COMPUTER NETWORK SECURITY

LAB-7

FIREWALL EXPLORATION LAB

NAME: VISHWAS M

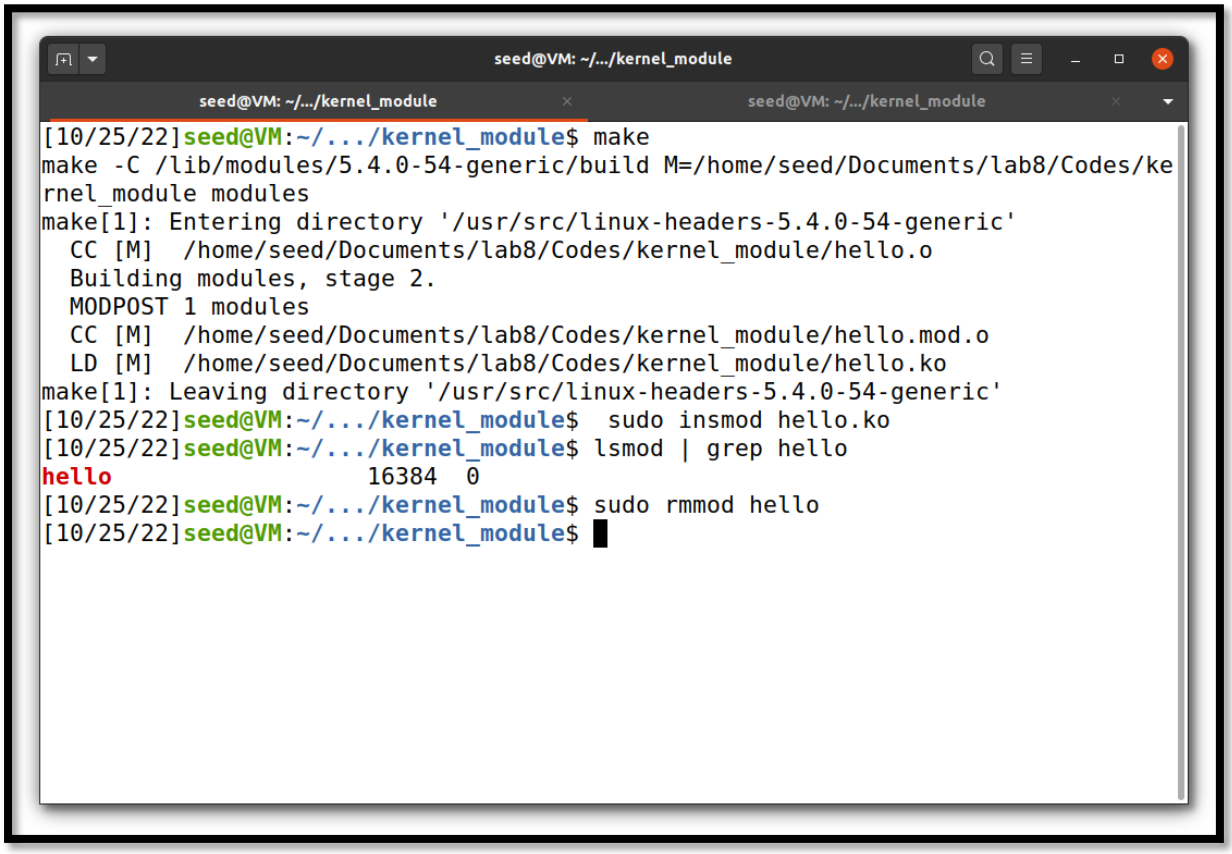
SRN: PES2UG20CS390

SEC: F

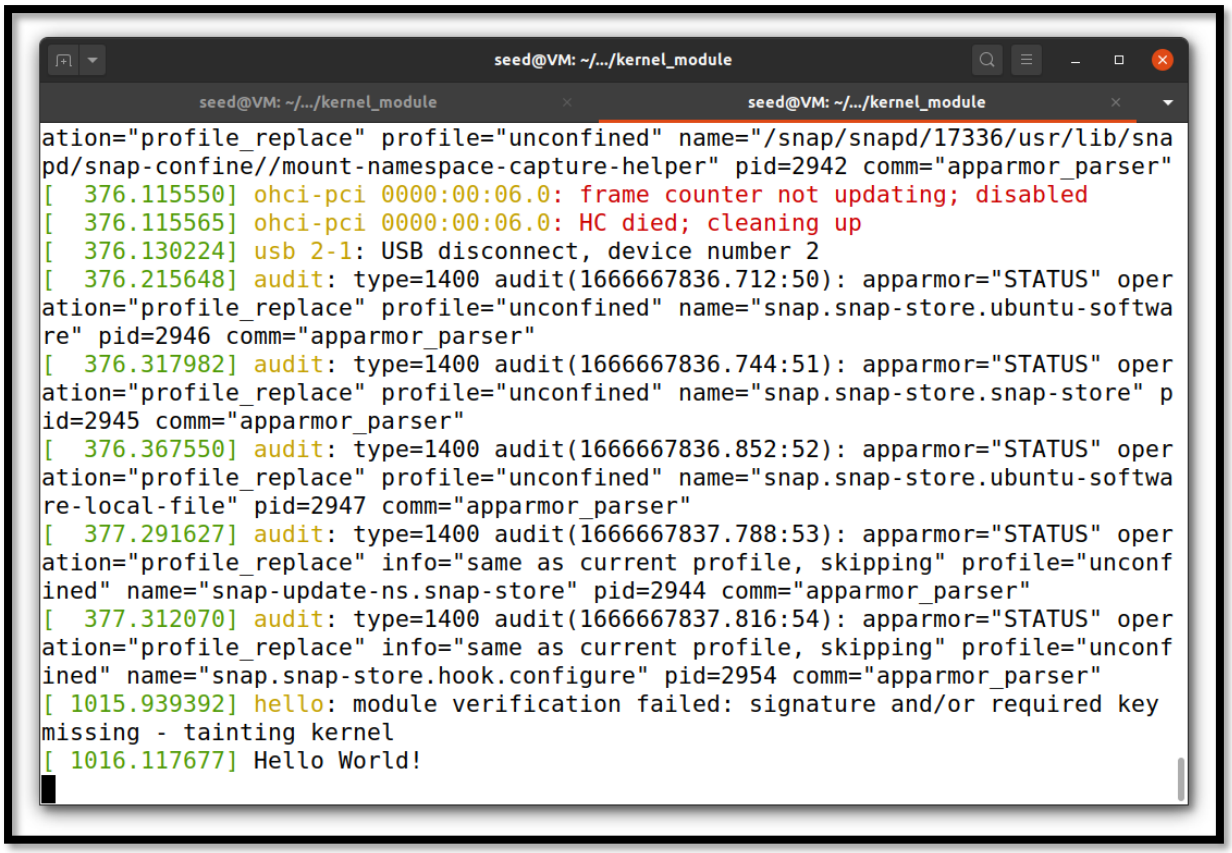
DATE:26/10/2022

Task 1:

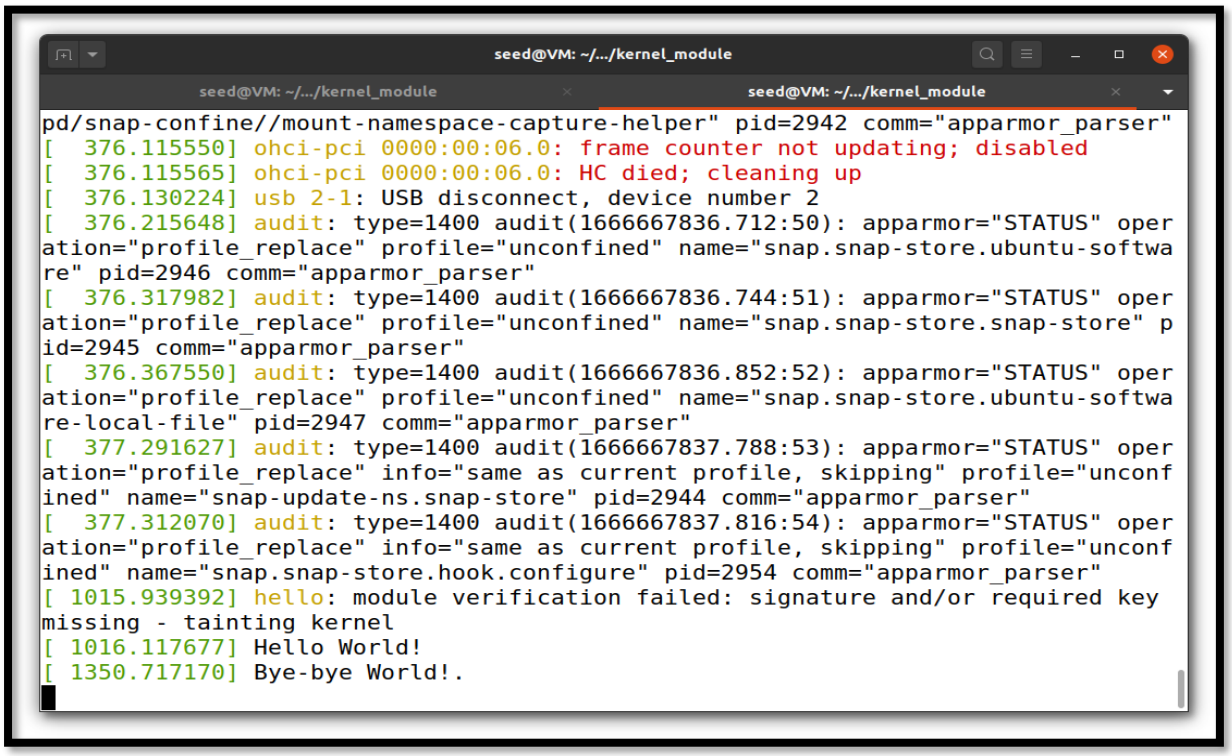
Task 1.A: Implement a simple kernel module



In this task we are trying to make a new kernel module with the help of make command. Then we are making use of insmod command to insert the kernel module into the kernel. Then we use the grep command to view all the files which are uploaded now.



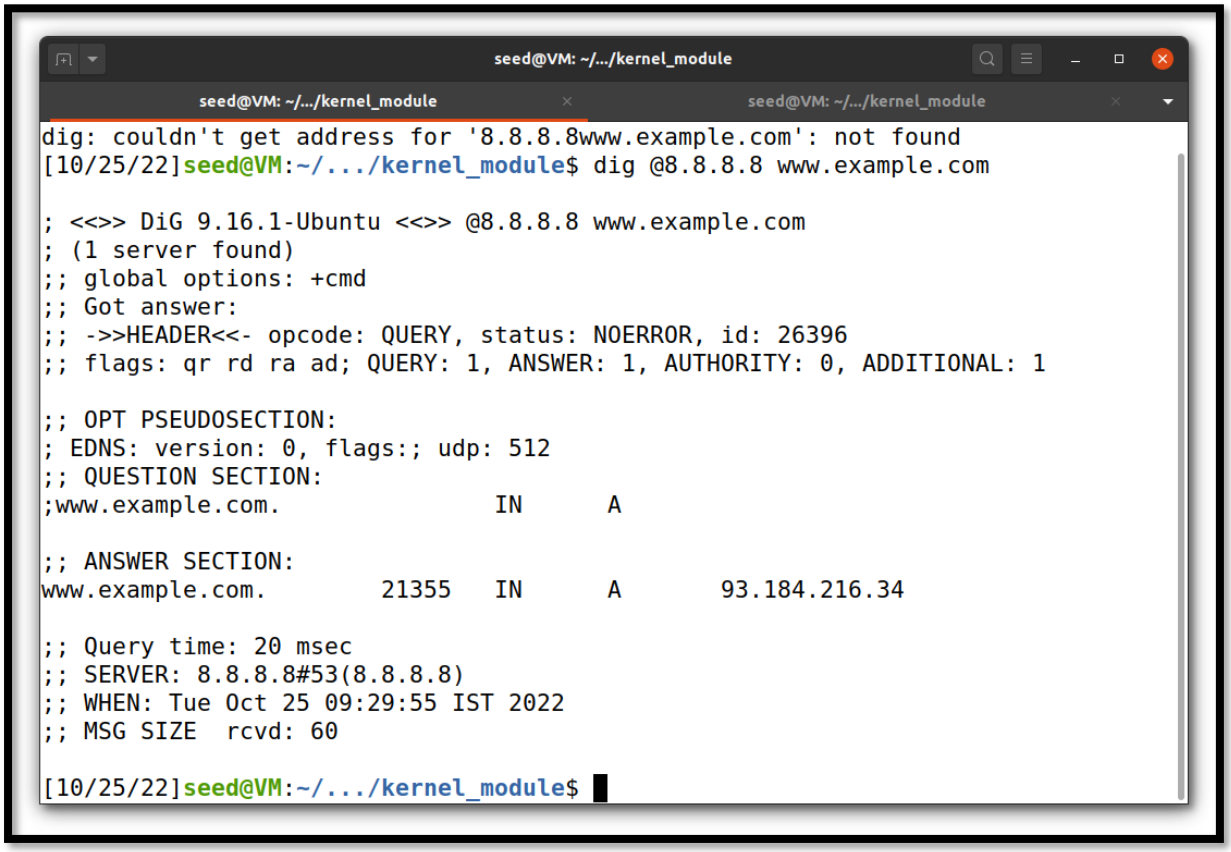
The messages are not printed on the terminal wen we run this module. It will get stored in /var/log/syslog . So we use the dmesg command to view the “Hello Word” message in the host terminal.



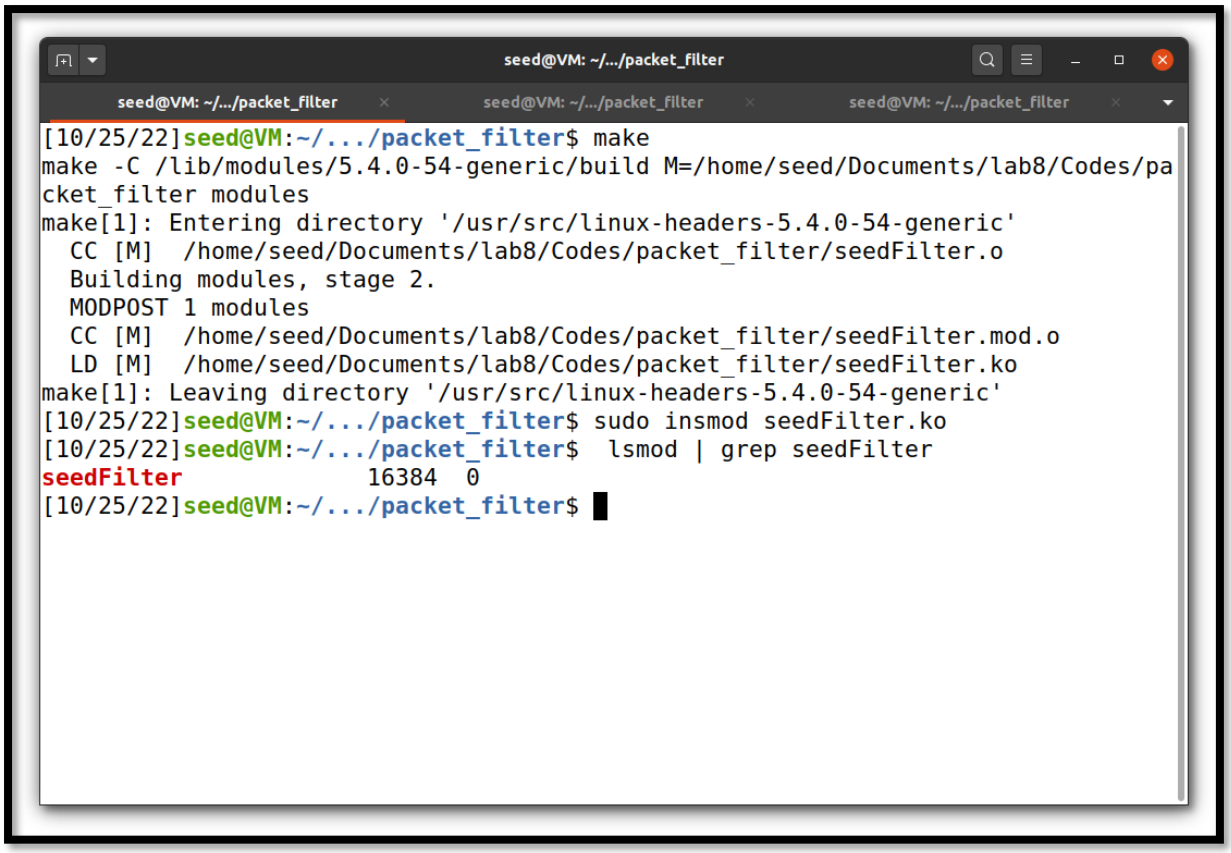
After executing the rmmod command to delete the kernel module that we just uploaded, the message “Bye-bye World” will get printed in the terminal.

