneet:/ telnet 10.9.0.5
Trying 10.9.0.5...
^C
Host1:PES1UG23CS439:Praneet:/
```

Here we see the tests from the internal network (host1). The command **ping 10.9.0.5 (inside to outside) works perfectly**. This demonstrates that the firewall is stateful (by the manual rules) to permit an internal user to ping an external server. But the **telnet 10.9.0.5 command fails** and times out. This confirms that the "default-deny" policy is working, blocking all non-ICMP traffic from being forwarded.

Purpose of each rule:

iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-request -j DROP

- **Purpose:** This is the protection for the internal network. It tells the router to look at any packet attempting to be **forwarded** that came in from the outside (-i eth0). If that packet is a ping request (icmp-type echo-request), it is immediately dropped.

iptables -A FORWARD -i eth1 -p icmp --icmp-type echo-request -j ACCEPT

- **Purpose:** This rule allows internal users to initiate pings to the outside world. It looks for packets coming in from the inside (-i eth1) that are ping requests and accepts them, allowing them to be forwarded out through eth0.

iptables -A FORWARD -i eth0 -p icmp --icmp-type echo-reply -j ACCEPT

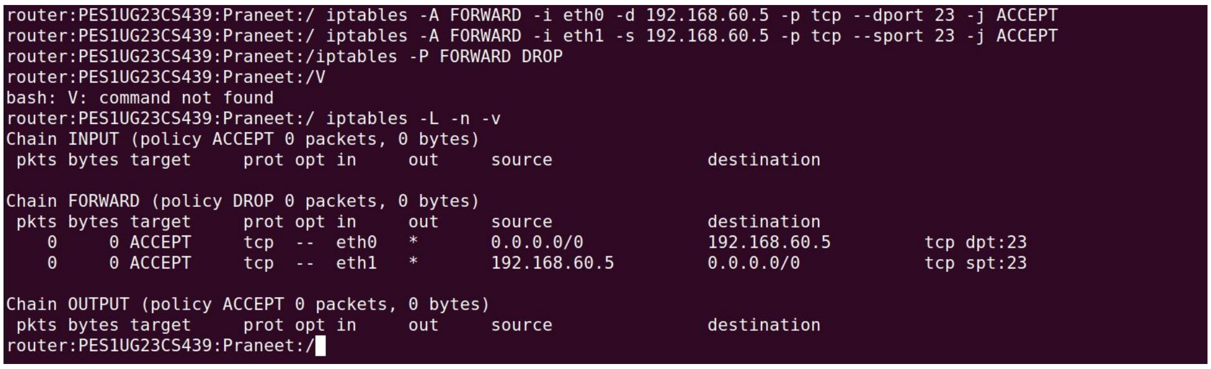
- **Purpose:** This is the crucial second half for allowing internal pings. When an internal host pings an outside host, the outside host sends a ping reply *back*. This rule looks for those replies coming in from the outside (-i eth0) and accepts them so they can be forwarded back to the original internal host. Without this, internal pings would go out but never get a response.

iptables -P FORWARD DROP

- **Purpose:** This is the "default-deny" or "catch-all" rule. It sets the default policy for the entire FORWARD chain to **DROP**. This means any packet that doesn't match one of the specific Aceept rules above is automatically discarded.

Task 2.C:

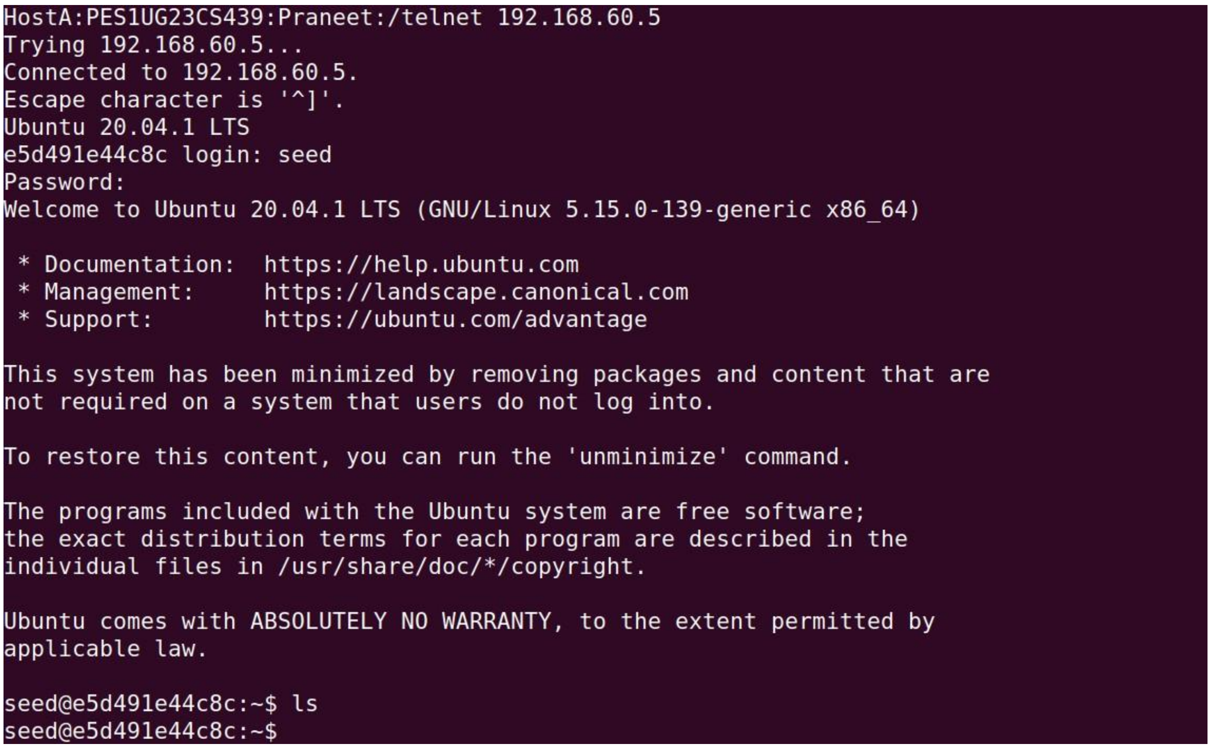
Router:



```
router:PES1UG23CS439:Praneet:/ iptables -A FORWARD -i eth0 -d 192.168.60.5 -p tcp --dport 23 -j ACCEPT
router:PES1UG23CS439:Praneet:/ iptables -A FORWARD -i eth1 -s 192.168.60.5 -p tcp --sport 23 -j ACCEPT
router:PES1UG23CS439:Praneet:/iptables -P FORWARD DROP
router:PES1UG23CS439:Praneet:/V
bash: V: command not found
router:PES1UG23CS439:Praneet:/ iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 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
router:PES1UG23CS439:Praneet:/
```

we see the firewall configuration being applied to the seed-router. The iptables -L -n -v command confirms that the default policy for the **FORWARD** chain has been set to **DROP**. This is a "default-deny" stance, meaning the router will block all traffic between the internal and external networks unless it's specifically allowed. The two ACCEPT rules are added to create a single and narrow exception for the telnet service on one specific host.

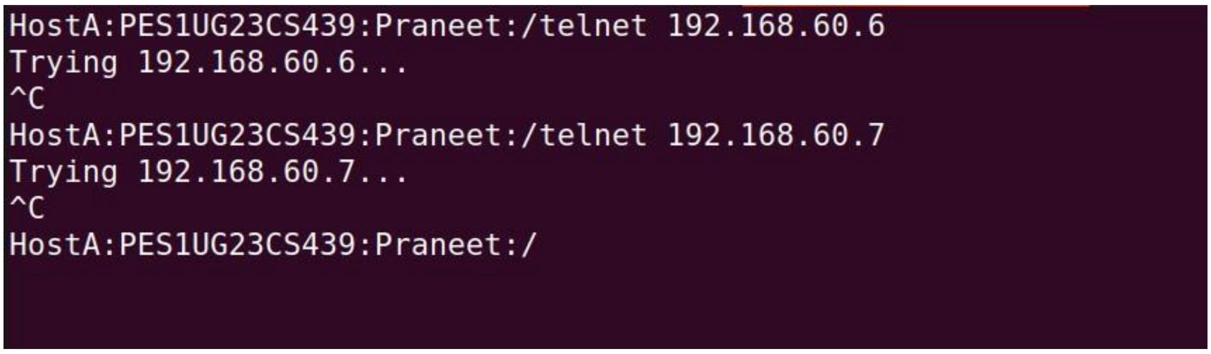
Host A:



```
HostA:PES1UG23CS439:Praneet:/telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
e5d491e44c8c login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-139-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
applicable law.

seed@e5d491e44c8c:~$ ls
seed@e5d491e44c8c:~$
```

The tests from the outside (HostA). As intended by the firewall policy, the telnet 192.168.60.5 command here is **successful**. We see the login prompt because this traffic (to .5 on port 23) perfectly matched the first ACCEPT rule.

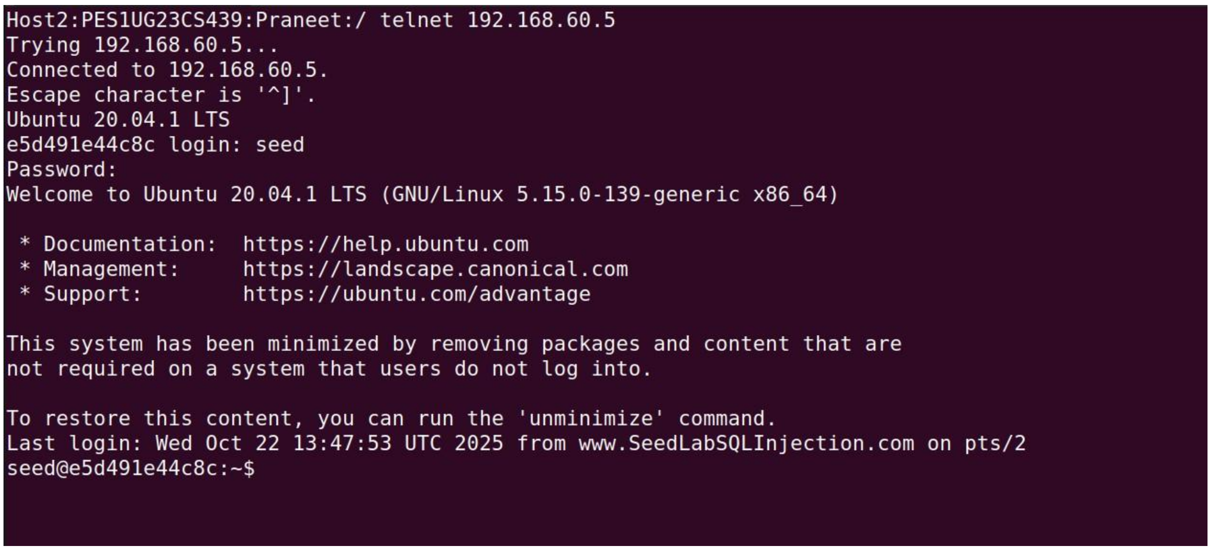
Host A:



```
HostA:PES1UG23CS439:Praneet:/telnet 192.168.60.6
Trying 192.168.60.6...
^C
HostA:PES1UG23CS439:Praneet:/telnet 192.168.60.7
Trying 192.168.60.7...
^C
HostA:PES1UG23CS439:Praneet:/
```

Here the telnet attempts to 192.168.60.6 and 192.168.60.7 both **fail** and time out. This is also the firewall working correctly. Since this traffic was not destined for .5 it didn't match the exception rule and was discarded by the default DROP policy.

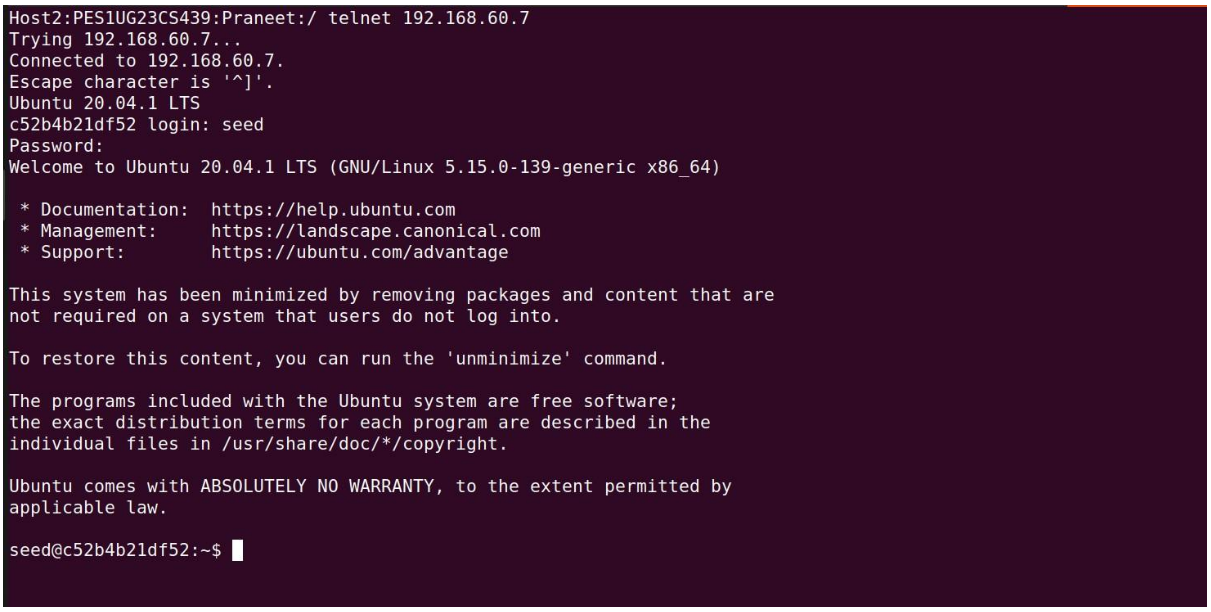
Host 2:



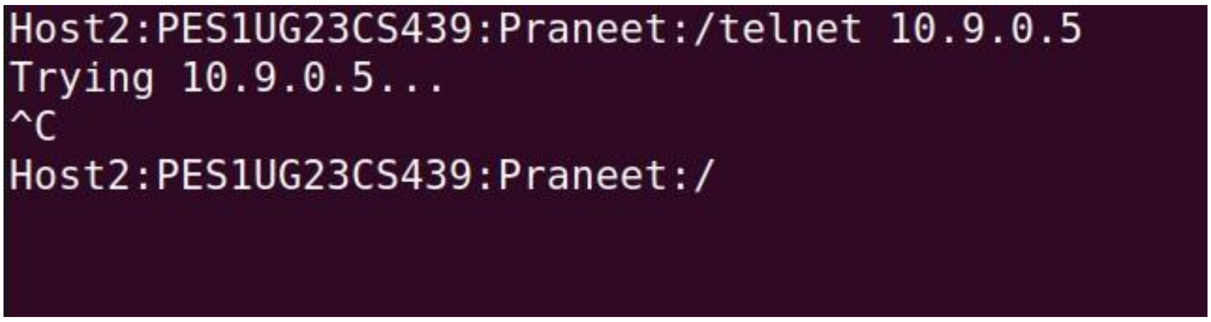
```
Host2:PES1UG23CS439:Praneet:/ telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
e5d491e44c8c login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-139-generic x86_64)

 * 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: Wed Oct 22 13:47:53 UTC 2025 from www.SeedLabSQLInjection.com on pts/2
seed@e5d491e44c8c:~$
```

These tests from within the internal network (Host2). We see that Host2 (.6) can **successfully** telnet to both 192.168.60.5 and 192.168.60.7. This demonstrates that internal hosts can freely communicate. This is because this traffic is all on the same subnet (192.168.60.0/24) and is handled by the network switch and it never goes to the router so the FORWARD rules don't even apply.

Host 2:


```
Host2:PES1UG23CS439:Praneet:/ telnet 192.168.60.7
Trying 192.168.60.7...
Connected to 192.168.60.7.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
c52b4b21df52 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-139-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
applicable law.

seed@c52b4b21df52:~$ 
```

Host 2:


```
Host2:PES1UG23CS439:Praneet:/telnet 10.9.0.5
Trying 10.9.0.5...
^C
Host2:PES1UG23CS439:Praneet:/
```

This shows an internal host (Host2) attempting to telnet to an external server (10.9.0.5). This connection **fails**. This traffic does go to the router to be forwarded, but it doesn't match any ACCEPT rules (which were only for incoming traffic). Therefore, it's caught and discarded by the default DROP policy, successfully preventing internal hosts from accessing external servers.

Purpose of the rules:

iptables -A FORWARD -i eth0 -d 192.168.60.5 -p tcp --dport 23 -j ACCEPT

- **Purpose:** This is the rule that allows specific external access. It tells the router to **ACCEPT** packets that are:
  - Coming in from the outside (-i eth0).
  - Are destined only for the server 192.168.60.5 (-d 192.168.60.5).
  - Are using the TCP protocol (-p tcp) and aiming for the telnet port (--dport 23).

iptables -A FORWARD -i eth1 -s 192.168.60.5 -p tcp --sport 23 -j ACCEPT

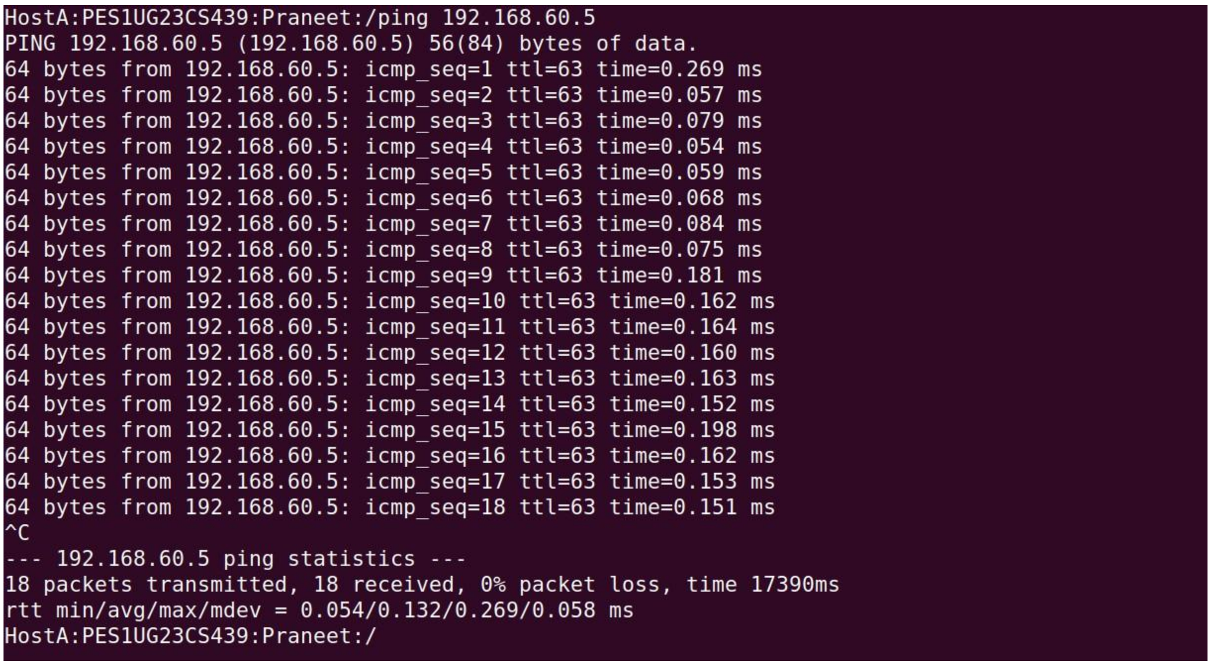
- **Purpose:** This is the reply rule. Since these are stateless rules, we must also explicitly allow the *answers* from the server to get back out. This rule accepts the packets that are:
    - Coming in from the inside (-i eth1).
    - Are coming from the server 192.168.60.5 (-s 192.168.60.5).
    - Are using TCP and originating from the telnet port (--sport 23).

iptables -P FORWARD DROP

- **Purpose:** This is the master original "default-deny" policy. It sets the default action for all packets on the FORWARD chain to **DROP**.

Task 3.A:

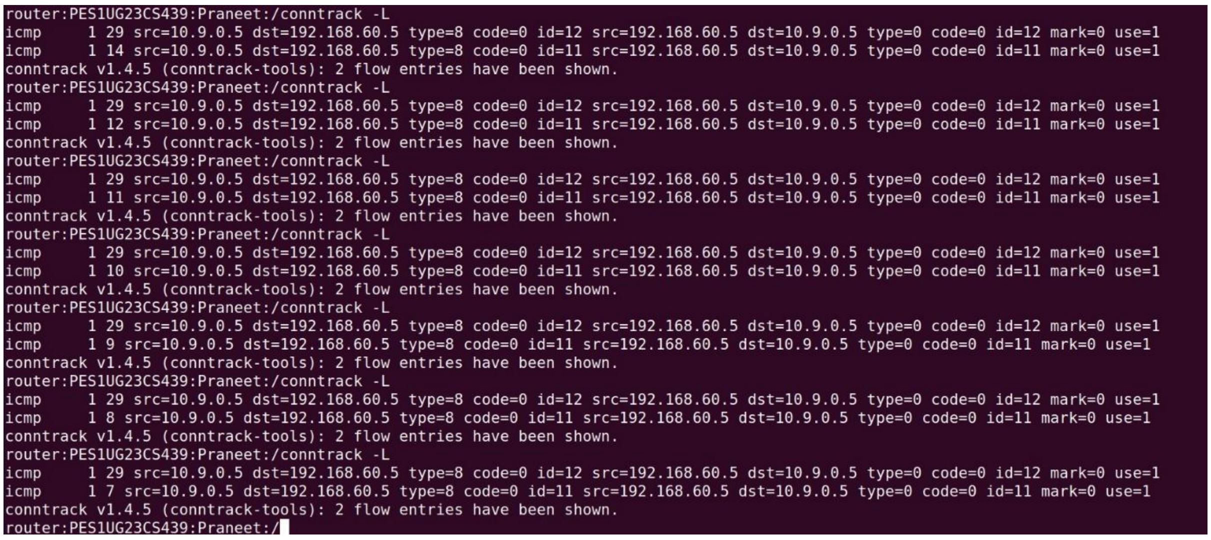
Host A:



```
HostA:PES1UG23CS439:Praneet:/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=0.269 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.079 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.054 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.068 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.075 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.181 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.162 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.164 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.160 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.163 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.152 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.198 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.162 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.153 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.151 ms
^C
--- 192.168.60.5 ping statistics ---
18 packets transmitted, 18 received, 0% packet loss, time 17390ms
rtt min/avg/max/mdev = 0.054/0.132/0.269/0.058 ms
HostA:PES1UG23CS439:Praneet:/
```

Shows the **action** performed from HostA: where I successfully pinged the internal machine 192.168.60.5. This generated a stream of ICMP packets that had to pass through the seed-router to reach their destination.

Router:

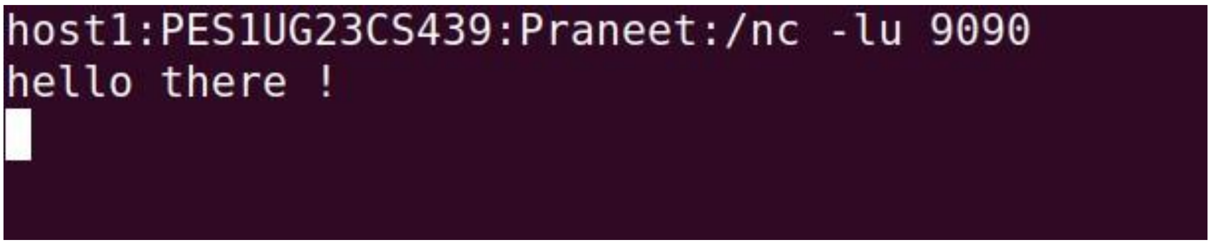


```
router:PES1UG23CS439:Praneet:/conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=12 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=12 mark=0 use=1
icmp     1 14 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=11 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=11 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=12 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=12 mark=0 use=1
icmp     1 12 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=11 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=11 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=12 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=12 mark=0 use=1
icmp     1 11 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=11 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=11 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=12 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=12 mark=0 use=1
icmp     1 10 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=11 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=11 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=12 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=12 mark=0 use=1
icmp     1 9 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=11 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=11 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=12 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=12 mark=0 use=1
icmp     1 8 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=11 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=11 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
icmp     1 29 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=12 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=12 mark=0 use=1
icmp     1 7 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=11 src=192.168.60.5 dst=10.9.0.5 type=0 code=0 id=11 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
router:PES1UG23CS439:Praneet:/
```

This screenshot captures the conntrack -L output on the seed-router just after the ping from HostA was stopped. It clearly shows that the kernel's connection tracking is stateful even for the connection-less ICMP protocol. It created two distinct flow entries where one for the ICMP echo-request (type=8) from the HostA to the server and another for the echo-reply (type=0) coming back. The most important observation is the third number on each line, which is a **timeout timer**. By running the command repeatedly, we see this timer actively **counting down** (from 29, to 12, 9, 8, etc.), showing that the state is decaying and will be removed from the table once the timer reaches zero, which appears to be after about 30 seconds of inactivity.

UDP Experiment:

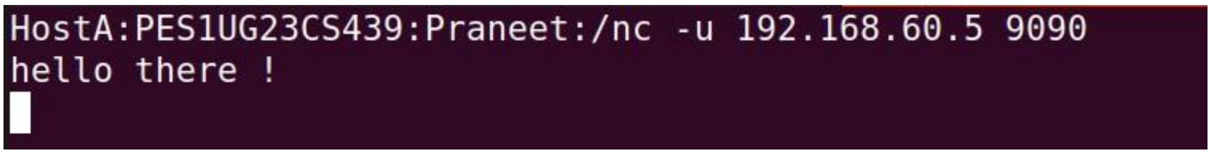
Host 1:



```
host1:PES1UG23CS439:Praneet:/nc -lu 9090
hello there !
```

This snip shows the **host1** machine (192.168.60.5) successfully receiving a UDP message. It was running a netcat listener (nc -lu 9090) on UDP port 9090, and the text "hello there !" has appeared, confirming it received the packet that was sent from HostA and allowed to pass through the router.

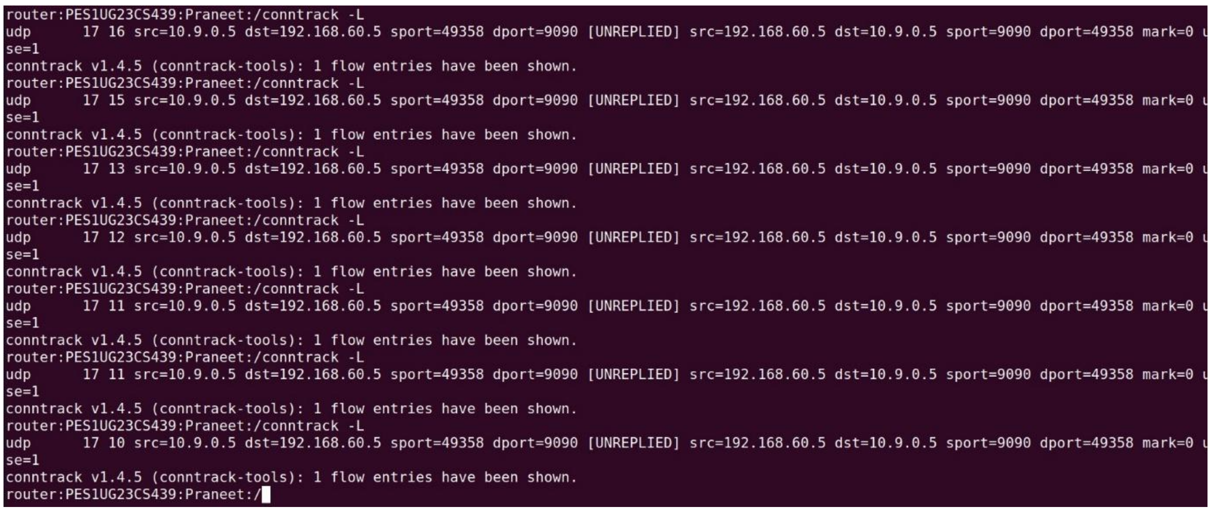
Host A:



```
HostA:PES1UG23CS439:Praneet:/nc -u 192.168.60.5 9090
hello there !
```

This screenshot shows the **HostA** machine (10.9.0.5) sending the UDP packet. Executing a netcat to send a UDP packet (-u) to the internal server's IP (192.168.60.5) on port 9090. The text "hello there !" is the data payload of the single packet being sent, which is the action that triggers the conntrack entry on the router.

Router:

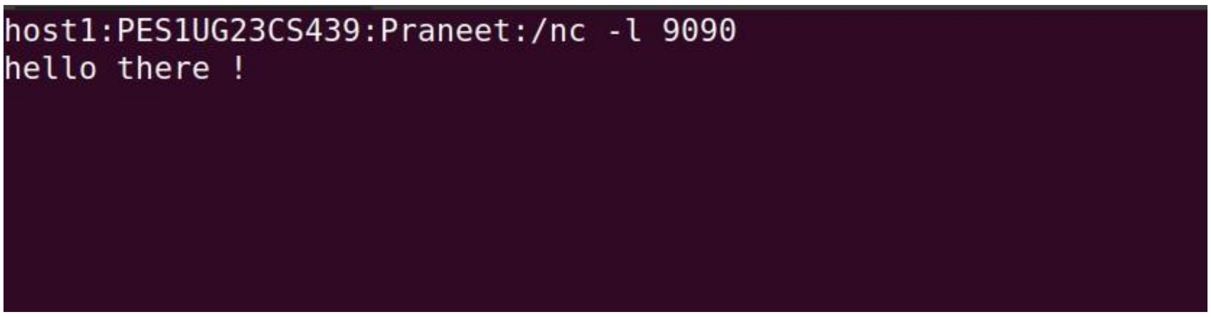