Task 1.B: Implement a single firewall using Netfilter

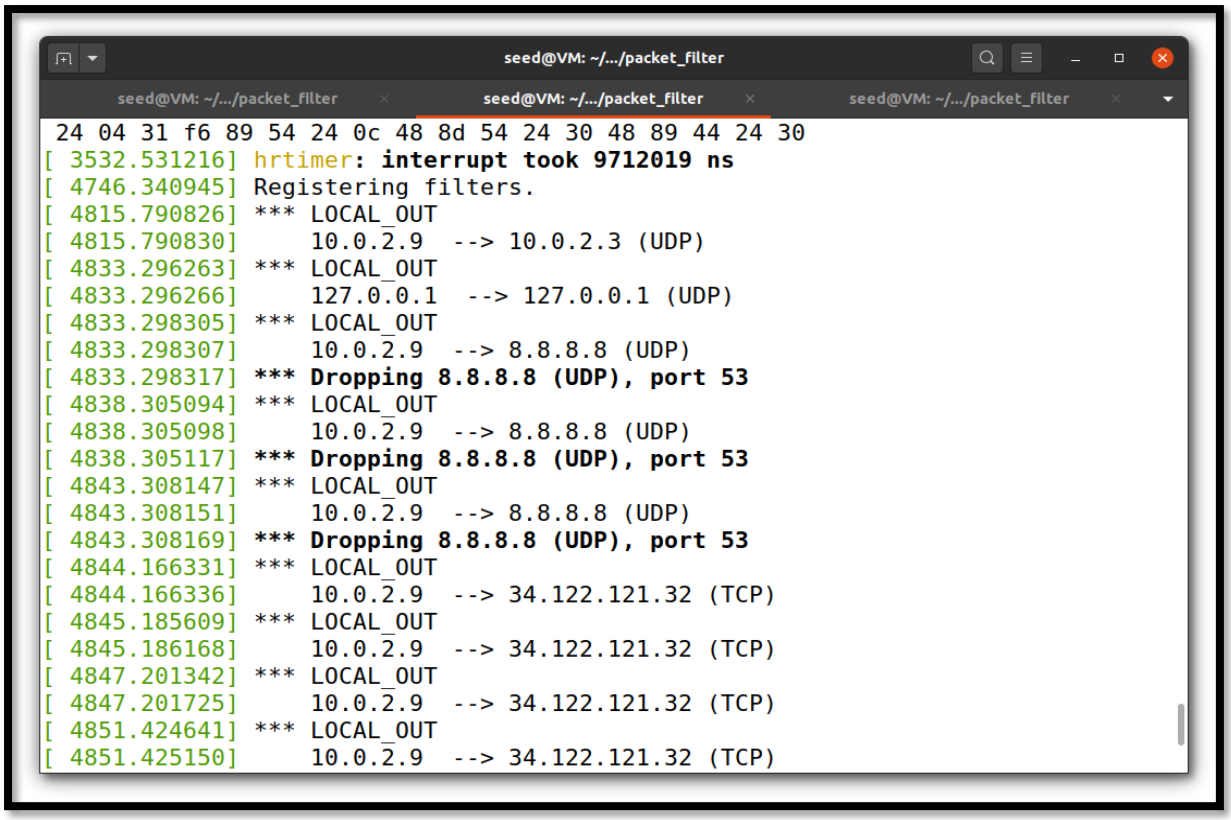
1)



First we are trying to see whether the server 8.8.8.8 is reachable or not. By the above screenshot we can say that the server is reachable. And this is before manipulating the firewall.



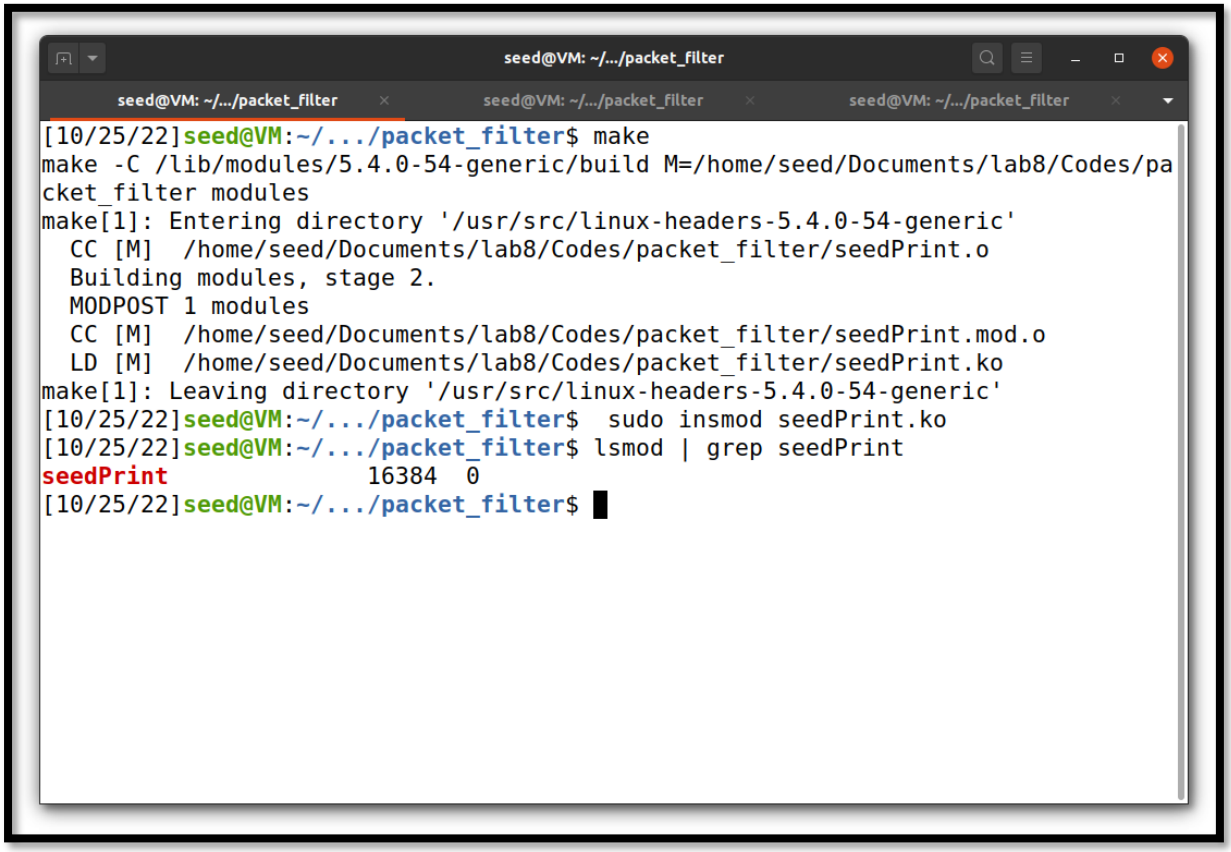
I am uploading a kernel module which is used to block all the requests going to 8.8.8.8 server.



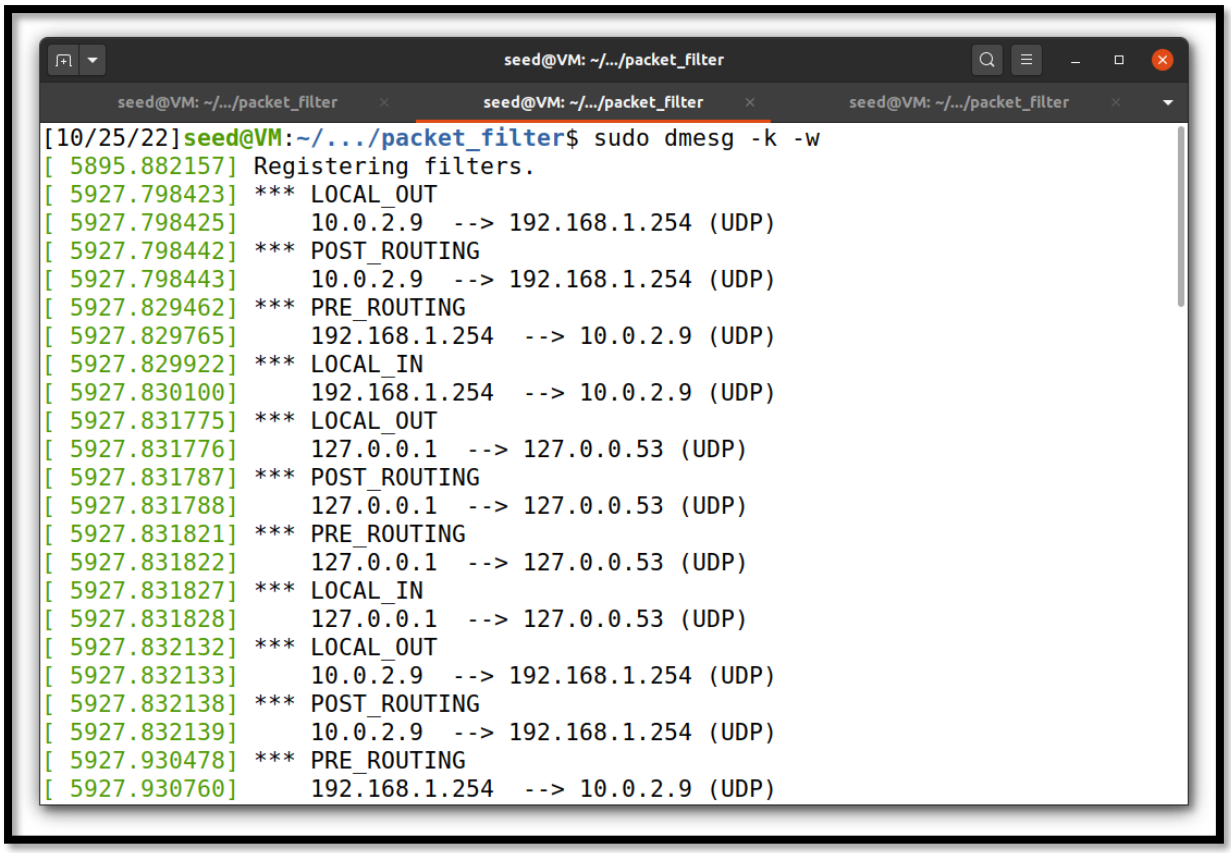
Here we can see that the packets are getting dropped and the requests are not reaching the server 8.8.8.8.

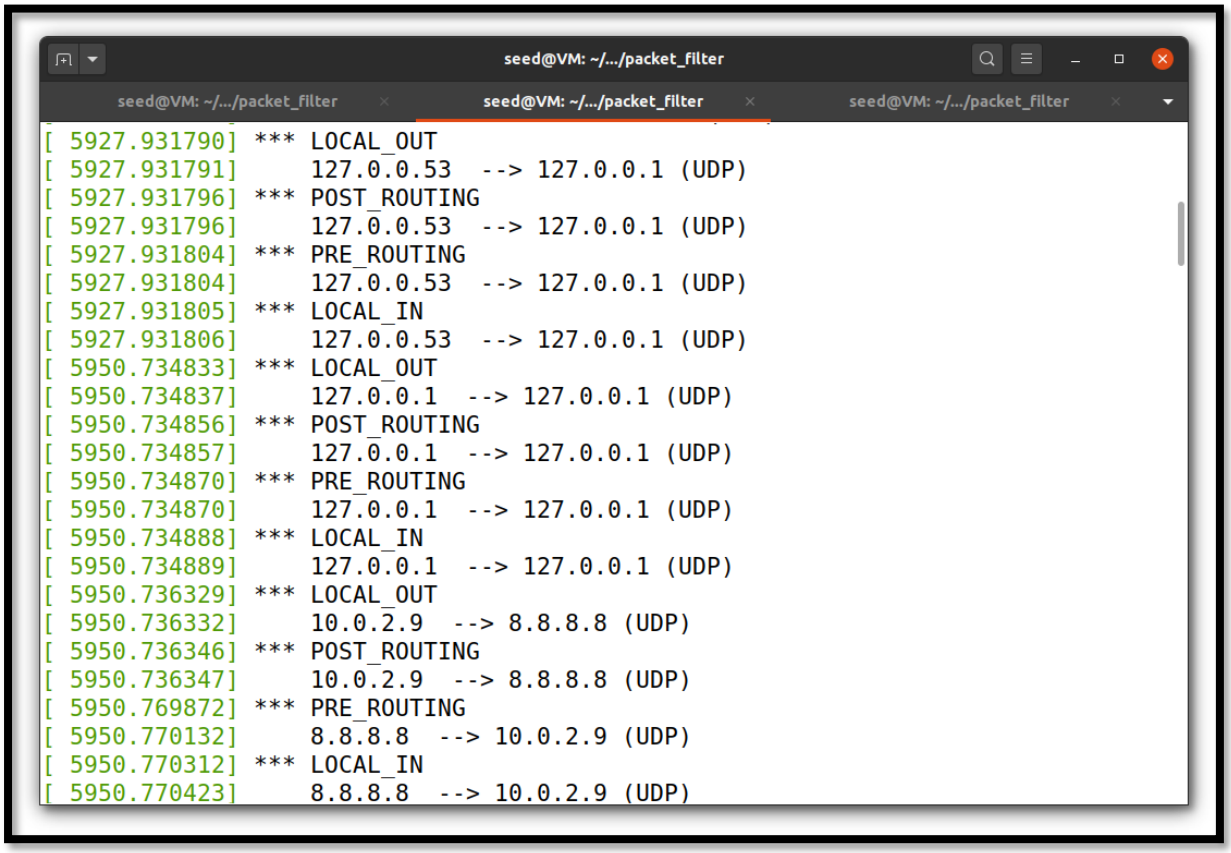
When we execute the dig command again, we can see that the server could not be reached because the firewall has blocked all the requests going to that particular server.

2)



In this subtask we are uploading a module which prints the netfilter hooks.


seed@VM: ~/.../packet_filter
seed@VM: ~/.../packet_filter
seed@VM: ~/.../packet_filter
seed@VM: ~/.../packet_filter
[10/25/22]seed@VM:~/.../packet_filter$ sudo dmesg -k -w
[5895.882157] Registering filters.
[5927.798423] *** LOCAL_OUT
[5927.798425] 10.0.2.9 --> 192.168.1.254 (UDP)
[5927.798442] *** POST_ROUTING
[5927.798443] 10.0.2.9 --> 192.168.1.254 (UDP)
[5927.829462] *** PRE_ROUTING
[5927.829765] 192.168.1.254 --> 10.0.2.9 (UDP)
[5927.829922] *** LOCAL_IN
[5927.830100] 192.168.1.254 --> 10.0.2.9 (UDP)
[5927.831775] *** LOCAL_OUT
[5927.831776] 127.0.0.1 --> 127.0.0.53 (UDP)
[5927.831787] *** POST_ROUTING
[5927.831788] 127.0.0.1 --> 127.0.0.53 (UDP)
[5927.831821] *** PRE_ROUTING
[5927.831822] 127.0.0.1 --> 127.0.0.53 (UDP)
[5927.831827] *** LOCAL_IN
[5927.831828] 127.0.0.1 --> 127.0.0.53 (UDP)
[5927.832132] *** LOCAL_OUT
[5927.832133] 10.0.2.9 --> 192.168.1.254 (UDP)
[5927.832138] *** POST_ROUTING
[5927.832139] 10.0.2.9 --> 192.168.1.254 (UDP)
[5927.930478] *** PRE_ROUTING
[5927.930760] 192.168.1.254 --> 10.0.2.9 (UDP)


seed@VM: ~/.../packet_filter
seed@VM: ~/.../packet_filter
seed@VM: ~/.../packet_filter
seed@VM: ~/.../packet_filter
[5927.931790] *** LOCAL_OUT
[5927.931791] 127.0.0.53 --> 127.0.0.1 (UDP)
[5927.931796] *** POST_ROUTING
[5927.931796] 127.0.0.53 --> 127.0.0.1 (UDP)
[5927.931804] *** PRE_ROUTING
[5927.931804] 127.0.0.53 --> 127.0.0.1 (UDP)
[5927.931805] *** LOCAL_IN
[5927.931806] 127.0.0.53 --> 127.0.0.1 (UDP)
[5950.734833] *** LOCAL_OUT
[5950.734837] 127.0.0.1 --> 127.0.0.1 (UDP)
[5950.734856] *** POST_ROUTING
[5950.734857] 127.0.0.1 --> 127.0.0.1 (UDP)
[5950.734870] *** PRE_ROUTING
[5950.734870] 127.0.0.1 --> 127.0.0.1 (UDP)
[5950.734888] *** LOCAL_IN
[5950.734889] 127.0.0.1 --> 127.0.0.1 (UDP)
[5950.736329] *** LOCAL_OUT
[5950.736332] 10.0.2.9 --> 8.8.8.8 (UDP)
[5950.736346] *** POST_ROUTING
[5950.736347] 10.0.2.9 --> 8.8.8.8 (UDP)
[5950.769872] *** PRE_ROUTING
[5950.770132] 8.8.8.8 --> 10.0.2.9 (UDP)
[5950.770312] *** LOCAL_IN
[5950.770423] 8.8.8.8 --> 10.0.2.9 (UDP)

Here are the macros that are invoked during the experiment. The macros used are:

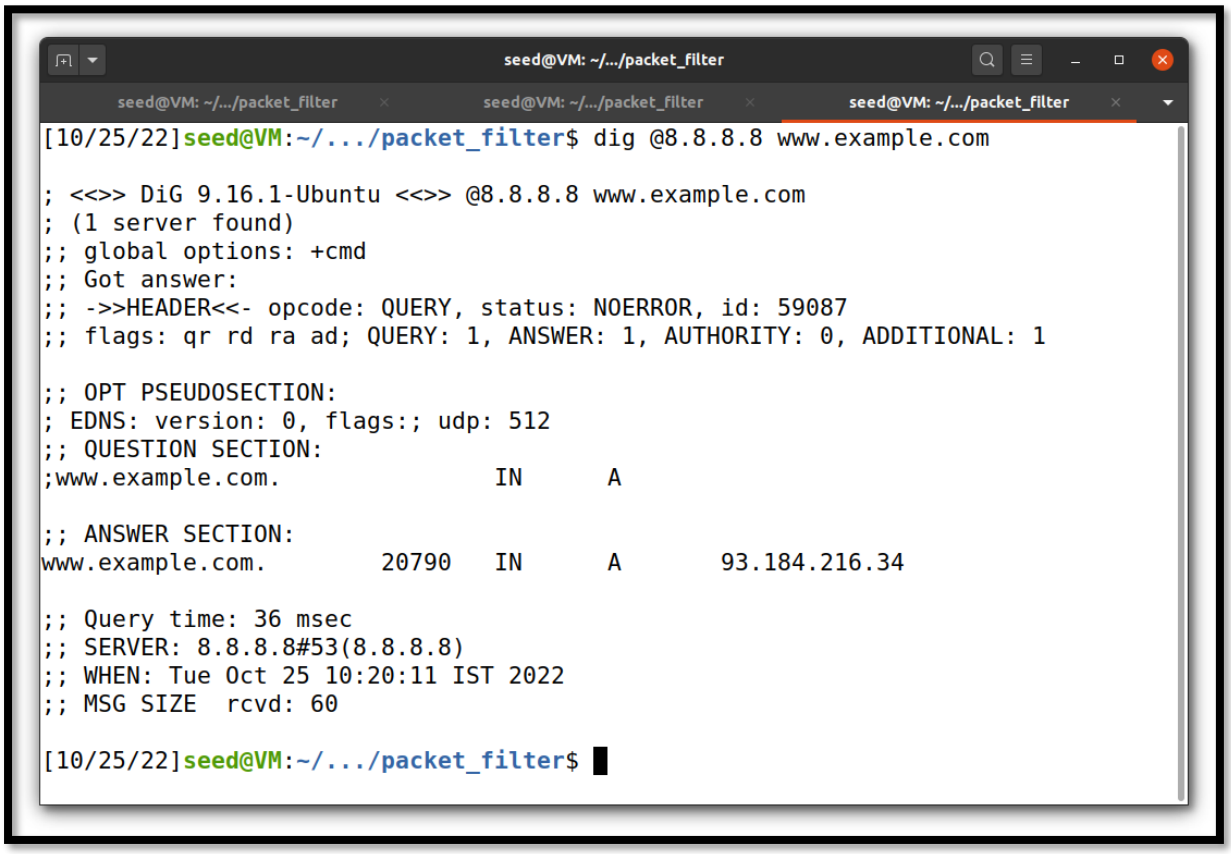
NF_INET_PRE_ROUTING

NF_INET_LOCAL_IN

NF_INET_FORWARD

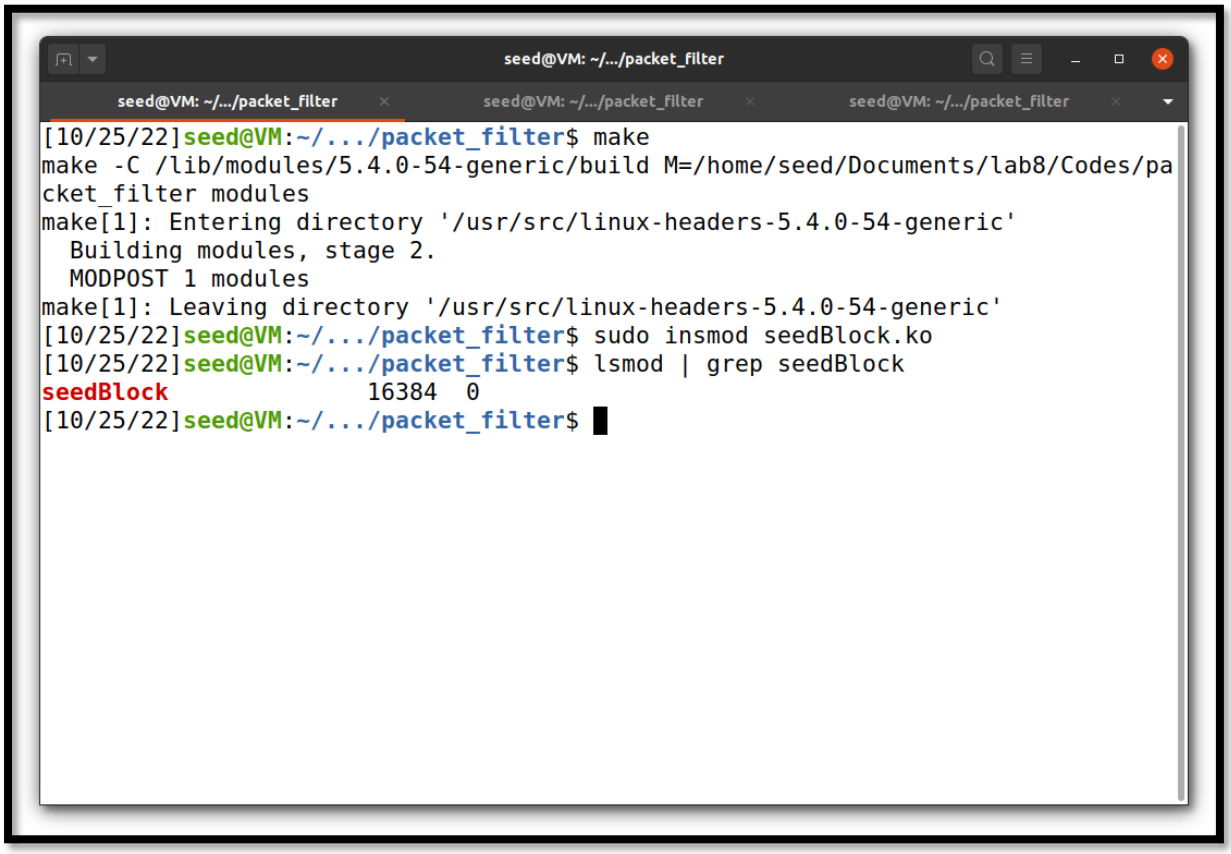
NF_INET_LOCAL_OUT

NF_INET_POST_ROUTING



We can reach the server 8.8.8.8 when we use the dig command and simultaneously observe how the hook functions work as shown in the previous screenshots.

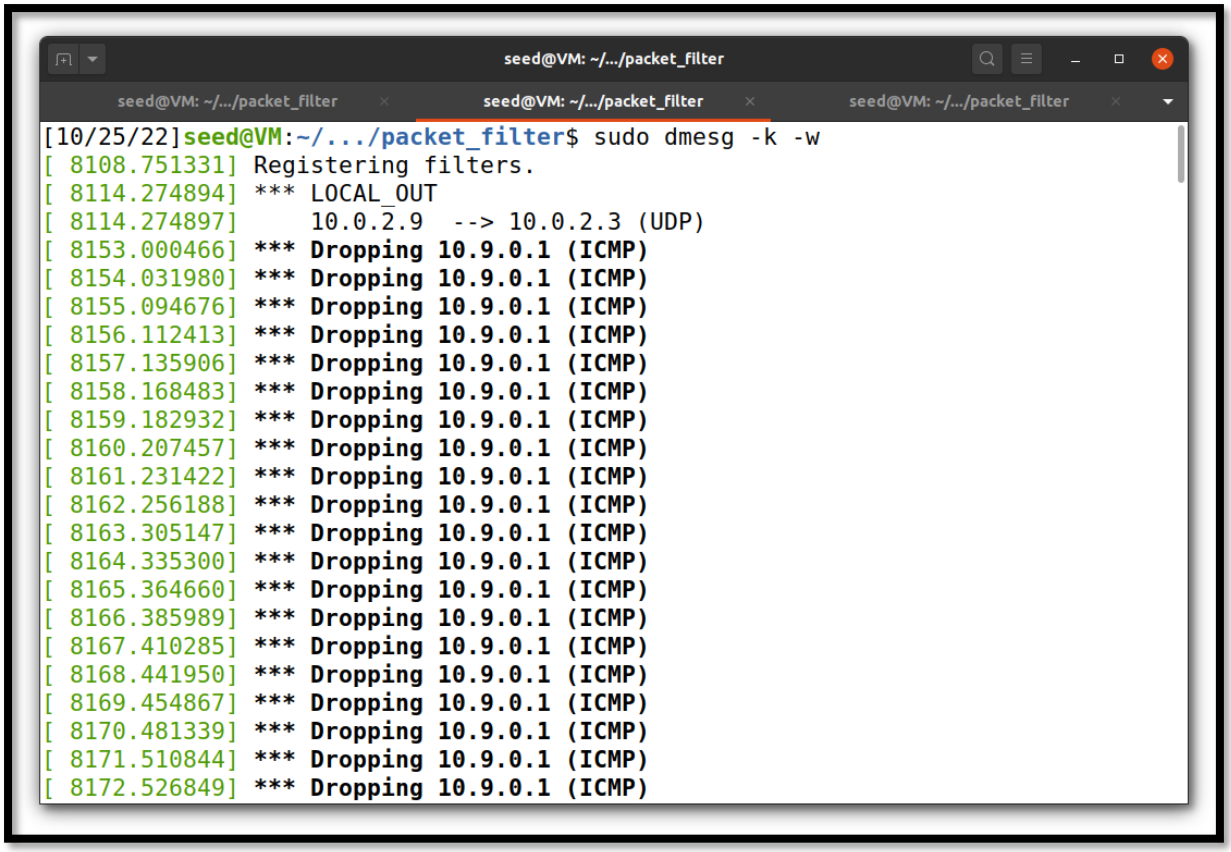
3)

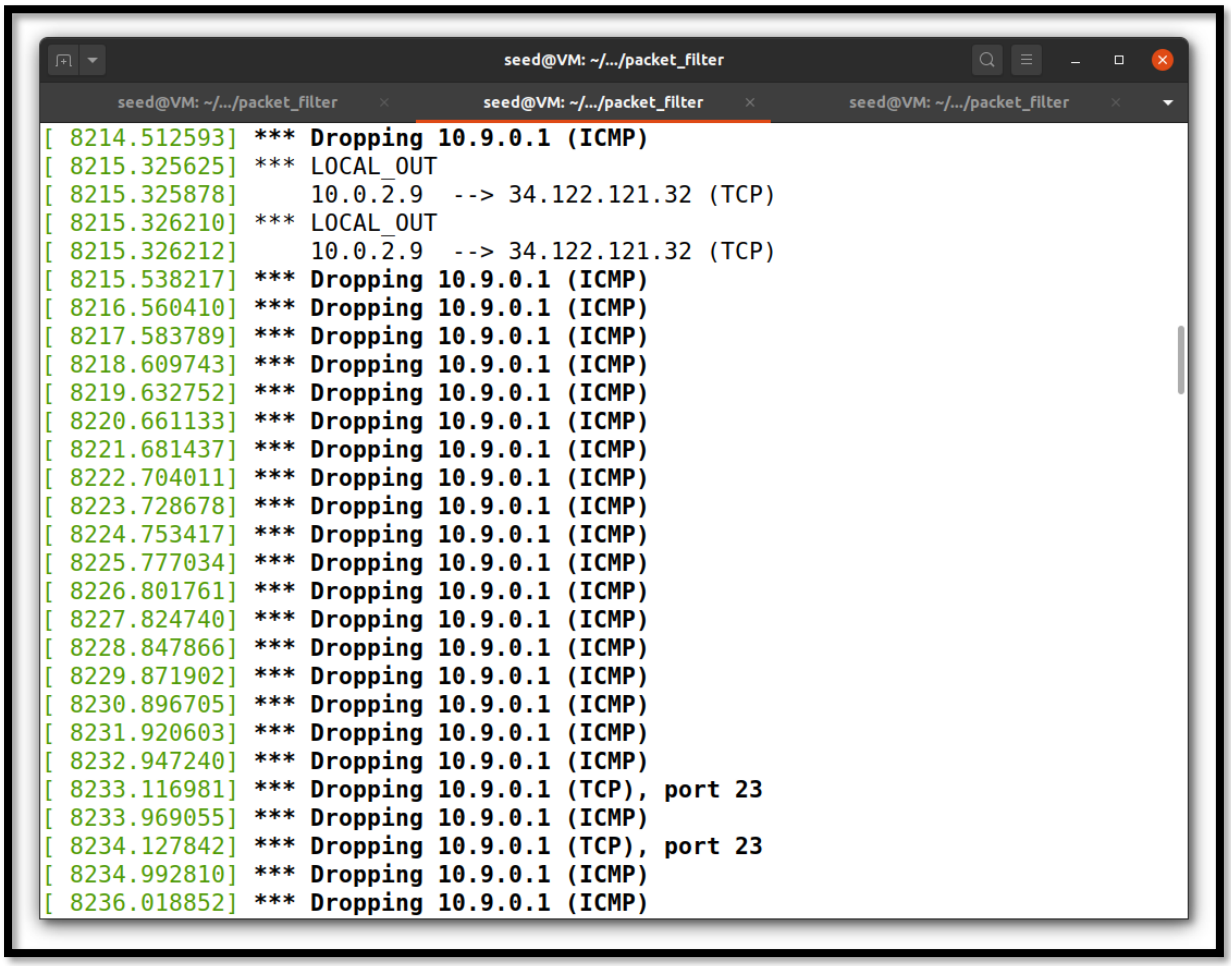


In this task we are trying to add 2 more hooks to achieve:

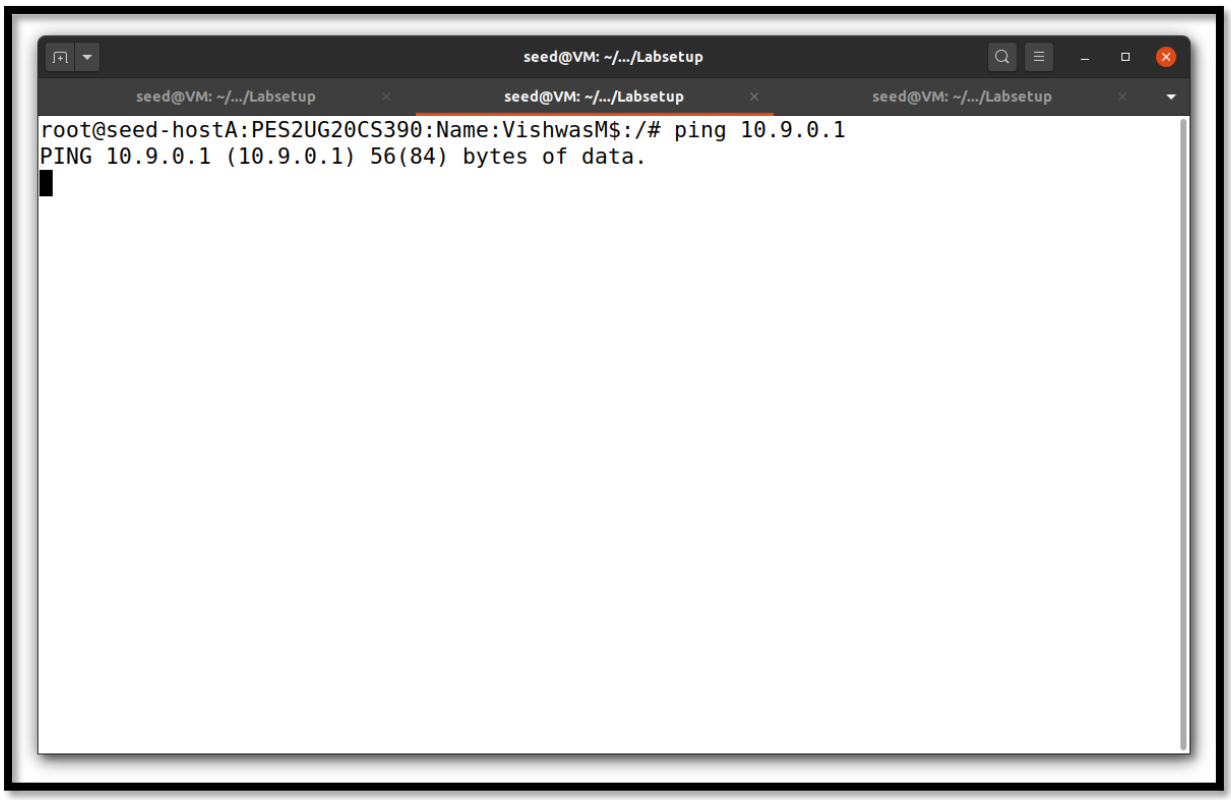
a) Prevent other computers to ping the VM
b) Prevent other computers from telnetting into the VM

We are adding a kernel module to perform the above two asks. We are adding the above module by using the insmod command.