```
router:PES1UG23CS439:Praneet:/conntrack -L
udp      17 16 src=10.9.0.5 dst=192.168.60.5 sport=49358 dport=9090 [UNREPLIED] src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=49358 mark=0 u
se=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
udp      17 15 src=10.9.0.5 dst=192.168.60.5 sport=49358 dport=9090 [UNREPLIED] src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=49358 mark=0 u
se=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
udp      17 13 src=10.9.0.5 dst=192.168.60.5 sport=49358 dport=9090 [UNREPLIED] src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=49358 mark=0 u
se=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
udp      17 12 src=10.9.0.5 dst=192.168.60.5 sport=49358 dport=9090 [UNREPLIED] src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=49358 mark=0 u
se=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
udp      17 11 src=10.9.0.5 dst=192.168.60.5 sport=49358 dport=9090 [UNREPLIED] src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=49358 mark=0 u
se=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
udp      17 11 src=10.9.0.5 dst=192.168.60.5 sport=49358 dport=9090 [UNREPLIED] src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=49358 mark=0 u
se=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
udp      17 10 src=10.9.0.5 dst=192.168.60.5 sport=49358 dport=9090 [UNREPLIED] src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=49358 mark=0 u
se=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/
```

This screenshot from the **seed-router** shows the kernel's connection tracking table. After the UDP packet from HostA passed through, conntrack created a stateful entry. This entry is marked as **[UNREPLIED]**, as the server on host1 did not send a packet back. By running the conntrack -L command repeatedly, we can clearly see the entry's **timeout value** (the third number) **counting down** (from 17 to 10), showing that this "connectionless" state is temporary and will be removed after its short timeout expires.

TCP Experiment:

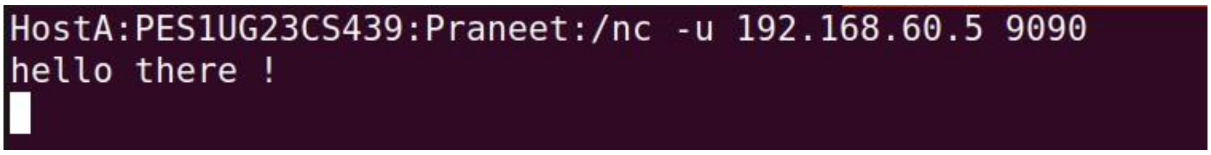
Host 1:



```
host1:PES1UG23CS439:Praneet:/nc -l 9090
hello there !
```

This screenshot shows the **host1** machine (192.168.60.5) running a netcat listener on TCP port 9090. It has successfully received the "hello there !" message, confirming that a client has connected and sent data over the TCP stream.

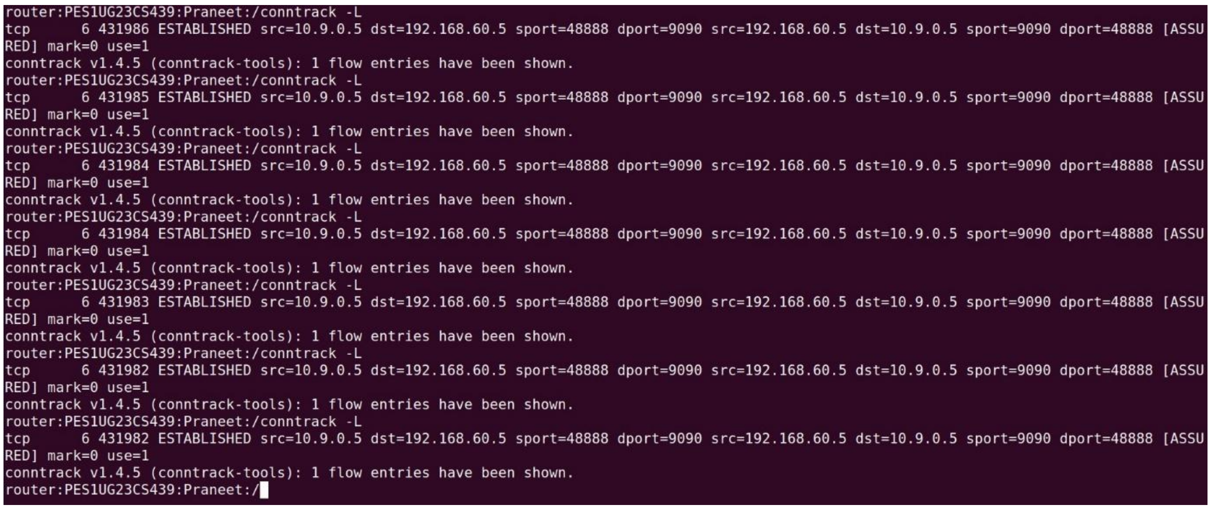
Host A:



```
HostA:PES1UG23CS439:Praneet:/nc -u 192.168.60.5 9090
hello there !
```

This is the **HostA client** machine (10.9.0.5) which initiated the TCP connection to host1's port 9090. The user typed "hello there !", sending that data to the server which successfully established the two-way connection.

Router:

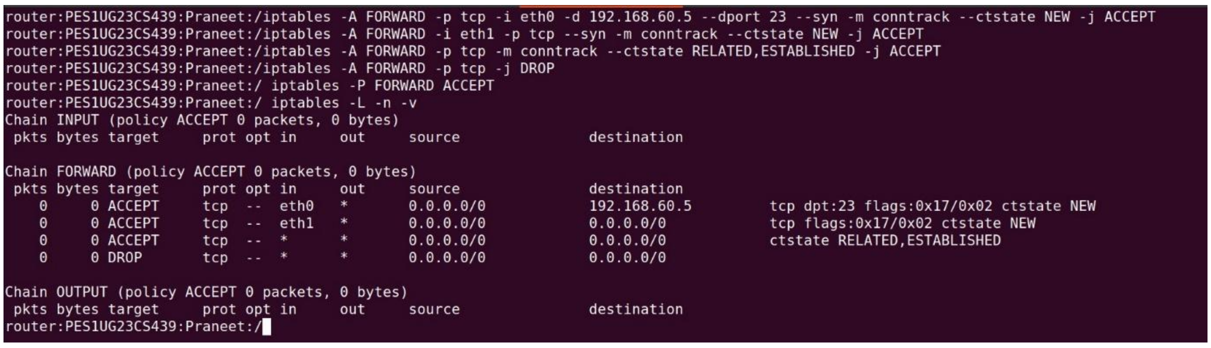


```
router:PES1UG23CS439:Praneet:/conntrack -L
tcp      6 431986 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=48888 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=48888 [ASSU
RED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
tcp      6 431985 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=48888 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=48888 [ASSU
RED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
tcp      6 431984 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=48888 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=48888 [ASSU
RED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
tcp      6 431984 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=48888 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=48888 [ASSU
RED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
tcp      6 431983 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=48888 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=48888 [ASSU
RED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
tcp      6 431982 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=48888 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=48888 [ASSU
RED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/conntrack -L
tcp      6 431982 ESTABLISHED src=10.9.0.5 dst=192.168.60.5 sport=48888 dport=9090 src=192.168.60.5 dst=10.9.0.5 sport=9090 dport=48888 [ASSU
RED] mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 1 flow entries have been shown.
router:PES1UG23CS439:Praneet:/
```

This conntrack -L output from the **seed-router** shows the kernel tracking the active TCP connection. It correctly identifies the flow as tcp and, most importantly, as **ESTABLISHED** and **[ASSURED]**, since the 3-way handshake is complete and data has passed. The timer is a very large number (e.g., 431986), which is the long idle timeout for an established TCP session (approx. 5 days), not a short-lived decay timer like we saw with ICMP or UDP.

Task 3.B:

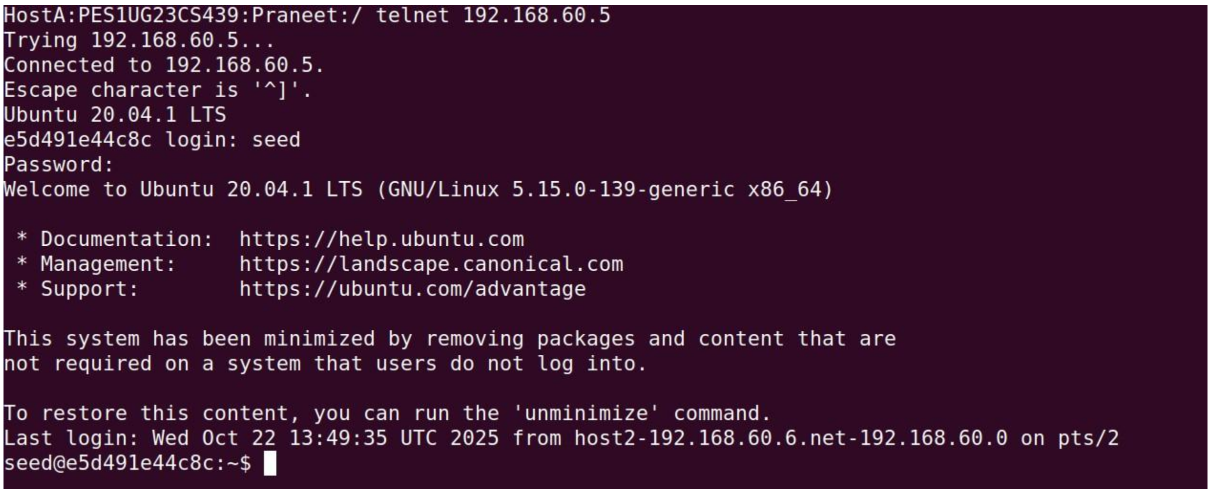
Router:



```
router:PES1UG23CS439:Praneet:/iptables -A FORWARD -p tcp -i eth0 -d 192.168.60.5 --dport 23 --syn -m conntrack --ctstate NEW -j ACCEPT
router:PES1UG23CS439:Praneet:/iptables -A FORWARD -i eth1 -p tcp --syn -m conntrack --ctstate NEW -j ACCEPT
router:PES1UG23CS439:Praneet:/iptables -A FORWARD -p tcp -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
router:PES1UG23CS439:Praneet:/iptables -A FORWARD -p tcp -j DROP
router:PES1UG23CS439:Praneet:/ iptables -P FORWARD ACCEPT
router:PES1UG23CS439:Praneet:/ 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 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 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
router:PES1UG23CS439:Praneet:/
```

This seed-router terminal shows the iptables rules for new stateful firewall. It's configured to ACCEPT NEW TCP connections from the outside only to .5 on port 23 (Rule 1), ACCEPT any NEW TCP connection from the inside (Rule 2), and then ACCEPT all ESTABLISHED traffic (Rule 3). A final DROP rule for TCP is added for cleanup, though the chain's default policy is ACCEPT (for non-TCP traffic).

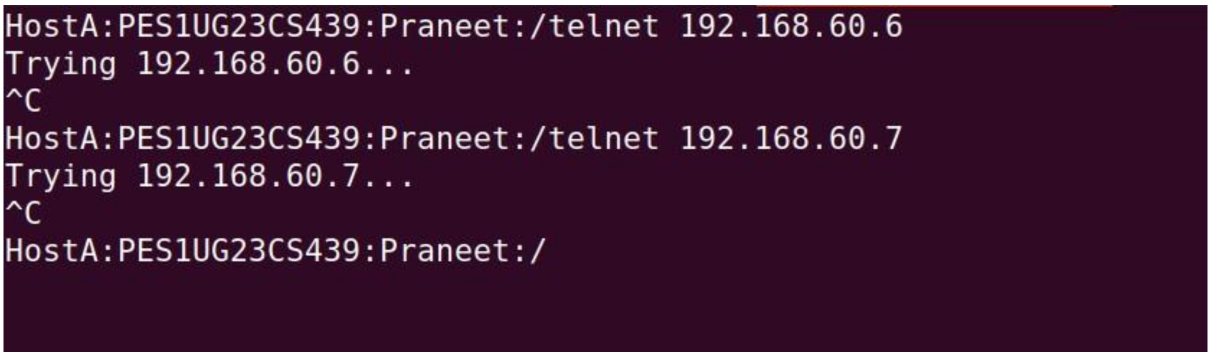
Host a:



```
HostA:PES1UG23CS439:Praneet:/ telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
e5d491e44c8c login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-139-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: Wed Oct 22 13:49:35 UTC 2025 from host2-192.168.60.6.net-192.168.60.0 on pts/2
seed@e5d491e44c8c:~$ 
```

This screenshot from HostA confirms the first rule works. The telnet to 192.168.60.5 **succeeded** and we're at the login. This is because the initial SYN packet matched the NEW connection rule we created specifically for this server.

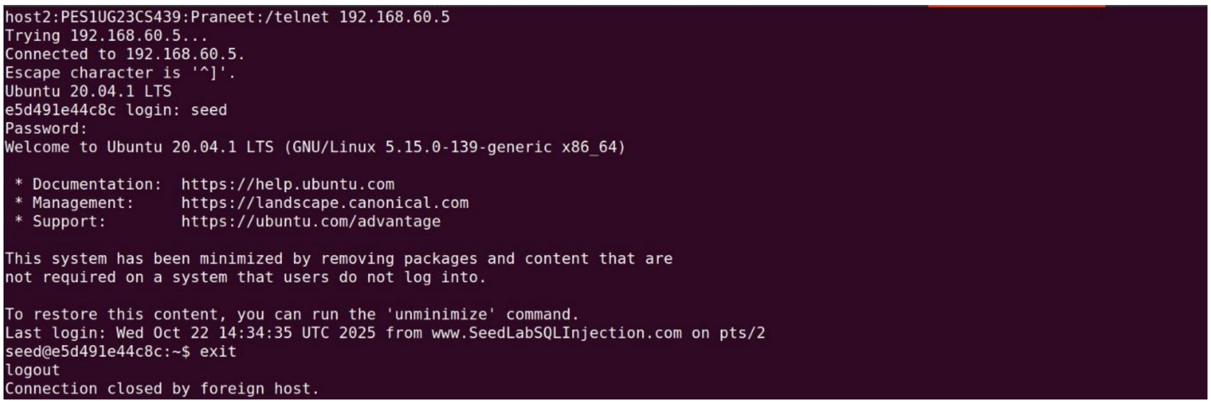
Host a:



```
HostA:PES1UG23CS439:Praneet:/telnet 192.168.60.6
Trying 192.168.60.6...
^C
HostA:PES1UG23CS439:Praneet:/telnet 192.168.60.7
Trying 192.168.60.7...
^C
HostA:PES1UG23CS439:Praneet:/
```

This HostA screenshot confirms firewall is correctly blocking unwanted traffic. The telnet attempts to 192.168.60.6 and 192.168.60.7 both **failed** and timed out. Their NEW connection packets didn't match any ACCEPT rule, so they were caught by the DROP rule.

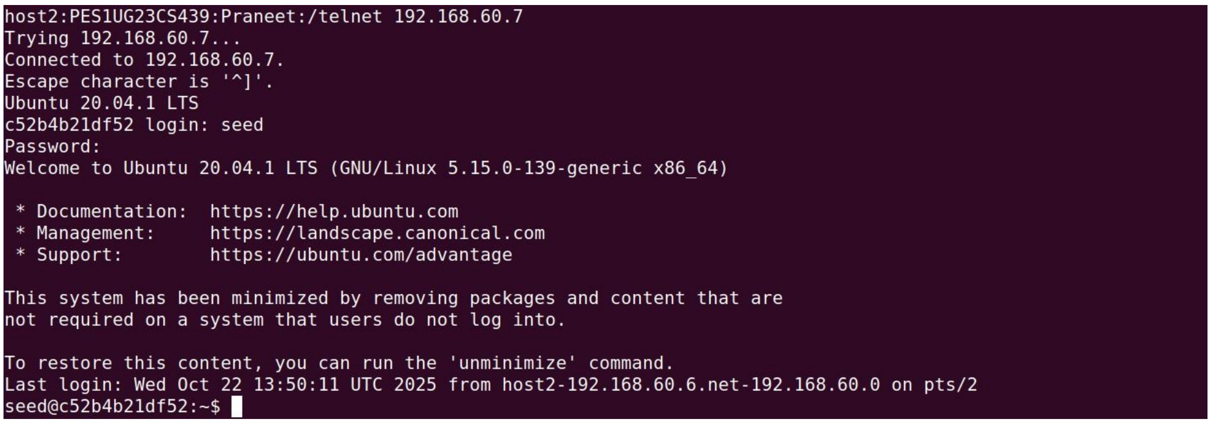
Host 2:



```
host2:PES1UG23CS439:Praneet:/telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
e5d491e44c8c login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-139-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: Wed Oct 22 14:34:35 UTC 2025 from www.SeedLabSQLInjection.com on pts/2
seed@e5d491e44c8c:~$ exit
logout
Connection closed by foreign host.
```

This screenshot from Host2 (.6) shows a **successful** telnet to the internal server 192.168.60.5. This connection works because both hosts are on the same internal network, so the traffic is just switched locally and never has to pass through the router's FORWARD chain.

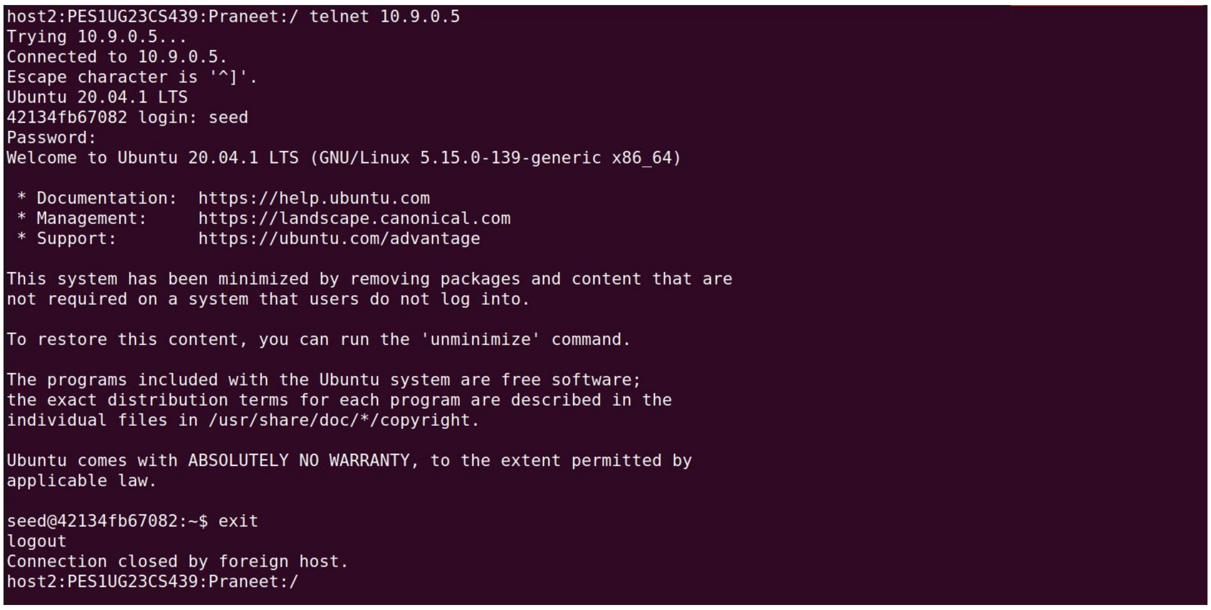
Host 2:



```
host2:PES1UG23CS439:Praneet:/telnet 192.168.60.7
Trying 192.168.60.7...
Connected to 192.168.60.7.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
c52b4b21df52 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-139-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: Wed Oct 22 13:50:11 UTC 2025 from host2-192.168.60.6.net-192.168.60.0 on pts/2
seed@c52b4b21df52:~$ 
```

This, also from Host2 (.6), shows another **successful** telnet, this time to 192.168.60.7. Just like the connection previously, this is purely internal traffic that bypasses the router's firewall rules entirely.

Host 2:



```
host2:PES1UG23CS439:Praneet:/ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
42134fb67082 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-139-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
applicable law.