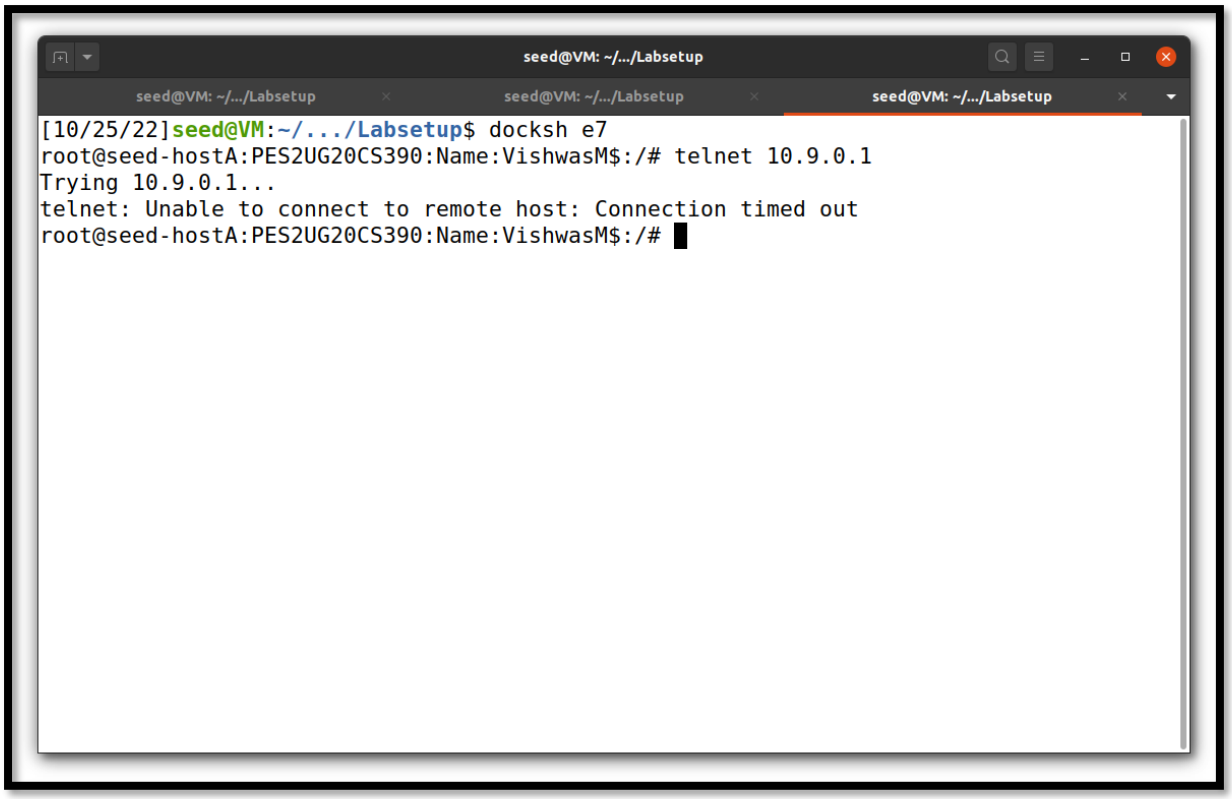

seed@VM: ~/.../packet_filter
seed@VM: ~/.../packet_filter
seed@VM: ~/.../packet_filter
seed@VM: ~/.../packet_filter
[10/25/22]seed@VM:~/.../packet_filter$ sudo dmesg -k -w
[8108.751331] Registering filters.
[8114.274894] *** LOCAL_OUT
[8114.274897] 10.0.2.9 --> 10.0.2.3 (UDP)
[8153.000466] *** Dropping 10.9.0.1 (ICMP)
[8154.031980] *** Dropping 10.9.0.1 (ICMP)
[8155.094676] *** Dropping 10.9.0.1 (ICMP)
[8156.112413] *** Dropping 10.9.0.1 (ICMP)
[8157.135906] *** Dropping 10.9.0.1 (ICMP)
[8158.168483] *** Dropping 10.9.0.1 (ICMP)
[8159.182932] *** Dropping 10.9.0.1 (ICMP)
[8160.207457] *** Dropping 10.9.0.1 (ICMP)
[8161.231422] *** Dropping 10.9.0.1 (ICMP)
[8162.256188] *** Dropping 10.9.0.1 (ICMP)
[8163.305147] *** Dropping 10.9.0.1 (ICMP)
[8164.335300] *** Dropping 10.9.0.1 (ICMP)
[8165.364660] *** Dropping 10.9.0.1 (ICMP)
[8166.385989] *** Dropping 10.9.0.1 (ICMP)
[8167.410285] *** Dropping 10.9.0.1 (ICMP)
[8168.441950] *** Dropping 10.9.0.1 (ICMP)
[8169.454867] *** Dropping 10.9.0.1 (ICMP)
[8170.481339] *** Dropping 10.9.0.1 (ICMP)
[8171.510844] *** Dropping 10.9.0.1 (ICMP)
[8172.526849] *** Dropping 10.9.0.1 (ICMP)


seed@VM: ~/.../packet_filter
seed@VM: ~/.../packet_filter
seed@VM: ~/.../packet_filter
seed@VM: ~/.../packet_filter
[8214.512593] *** Dropping 10.9.0.1 (ICMP)
[8215.325625] *** LOCAL_OUT
[8215.325878] 10.0.2.9 --> 34.122.121.32 (TCP)
[8215.326210] *** LOCAL_OUT
[8215.326212] 10.0.2.9 --> 34.122.121.32 (TCP)
[8215.538217] *** Dropping 10.9.0.1 (ICMP)
[8216.560410] *** Dropping 10.9.0.1 (ICMP)
[8217.583789] *** Dropping 10.9.0.1 (ICMP)
[8218.609743] *** Dropping 10.9.0.1 (ICMP)
[8219.632752] *** Dropping 10.9.0.1 (ICMP)
[8220.661133] *** Dropping 10.9.0.1 (ICMP)
[8221.681437] *** Dropping 10.9.0.1 (ICMP)
[8222.704011] *** Dropping 10.9.0.1 (ICMP)
[8223.728678] *** Dropping 10.9.0.1 (ICMP)
[8224.753417] *** Dropping 10.9.0.1 (ICMP)
[8225.777034] *** Dropping 10.9.0.1 (ICMP)
[8226.801761] *** Dropping 10.9.0.1 (ICMP)
[8227.824740] *** Dropping 10.9.0.1 (ICMP)
[8228.847866] *** Dropping 10.9.0.1 (ICMP)
[8229.871902] *** Dropping 10.9.0.1 (ICMP)
[8230.896705] *** Dropping 10.9.0.1 (ICMP)
[8231.920603] *** Dropping 10.9.0.1 (ICMP)
[8232.947240] *** Dropping 10.9.0.1 (ICMP)
[8233.116981] *** Dropping 10.9.0.1 (TCP), port 23
[8233.969055] *** Dropping 10.9.0.1 (ICMP)
[8234.127842] *** Dropping 10.9.0.1 (TCP), port 23
[8234.992810] *** Dropping 10.9.0.1 (ICMP)
[8236.018852] *** Dropping 10.9.0.1 (ICMP)

We can see that the ICMP packets as well as the TCP packets are getting dropped at the firewall because of the hooks that we had defined earlier. As the packets are getting dropped the host VM(10.9.0.1) is unable to reach from the container host(10.9.0.5). VM(10.9.0.1) is unable to reach from the container host(10.9.0.5).



Here we can see that the ping command is not working as the packets are getting dropped at the firewall.

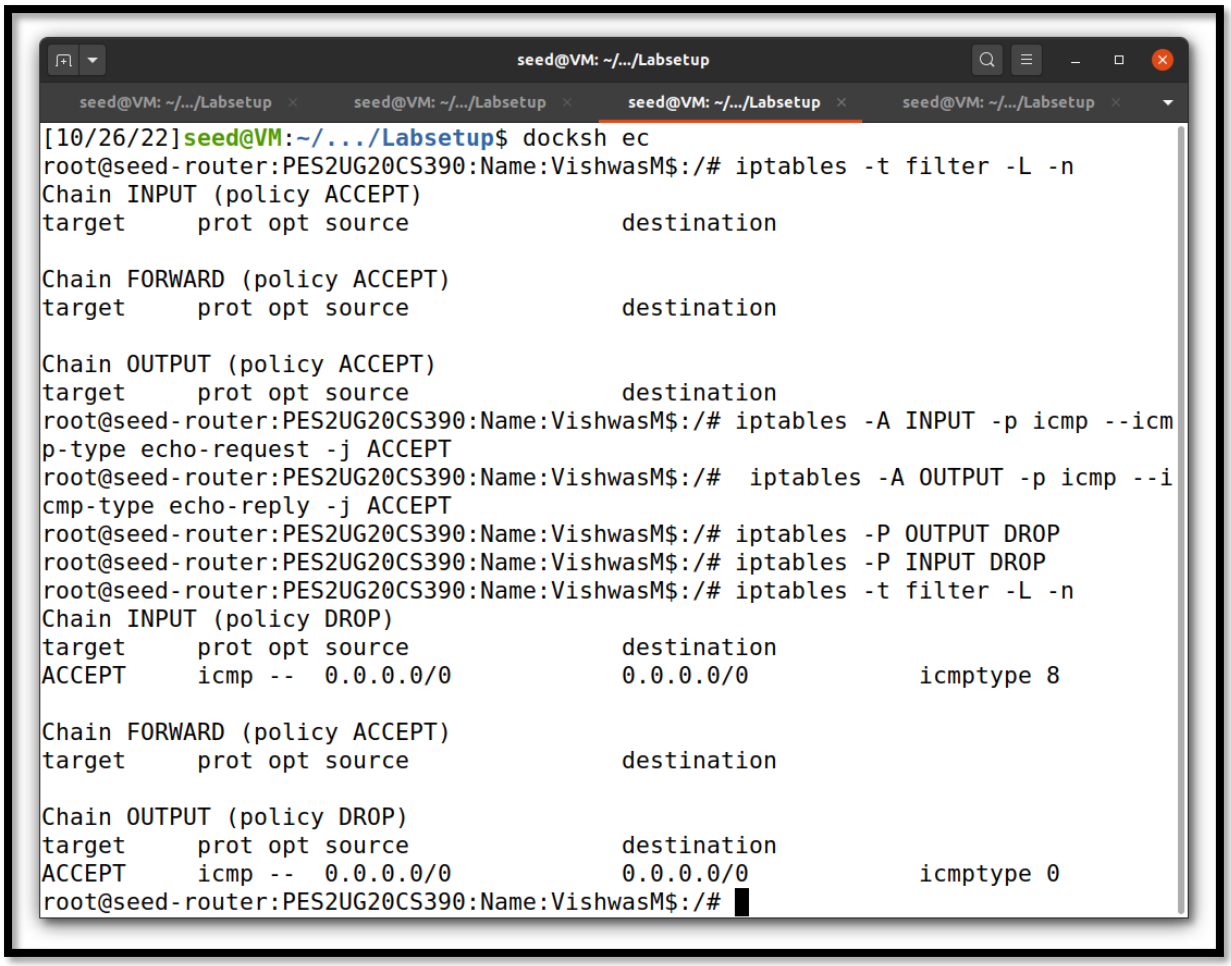


Here we can see that telnet is also not working as the TCP packets are also getting dropped at the firewall.

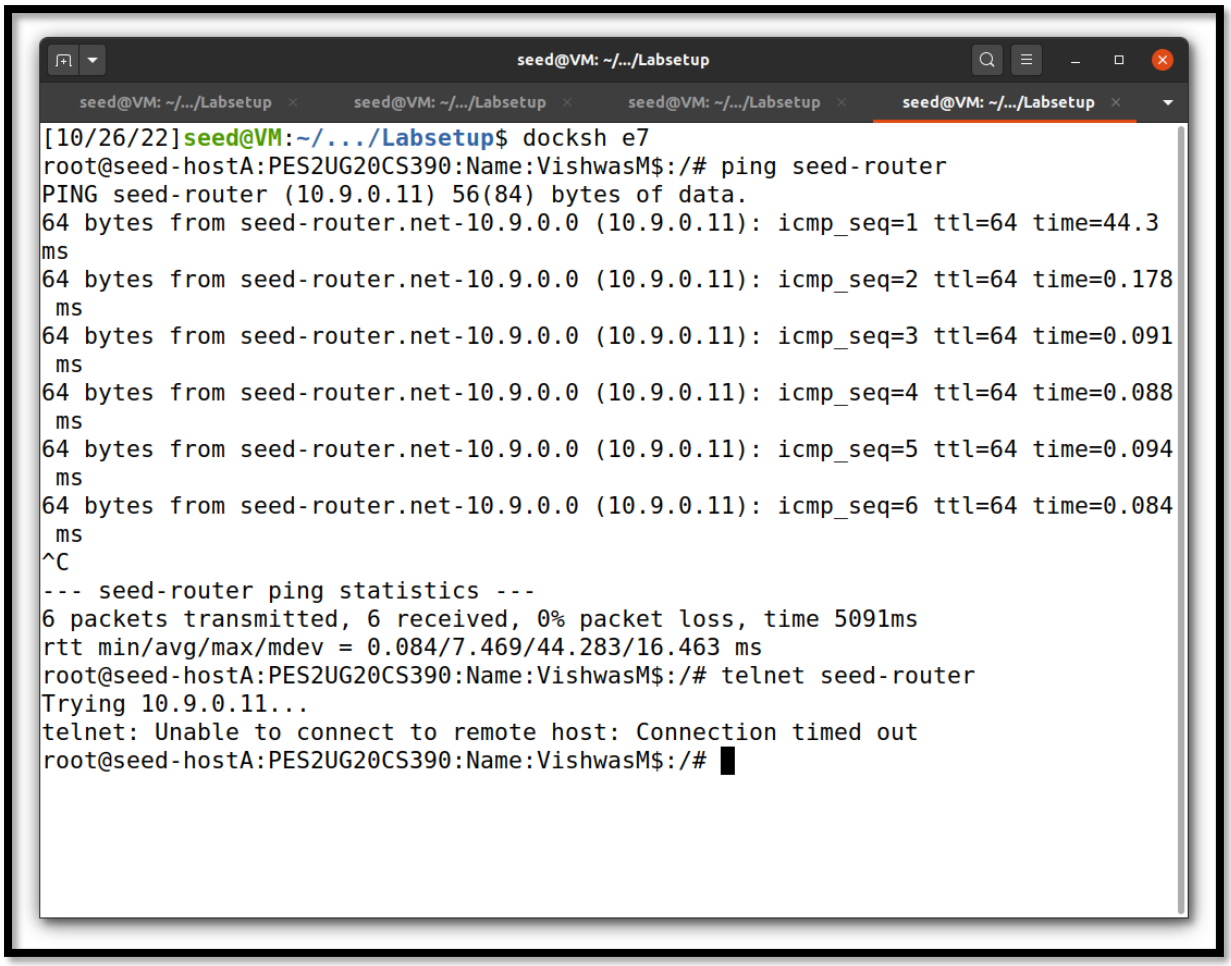
Task 2: Experimenting with Stateless Firewall Rules

Task 2.A: Protecting the router

In this task, we will set some rules to prevent outside machines from accessing the router machines, except ping.



As we can see in the above screenshot, we have applied some rules to the firewall. Here we are only accepting the ping requests and ping replies in the firewall and dropping all other packets.



Here we can see that when we ping the router from the host machine, the packets are getting routed properly. But when we do telnet, the packets won't get routed as the firewall blocks and drops the packets. Therefore connection will not take place between router and host.

1) Can you ping the router?

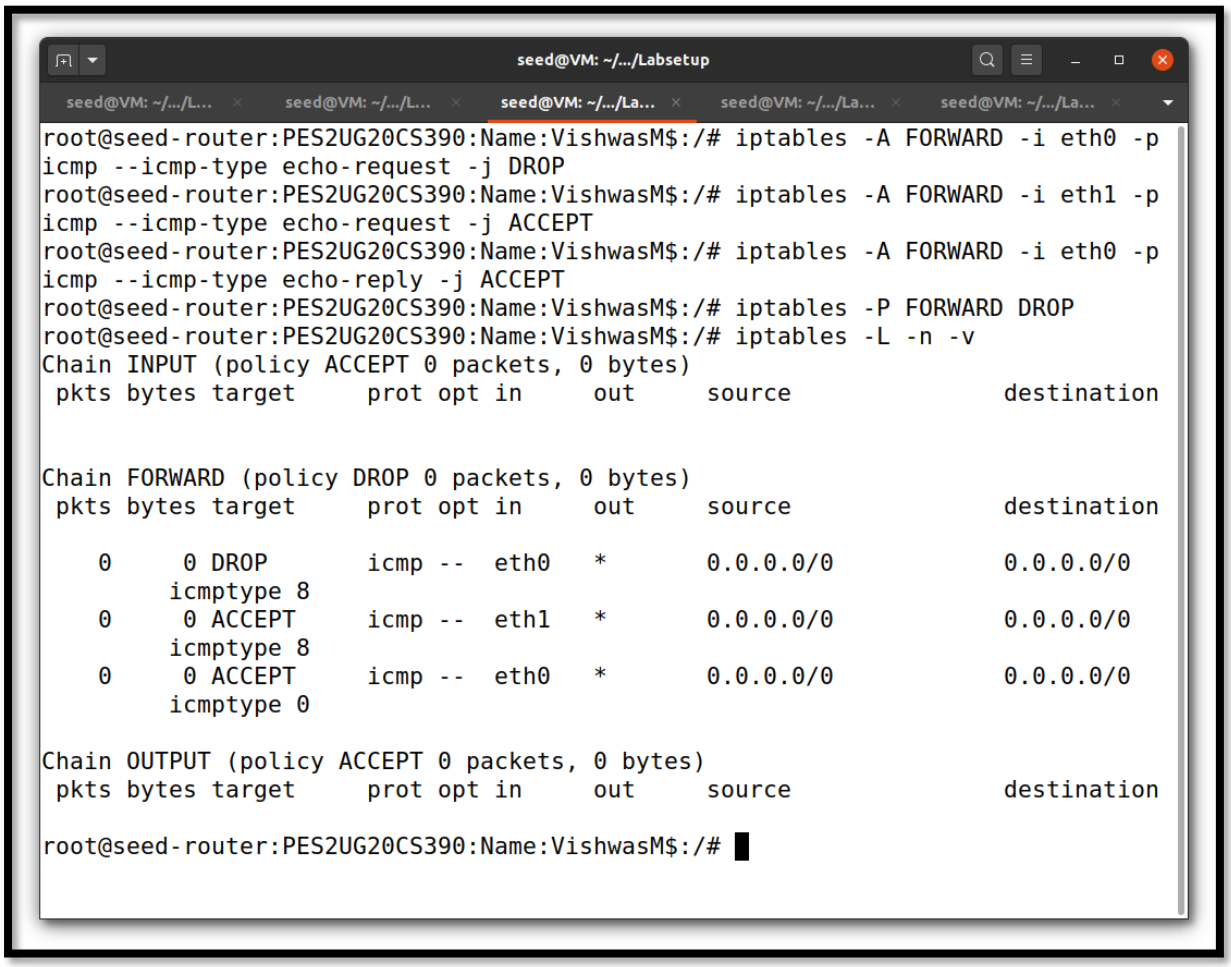
   Ans: Yes, we can ping the router because of the rule that we applied to the router that only ping should be accepted.

2) Can you telnet into the network?

Ans: No, we can’t telnet into the network because we have applied the rule that except ping no other packets are sent or accepted by the firewall.

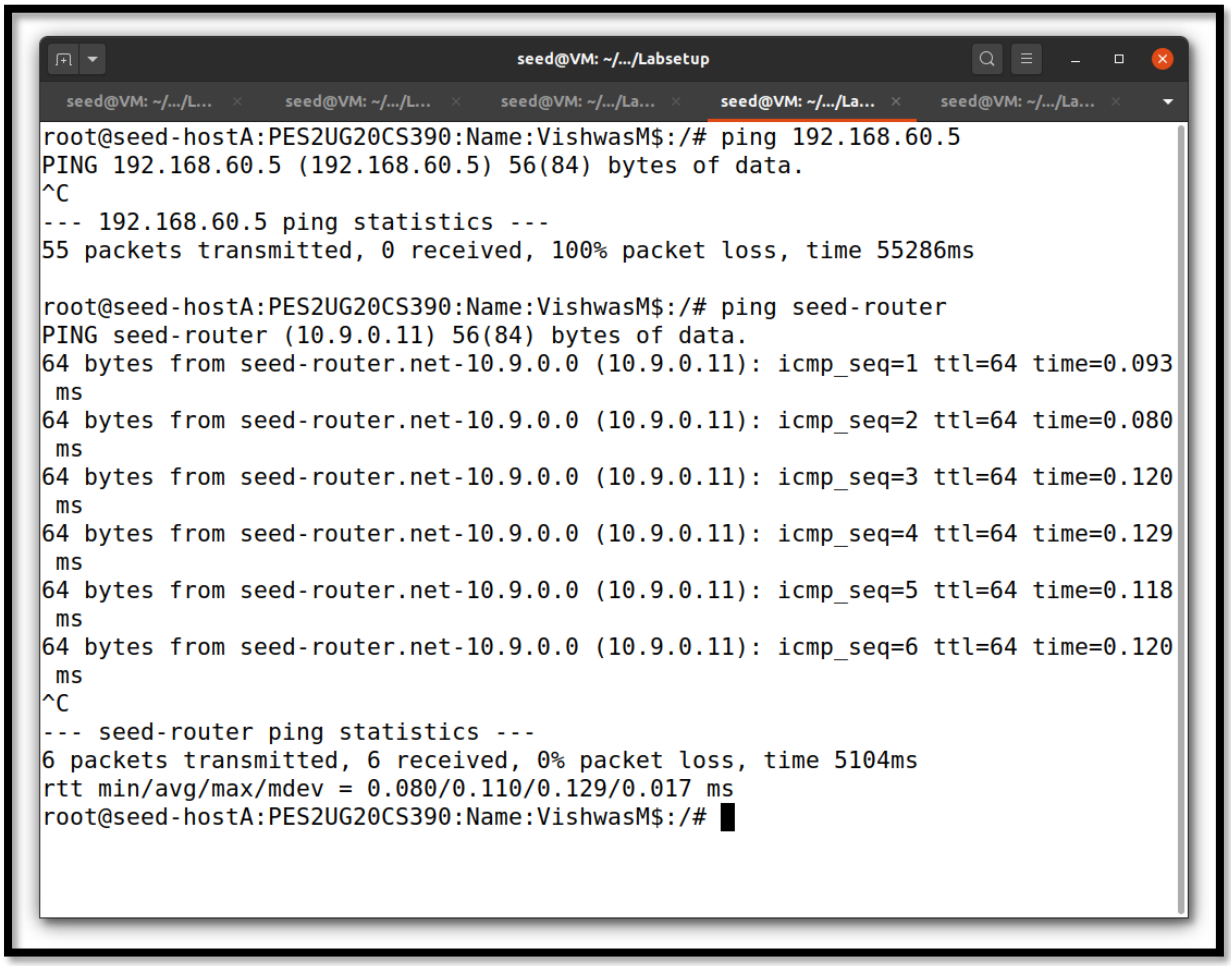
Task 2.B: Protecting the Internal Network

In this task we are going to set up firewall rules on the router to protect the internal network 192.168.60.0/24. We need to use the forward chain for this.



We are applying all the rules to the firewall in order to protect the internal network.

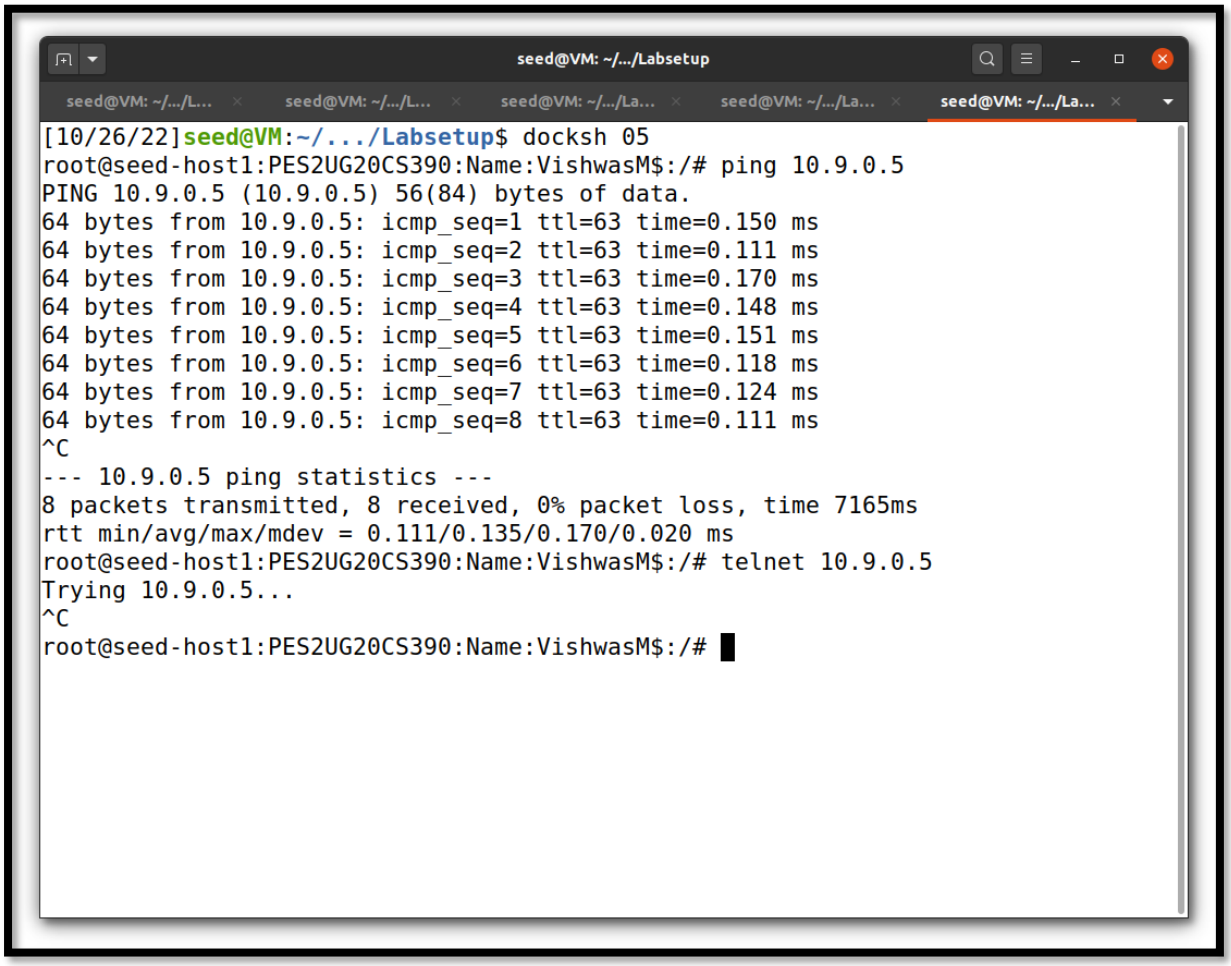
The restrictions applied are:



In host A:

1. <u>Outside hosts cannot ping internal hosts</u>:
   In the above screenshot we can observe that the ping command is not working when pinged to internal network from outside network.

2. <u>Outside hosts can ping the router</u>:
   In the above screenshot we can observe that the ping command is working when pinged to router from outside network.



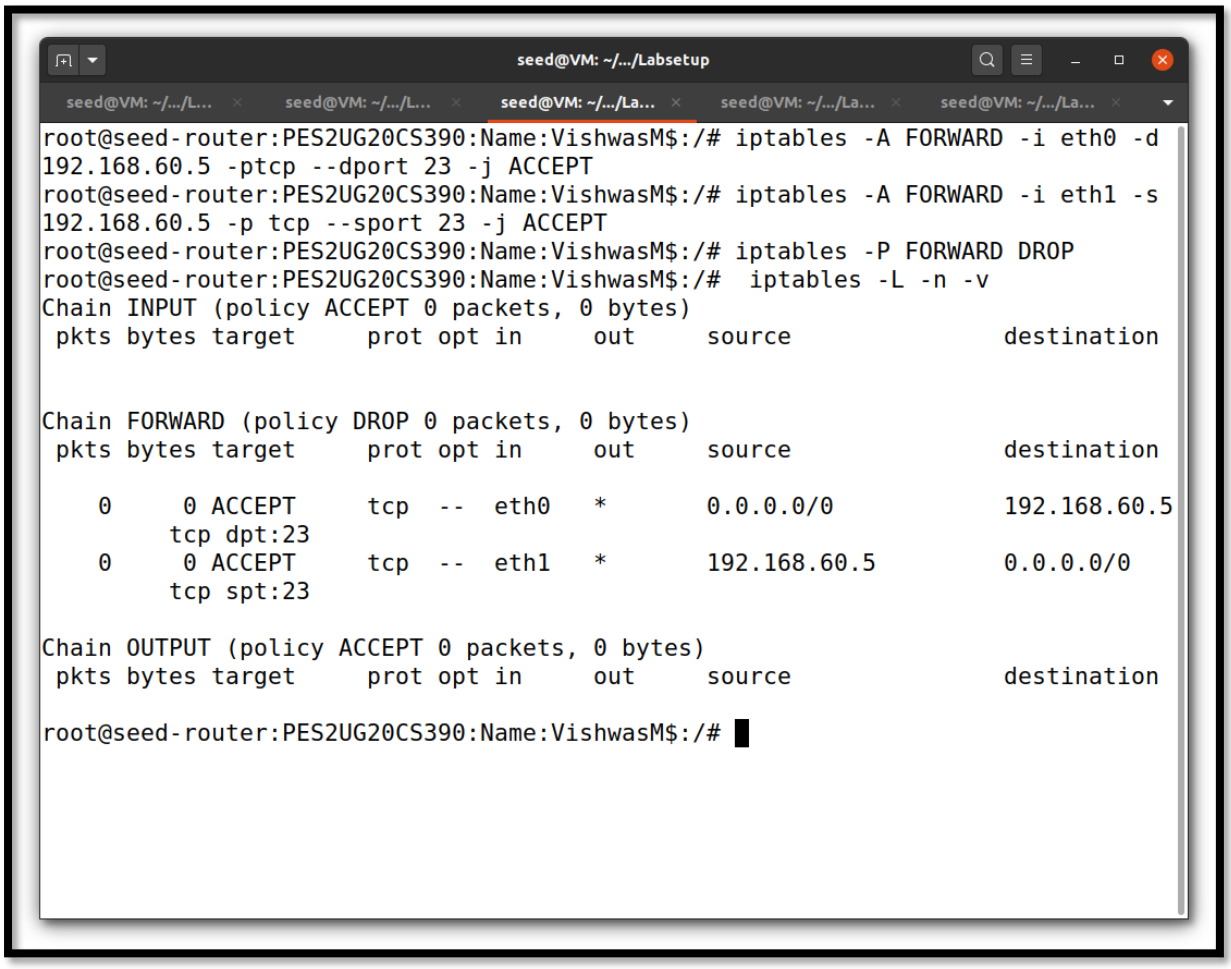
In host 1:

3. <u>Internal hosts can ping outside hosts:</u>
   In the above screenshot we can observe that the ping command is working when pinged to external network from internal network.

4. <u>All other packets between the internal and external networks should be blocked:</u>
   In the above screenshot we can observe that the telnet command is not working from internal network to outside network.