seed@42134fb67082:~$ exit
logout
Connection closed by foreign host.
host2:PES1UG23CS439:Praneet:/
```

This Host2 screenshot is the most important result. It shows a **successful** telnet to an external server (10.9.0.5). This is the key difference from Task 2.C and works because our new stateful rules (Rule 2 and Rule 3) explicitly allow NEW connections out and ESTABLISHED traffic back in.

Purpose of each rule:

.....-i eth0 -d 192.168.60.5 --dport 23 --syn ... --ctstate NEW -j ACCEPT

- **Purpose:** This is the "server" rule. It's an opening that only allows **NEW** connections (specifically SYN packets) coming from the outside (-i eth0) that are destined for the one specific server 192.168.60.5 on the telnet port.

.....-i eth1 -p tcp --syn ... --ctstate NEW -j ACCEPT

- **Purpose:** This is the "client" rule. It's very broad where it allows any **NEW** TCP connection (--syn packet) that comes from the inside (-i eth1) to go out.

.....-p tcp -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

- **Purpose:** This is the work of a stateful firewall. This single rule tells the router to ACCEPT all TCP packets that are part of a connection it already knows about (one that is **ESTABLISHED**) or **RELATED** to one. It handles all the reply traffic for both Rule #1 and Rule #2, without having to write specific "reply" rules.

.....-p tcp -j DROP

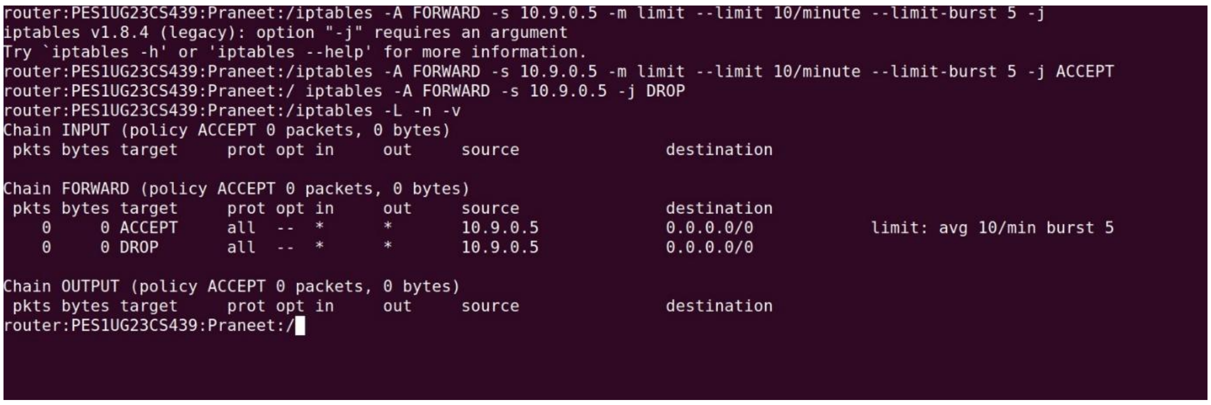
- **Purpose:** This is a cleanup rule. It says that any TCP packet that isn't a NEW packet matching Rule 1 or 2, and isn't part of an ESTABLISHED connection and it should be dropped. This blocks malformed packets or other invalid traffic.

In the **stateless firewall**, the system was “dumb” where it treated every packet separately. To allow something like a Telnet session we had to manually add two rules: one for outgoing requests and another for incoming replies. This approach was since allowing all replies could open security holes.

The **stateful firewall** is “smart.” It uses **connection tracking (conntrack)** to remember active connections. We only need to specify which new connections can start and the firewall automatically allows all related reply traffic using the --ctstate RELATED,ESTABLISHED rule. This makes the setup both simpler and more secure.

Task 4:

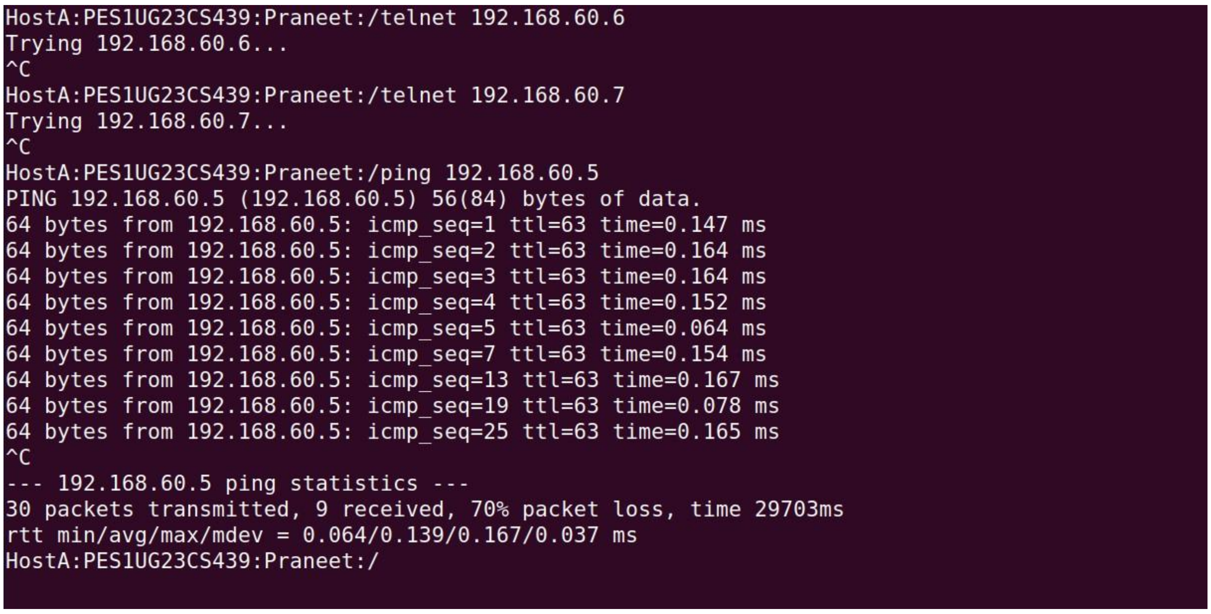
Router:



```
router:PES1UG23CS439:Praneet:/iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute --limit-burst 5 -j
iptables v1.8.4 (legacy): option "-j" requires an argument
Try `iptables -h' or 'iptables --help' for more information.
router:PES1UG23CS439:Praneet:/iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute --limit-burst 5 -j ACCEPT
router:PES1UG23CS439:Praneet:/ iptables -A FORWARD -s 10.9.0.5 -j DROP
router:PES1UG23CS439:Praneet:/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     all  --  *      *       10.9.0.5             0.0.0.0/0            limit: avg 10/min burst 5
    0     0 DROP       all  --  *      *       10.9.0.5             0.0.0.0/0

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
router:PES1UG23CS439:Praneet:/
```

This seed-router terminal shows the successful setup of a rate-limiting policy. Two FORWARD rules are added for traffic from 10.9.0.5 where the first rule ACCEPTs traffic but limits it to 10 packets/minute with a 5-packet burst and the second rule DROPs any traffic from that source that exceeds the limit.

Host a:



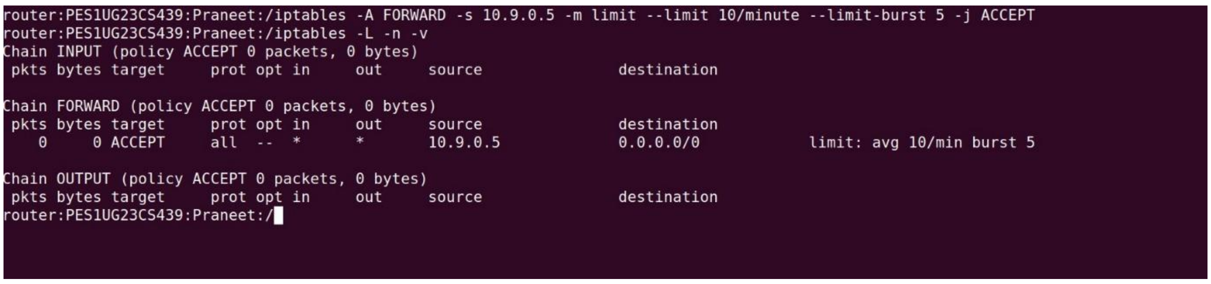
```
HostA:PES1UG23CS439:Praneet:/telnet 192.168.60.6
Trying 192.168.60.6...
^C
HostA:PES1UG23CS439:Praneet:/telnet 192.168.60.7
Trying 192.168.60.7...
^C
HostA:PES1UG23CS439:Praneet:/ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.147 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.164 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.164 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=0.064 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.154 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.167 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.078 ms
64 bytes from 192.168.60.5: icmp_seq=25 ttl=63 time=0.165 ms
^C
--- 192.168.60.5 ping statistics ---
30 packets transmitted, 9 received, 70% packet loss, time 29703ms
rtt min/avg/max/mdev = 0.064/0.139/0.167/0.037 ms
HostA:PES1UG23CS439:Praneet:/
```

This HostA snip shows the effect of the rate limit. When pinging an internal server, the connection is not fully blocked but it experiences 70% packet loss. This proves the firewall is throttling the connection allowing the initial burst and a slow trickle of packets through while dropping the majority.

Purpose of each rule:

1. **iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute --limit-burst 5 -j ACCEPT**
   - **Purpose:** This rule defines the allowance. It tells the router to use the limit module for any packet from 10.9.0.5. If the packet is within the defined limit (the first 5, or within the 10/minute rate), this rule matches and the packet is accepted. If the packet is over the limit, this rule simply "does not match" and the packet moves to the next rule.
2. **iptables -A FORWARD -s 10.9.0.5 -j DROP**
   - **Purpose:** This rule puts the limit. It acts as the catching for any packet from 10.9.0.5 that was not accepted by the first rule. Without this rule, any packet exceeding the limit would simply fall through and be allowed by the chain's default ACCEPT policy making the limit pointless.

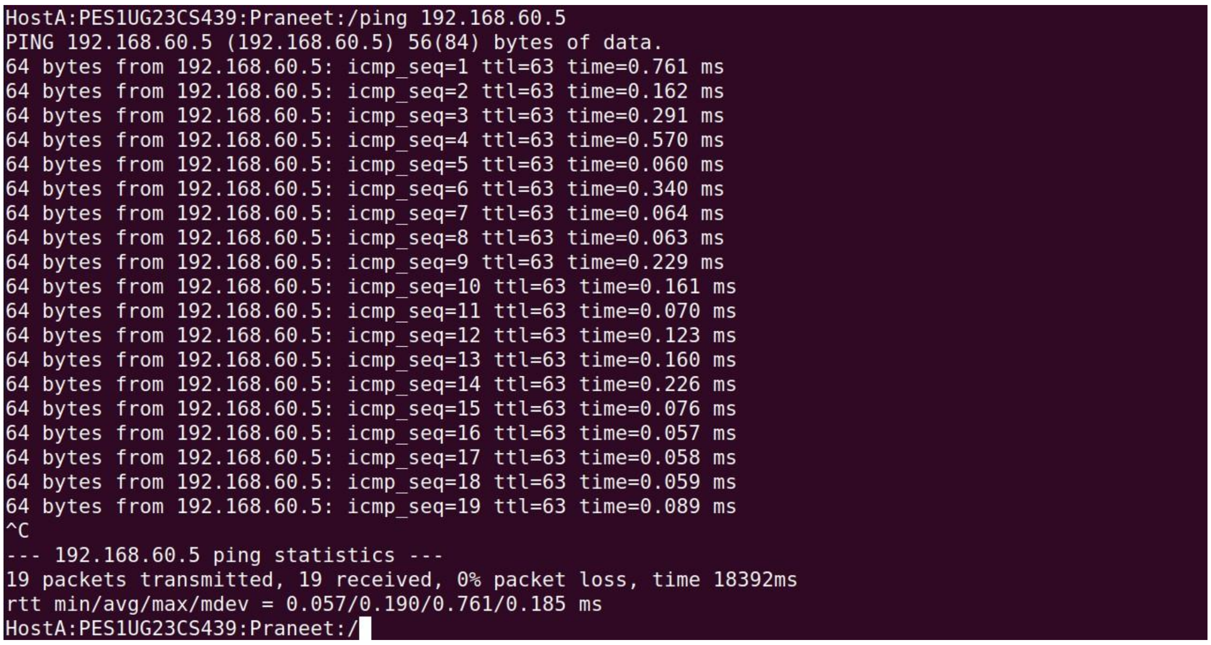
Router:



```
router:PES1UG23CS439:Praneet:/iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute --limit-burst 5 -j ACCEPT
router:PES1UG23CS439:Praneet:/iptables -L -n -v
hain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

hain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       10.9.0.5             0.0.0.0/0            limit: avg 10/min burst 5

hain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
router:PES1UG23CS439:Praneet:/
```

This screenshot shows an incomplete rate-limiting setup. The iptables -L -n -v output confirms that only the ACCEPT rule with the limit module has been added. The essential DROP rule is missing, and the chain's default policy remains ACCEPT.

Host A:



```
HostA:PES1UG23CS439:Praneet:/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=0.761 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.162 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.291 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.570 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.060 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.340 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.064 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.063 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.229 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.161 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.070 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.123 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.160 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.226 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.076 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.058 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.089 ms
^C
--- 192.168.60.5 ping statistics ---
19 packets transmitted, 19 received, 0% packet loss, time 18392ms
rtt min/avg/max/mdev = 0.057/0.190/0.761/0.185 ms
HostA:PES1UG23CS439:Praneet:/
```

This snip from HostA shows the result of the incomplete firewall. The ping to 192.168.60.5 is **100% successful** with 0% packet loss. This proves that without a corresponding DROP rule, ACCEPT rule is ineffective and fails to manage/control the traffic.

Purpose of each rule:

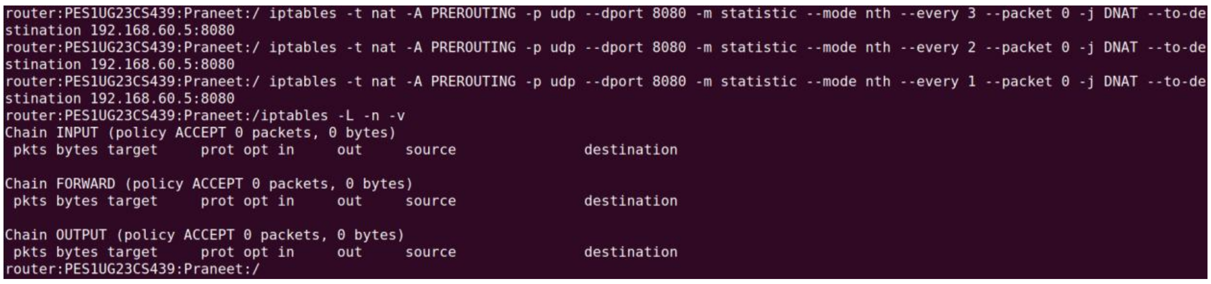
iptables -A FORWARD -s 10.9.0.5 -m limit --limit 10/minute --limit-burst 5 -j ACCEPT

- **Purpose:** This rule's purpose is to ACCEPT packets from 10.9.0.5 as long as they are within the specified rate. It defines an allowing limit. It does not have any built-in action for packets that exceed this allowance it simply doesn't match them.

Task 5:

Using round robin:

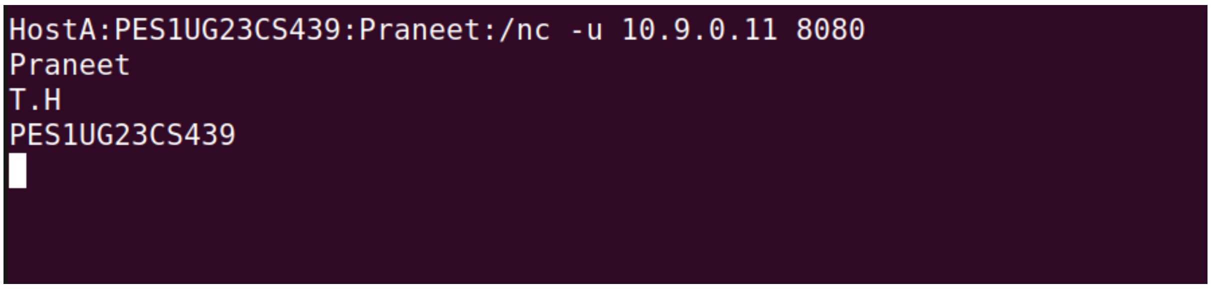
Router:



```
router:PES1UG23CS439:Praneet:/ iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 3 --packet 0 -j DNAT --to-de
stination 192.168.60.5:8080
router:PES1UG23CS439:Praneet:/ iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 2 --packet 0 -j DNAT --to-de
stination 192.168.60.5:8080
router:PES1UG23CS439:Praneet:/ iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 1 --packet 0 -j DNAT --to-de
stination 192.168.60.5:8080
router:PES1UG23CS439:Praneet:/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
router:PES1UG23CS439:Praneet:/
```

This seed-router shows the iptables rules being set up. Adding three DNAT rules to the PREROUTING chain of the nat table. These rules use the statistic module (--mode nth) to create the round-robin logic, assigning the first packet of every three to .5, the second to .6, and the third to .7.

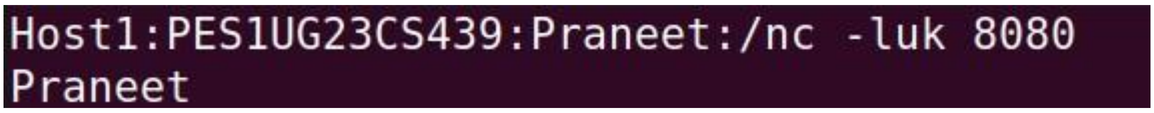
Host a:



```
HostA:PES1UG23CS439:Praneet:/nc -u 10.9.0.11 8080
Praneet
T.H
PES1UG23CS439
█
```

This screenshot shows the HostA client sending the test traffic. Using netcat in UDP mode (-u) to send three separate lines of text ("Praneet", "T.H", "PES1UG23CS439") as three separate packets. Crucially, sending them to the router's IP (10.9.0.11, not to the servers directly allowing the router to intercept and load-balance them.

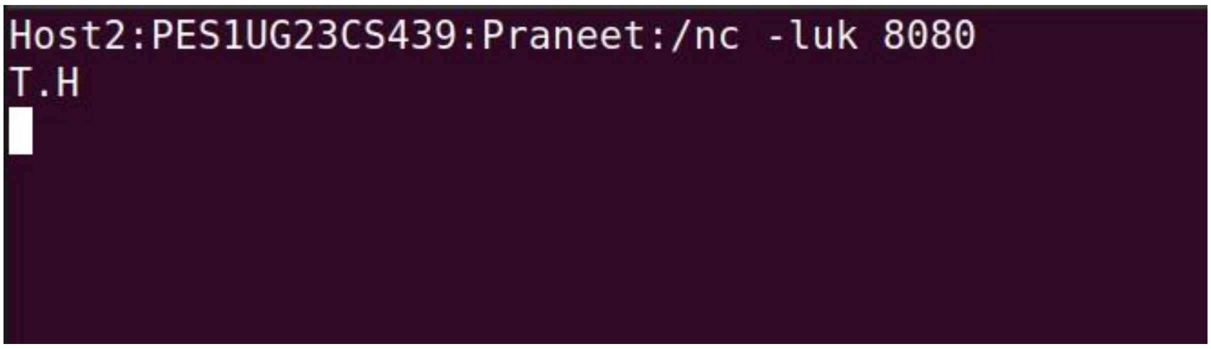
Host 1:



```
Host1:PES1UG23CS439:Praneet:/nc -luk 8080
Praneet
```

This is the netcat listener on **Host1** (192.168.60.5). It has successfully received the first packet.This confirms that the first iptables rule (--every 3) correctly matched this packet and redirected it to this first server.

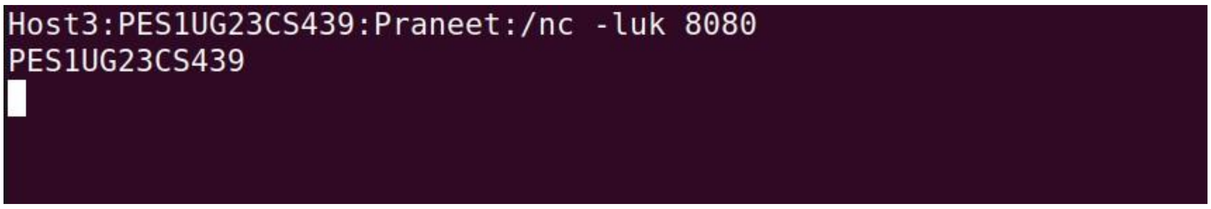
Host 2:



```
Host2:PES1UG23CS439:Praneet:/nc -luk 8080
T.H
█
```

This is the listener on **Host2** (192.168.60.6). It has received the second packet, "T.H". This demonstrates that the packet failed the first rule (it wasn't the 1st, 4th, etc.) and was correctly caught by the second rule (--every 2) which redirected it to this second server.

Host 3:



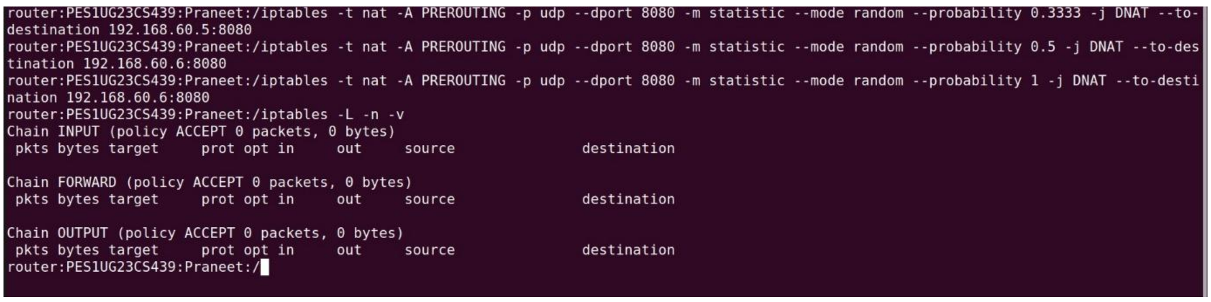
```
Host3:PES1UG23CS439:Praneet:/nc -luk 8080
PES1UG23CS439
```

This is the listener on **Host3** (192.168.60.7). It has received the third packet, "PES1UG23CS439". This packet was passed by the first two rules and correctly caught by the final "catch-all" (--every 1)completing the round-robin distribution.

<u>Note: The firewall rule observations are written together with the snip observation.</u>

Using random mode:

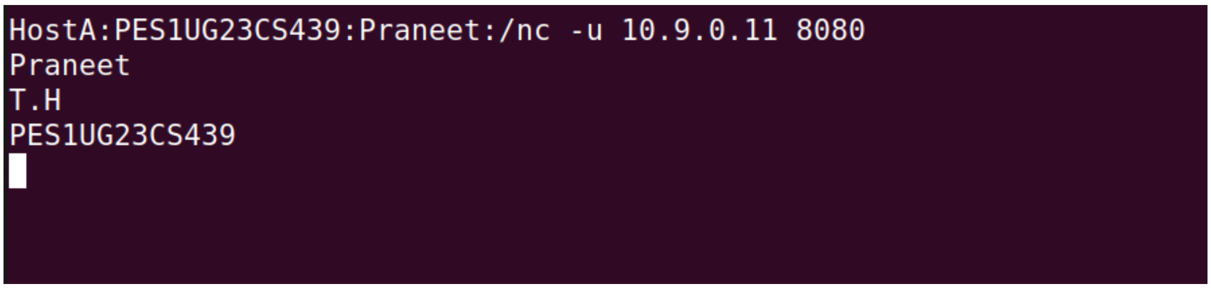
Router:



```
router:PES1UG23CS439:Praneet:/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
router:PES1UG23CS439:Praneet:/iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 0.5 -j DNAT --to-des
tination 192.168.60.6:8080
router:PES1UG23CS439:Praneet:/iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 1 -j DNAT --to-desti
nation 192.168.60.6:8080
router:PES1UG23CS439:Praneet:/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
router:PES1UG23CS439:Praneet:/
```

This screenshot shows the iptables rules for the probabilistic load balancer. It adds three DNAT rules to the nat table's PREROUTING chain, using statistic --mode random with cascading probabilities (0.3333, 0.5, and 1) to create an approximately equal 1/3-1/3-1/3 distribution of traffic to three different internal servers.

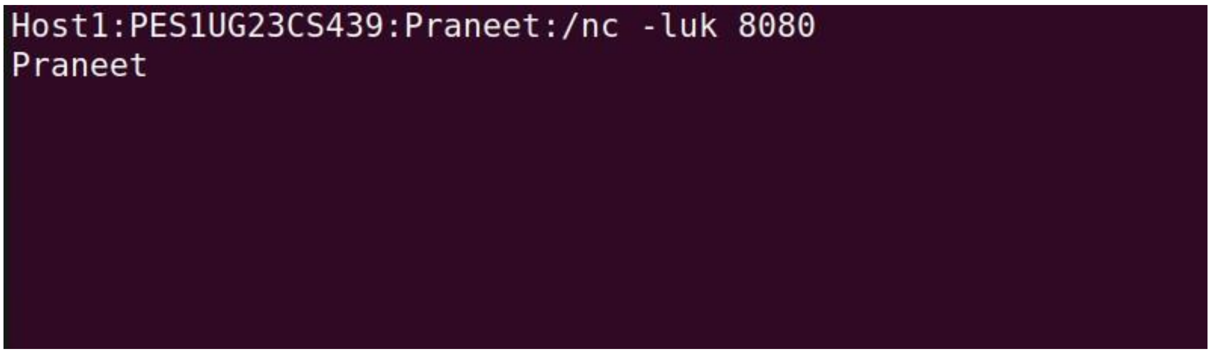
Host A:



```
HostA:PES1UG23CS439:Praneet:/nc -u 10.9.0.11 8080
Praneet
T.H
PES1UG23CS439
```

It shows the user sending three distinct UDP packets to the router's IP (10.9.0.11). The router will intercept these packets and redirect them according to its load-balancing rules.

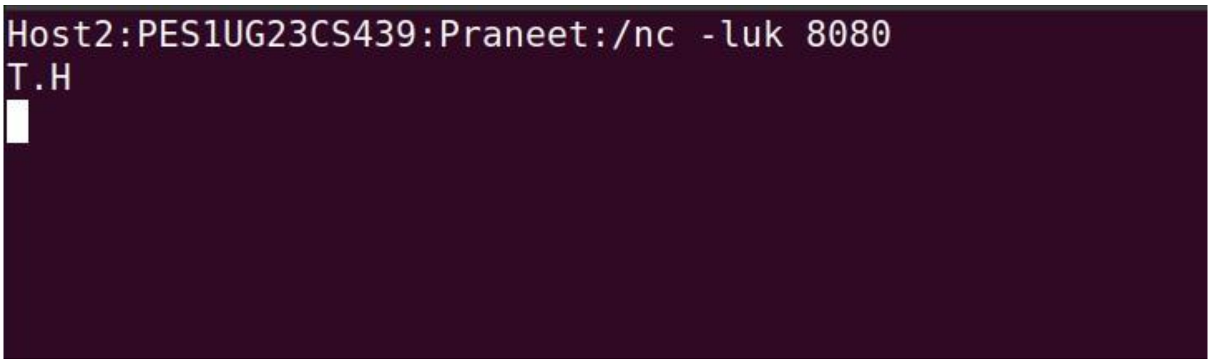
Host 1:



```
Host1:PES1UG23CS439:Praneet:/nc -luk 8080
Praneet
```

It has received the first packet. This indicates the packet hit the 33.33% chance defined in the first iptables rule and was correctly redirected to this server.

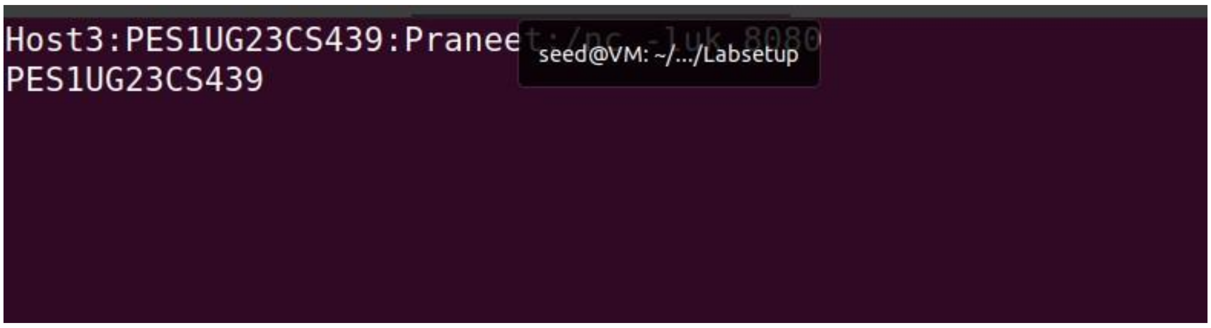
Host 2:



```
Host2:PES1UG23CS439:Praneet:/nc -luk 8080
T.H
```

It received the second packet. This means the packet missed the first rule's 33.33% chance but hit the 50% chance on the second rule and was sent here.

Host 3:



```
Host3:PES1UG23CS439:Praneet:/nc -luk 8080
PES1UG23CS439
```

It received the third. This packet must have missed both of the first two probabilistic rules and was therefore caught by the final "catch-all" rule (--probability 1) and sent to this server completing the distribution.

Purpose of each rule:

1)… --probability 0.3333 … --to-destination 192.168.60.5

- This rule catches an incoming packet with a **33.33% chance** and sends it to Host1.

2) ... --probability 0.5 ... --to-destination 192.168.60.6

- This rule only sees packets that **failed** the first rule. It has a **50% chance** of catching those packets. 50% of 66.66% is 33.33% so this rule also has a 1-in-3 chance overall of sending a packet to Host2.

3) ... --probability 1 ... --to-destination 192.168.60.7

- This rule sees any packet that **failed both** of the first two rules (the final 1/3 of the total). It has a **100% chance** (--probability 1) of catching them sending all remaining traffic to Host3.