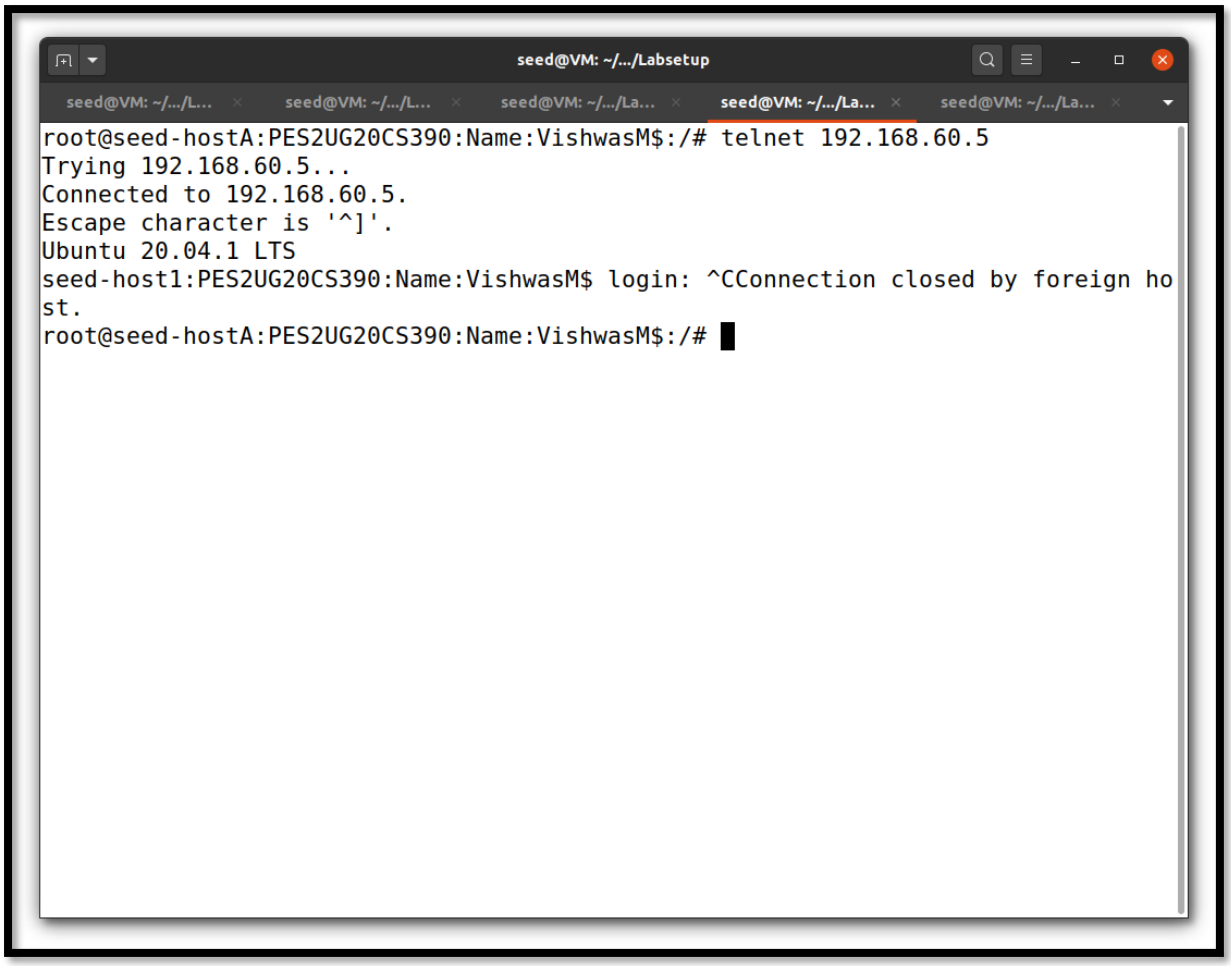
Task 2.C: Protecting Internal Servers

In this task we are going to protect the TCP servers inside the internal network.

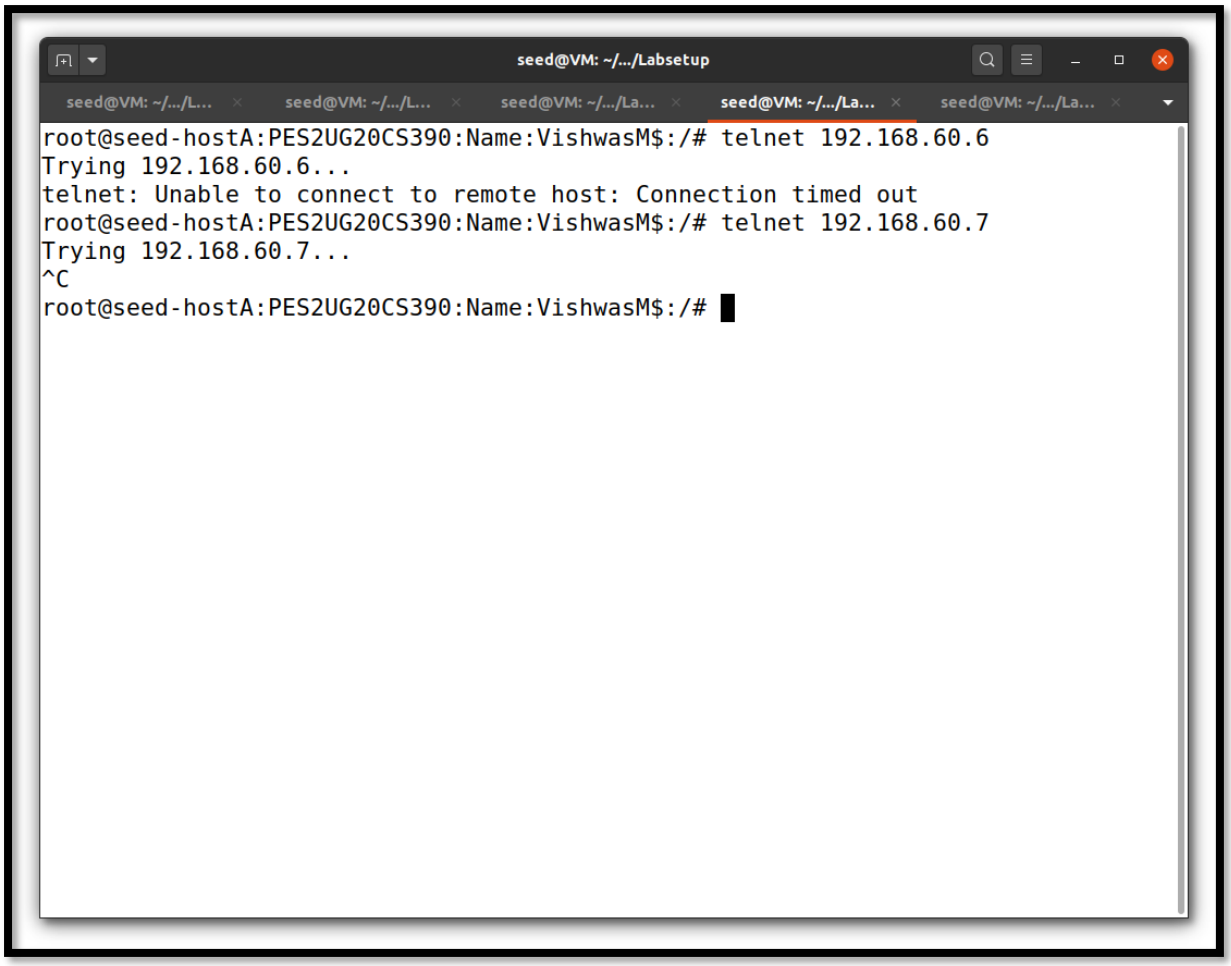


We have updated the rules in the firewall.

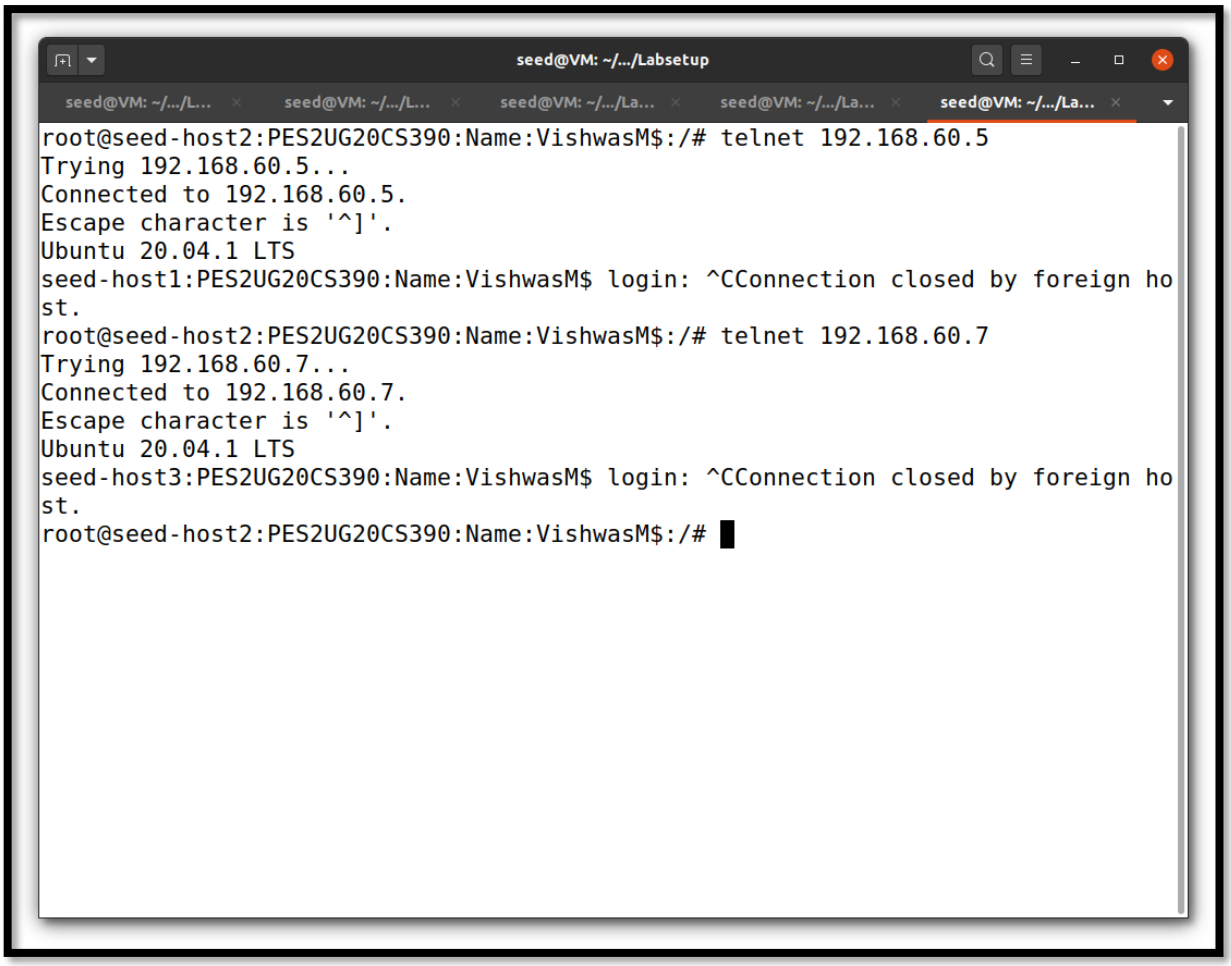
The restrictions applied are:



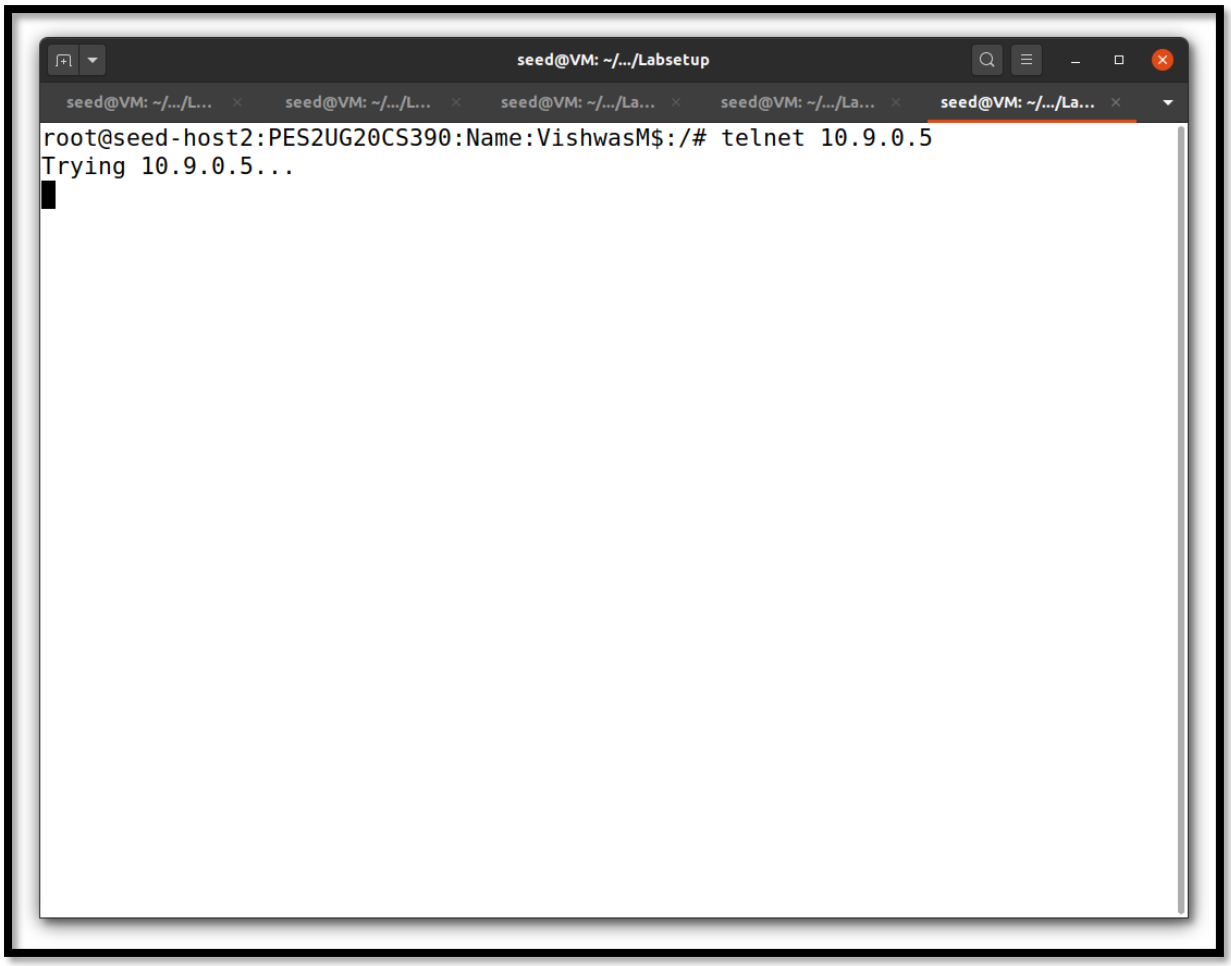
1. <u>Outside hosts can only access the telnet server on 192.168.60.5:</u>
   In the above screenshot we can only access telnet server on 192.168.60.5. When accessed on other IP addresses telnet will not work.



2. Outside hosts cannot access the internal servers:
   As we can see in the above screenshot, telnet is not working because of the rule that we applied.



3. Internal hosts can access the internal servers:
   In the above ss, we can see that the telnet works from internal hosts to internal server.



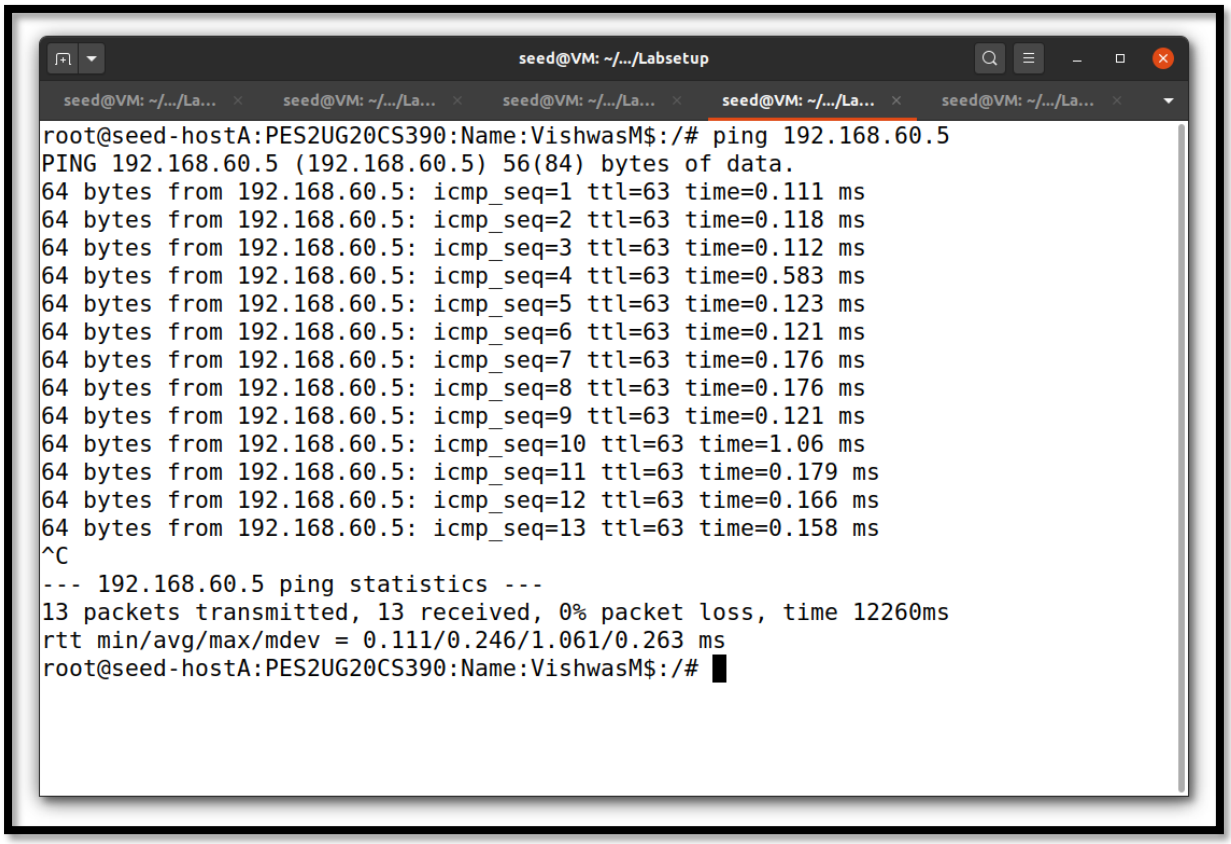
4. Internal servers cannot access the external hosts:
   In the above we are trying to connect with external host(10.9.0.5) from the internal host(host 2) which is failing to connect.

Task 3: Connection Tracking and Stateful Firewall

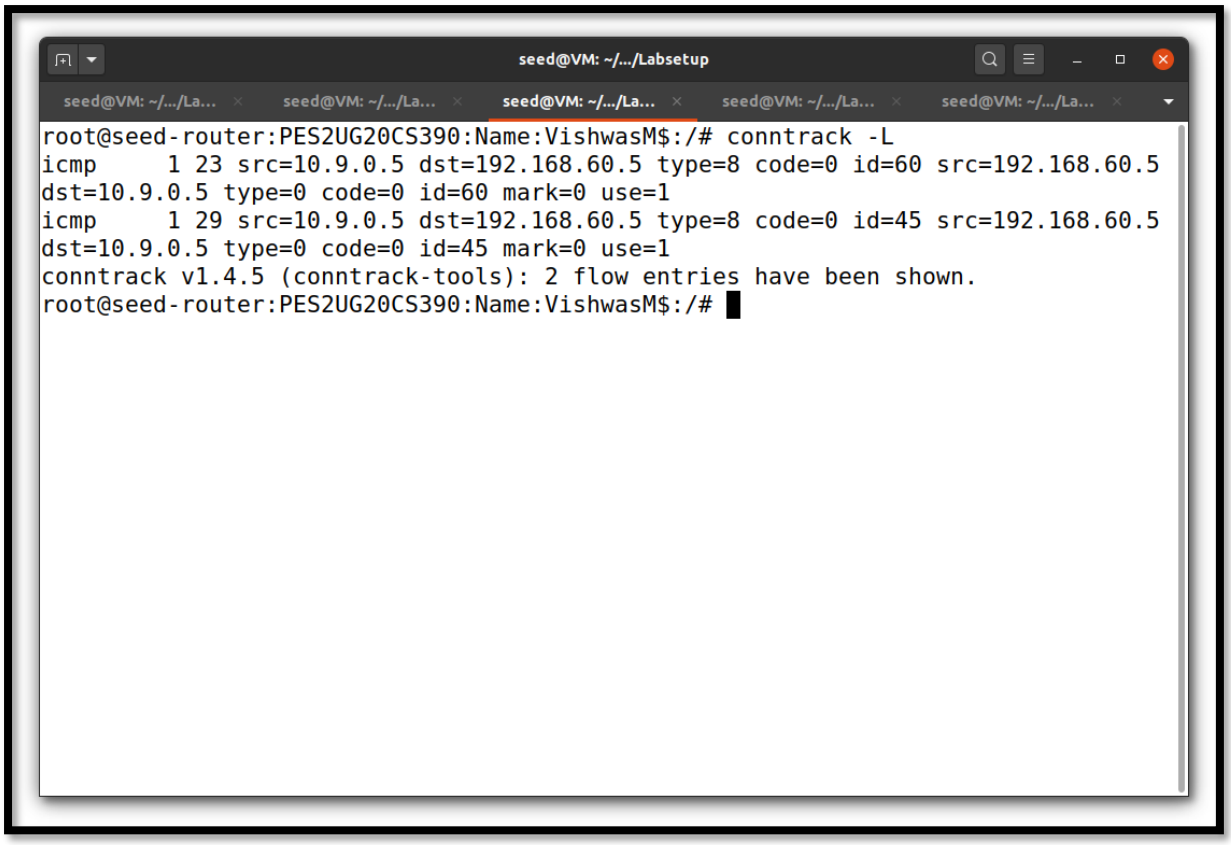
Task 3.A: Experimenting with the Connecting Tracking

The goal of this task is to use a series of experiments to understand the connection concept in this tracking mechanism, especially for the ICMP and UDP protocols, unlike TCP, they do not have connections.

ICMP Experiment:



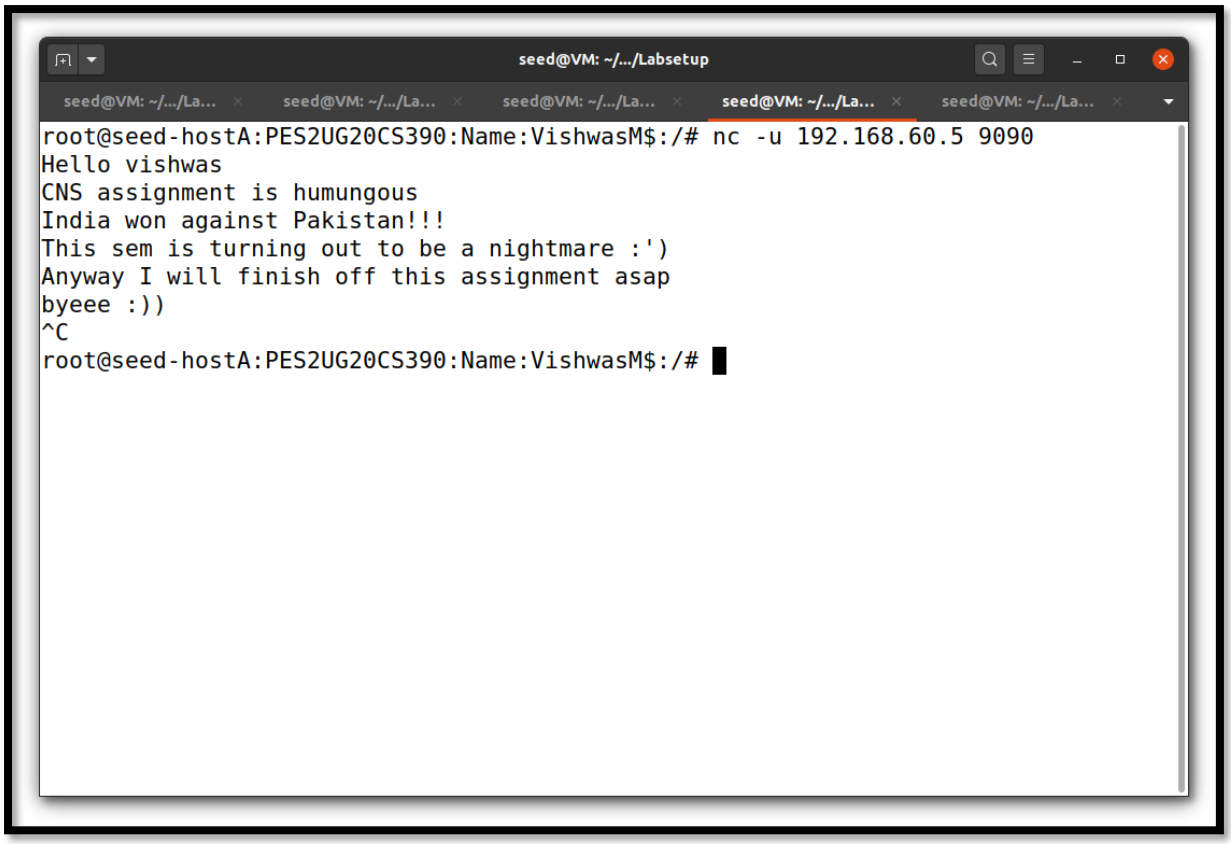
We are pinging from an external host to an internal host to observe the connection status inside the router.

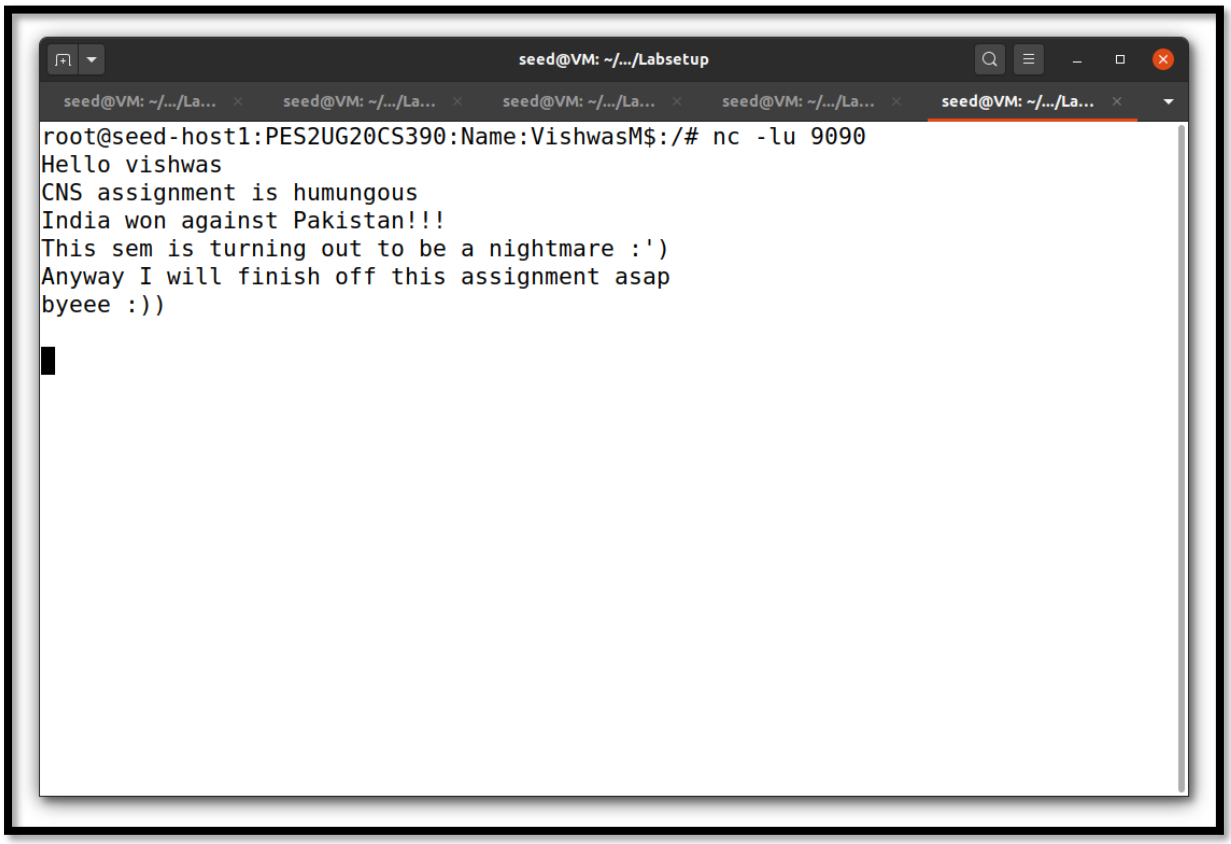


After stopping the connection in the external network, with the help of the command used in the above ss, we can find the connection status inside the router.

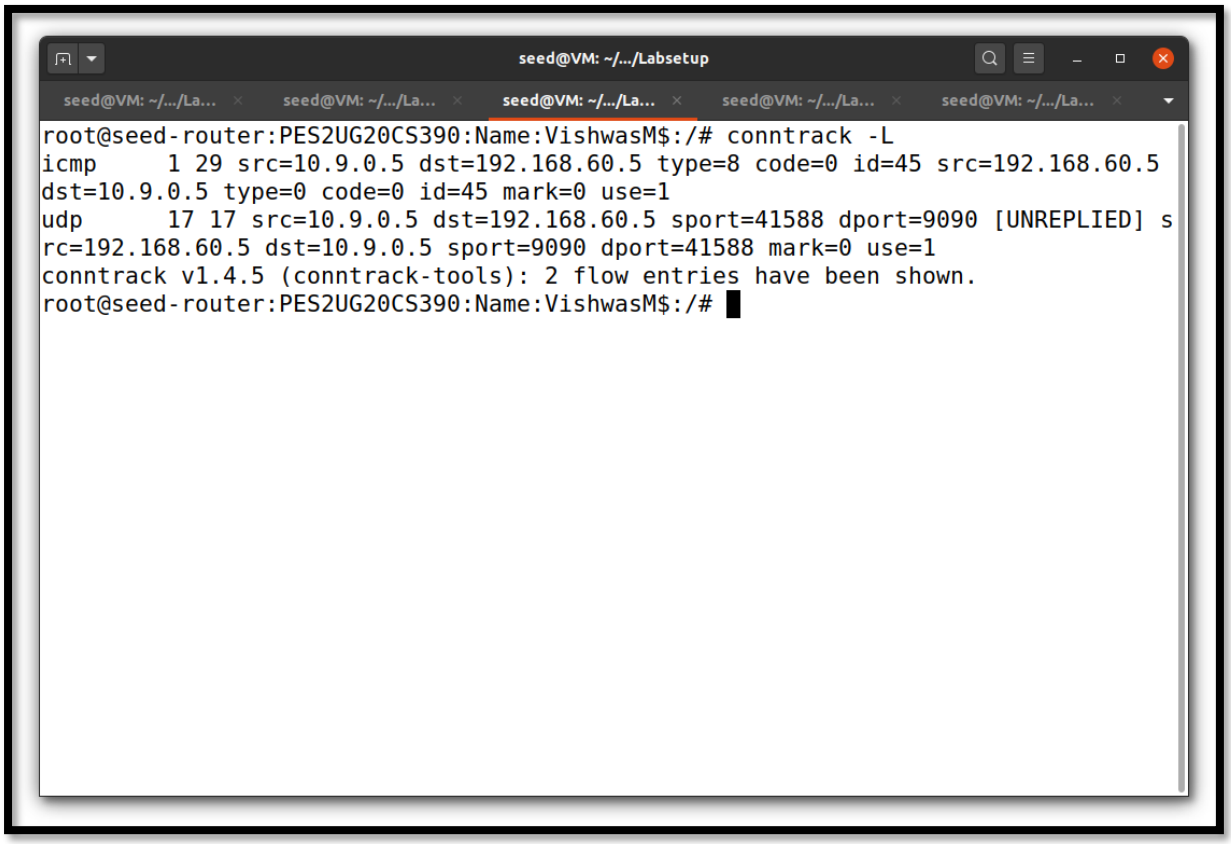
UDP Experiment:

First we are establishing a UDP connection between the external host(host A) and the internal host(host 1).


seed@VM: ~/.../Labsetup
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
root@seed-hostA:PES2UG20CS390:Name:VishwasM$:/# nc -u 192.168.60.5 9090
Hello vishwas
CNS assignment is humungous
India won against Pakistan!!!
This sem is turning out to be a nightmare :')
Anyway I will finish off this assignment asap
byeee :))
^C
root@seed-hostA:PES2UG20CS390:Name:VishwasM$:/#


seed@VM: ~/.../Labsetup
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
seed@VM: ~/.../La...
root@seed-host1:PES2UG20CS390:Name:VishwasM$:/# nc -lu 9090
Hello vishwas
CNS assignment is humungous
India won against Pakistan!!!
This sem is turning out to be a nightmare :')
Anyway I will finish off this assignment asap
byeee :))

After performing some conversations between the two hosts, we have to cut the connection.

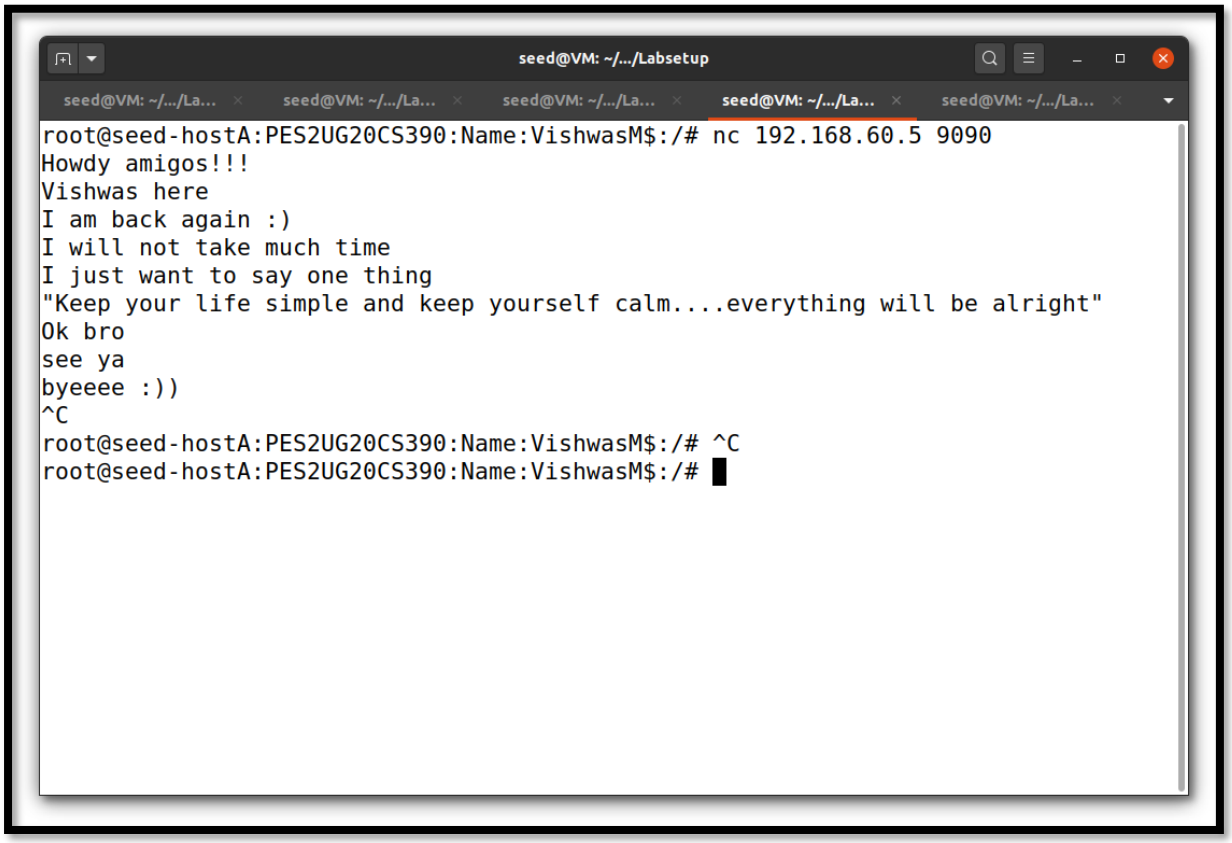


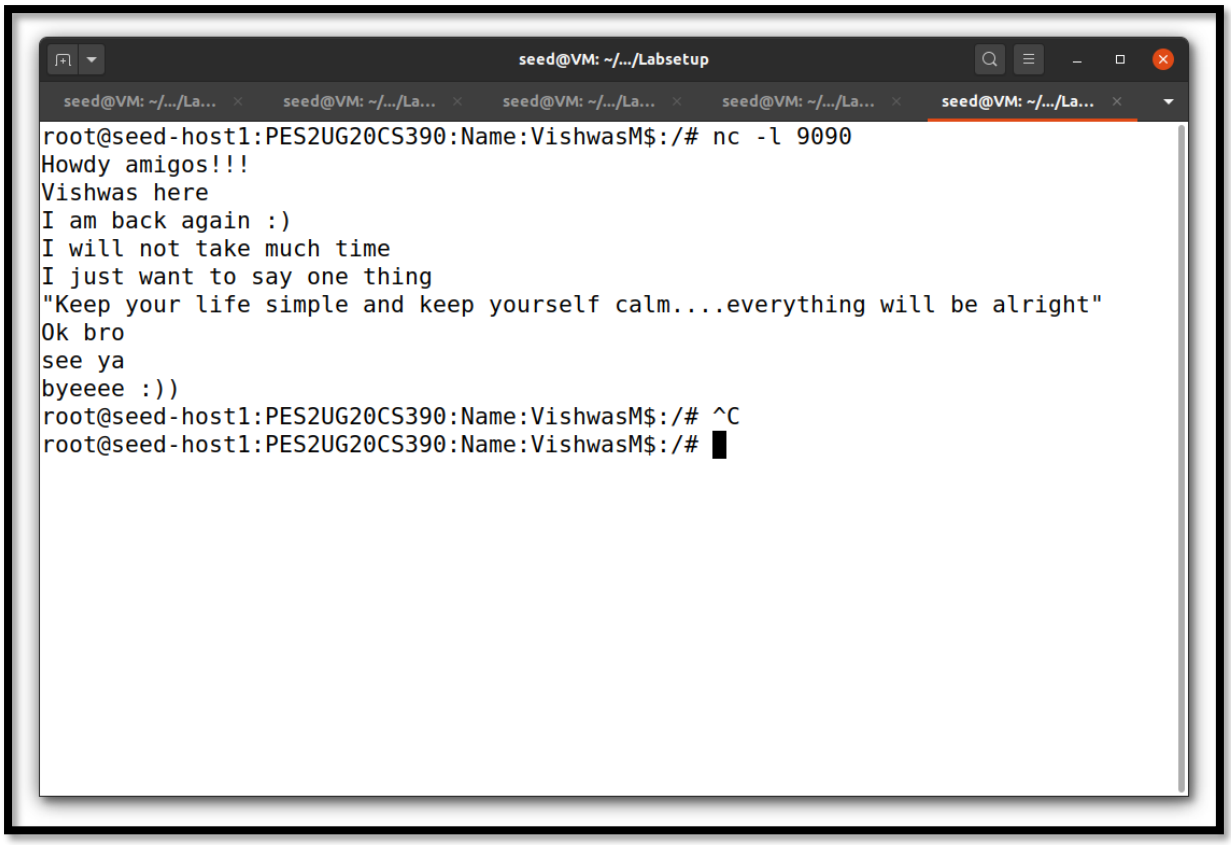
After cutting the connection, we have to execute the above command to see the connection status inside the router. We can see that the UDP connection was established inside the router as shown in the above ss.

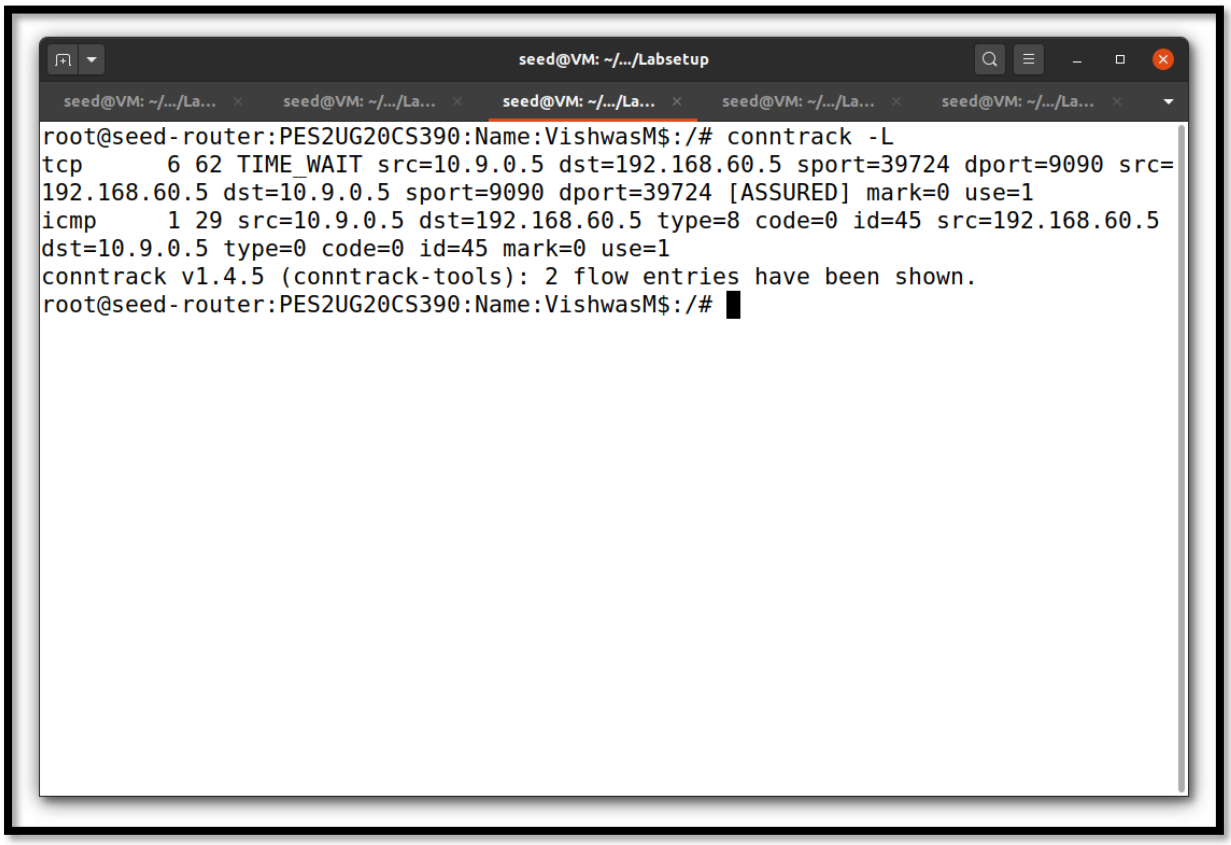
When we perform this experiment multiple times the time will reduce gradually.

TCP Experiment:

First we have to establish a TCP connection between the external host(host A) and the internal host(host 1).







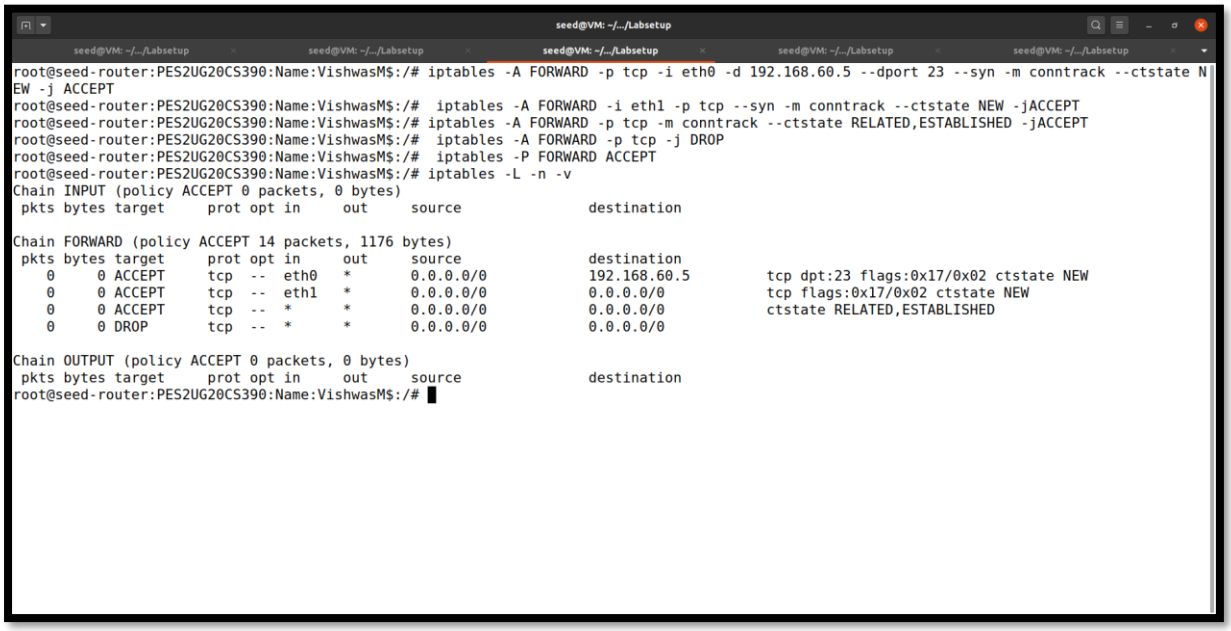
As we can see that TCP connection is shown in the router.

Difference:

In UDP, if we close the connection from host A(external) and still the host 1(internal) is still not closed, then only we can see the UDP connection inside the router. If we cut connection from both sides then the UDP connection will not be shown inside the router.

But in case of TCP, if we close the connection one side, then automatically it closes on the other side also. And we can see the TCP connection inside the router.

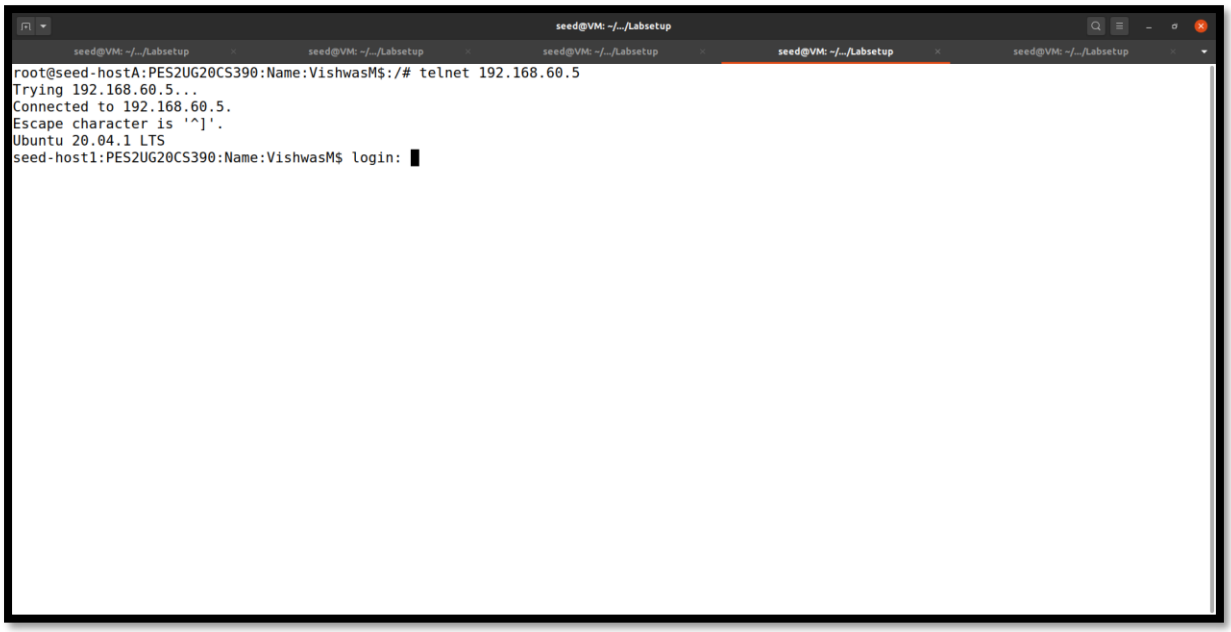
Task 3.B: Setting up a Stateful Firewall



We added some rules to the firewall to set up some restrictions in the firewall for the firewall to work like stateful.

We will check if the restrictions are working or not:

1. Outside hosts can only access the telnet server on 192.168.60.5, not the other internal hosts.

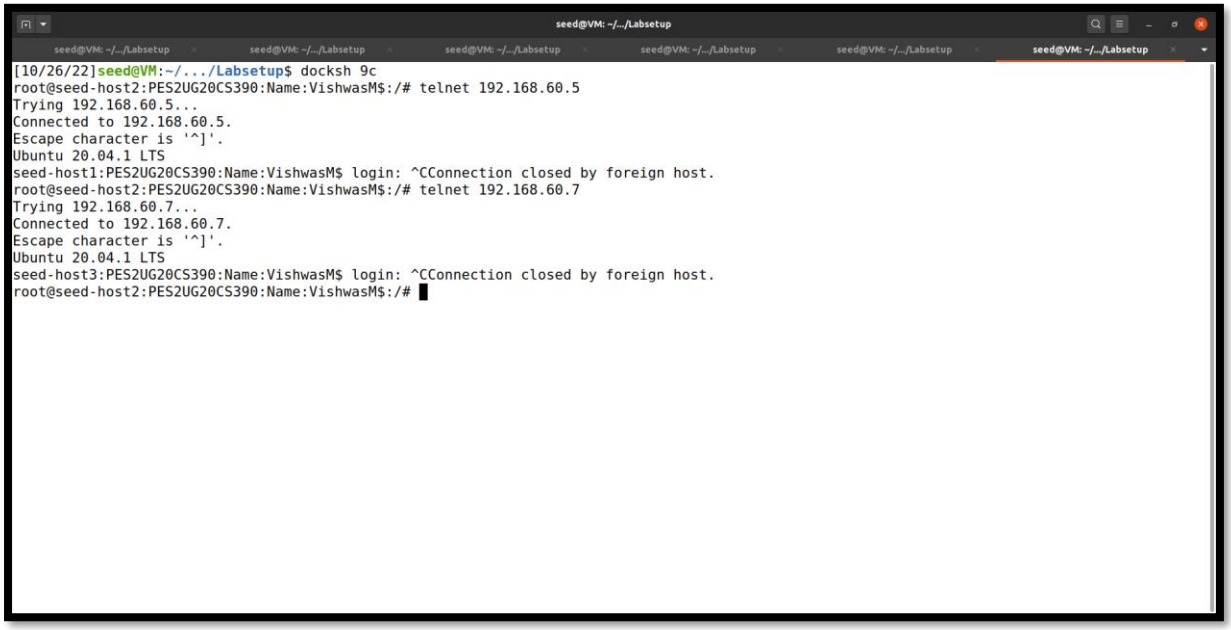


We can clearly see that the connection is established between outside host and server 192.168.60.5

2. Outside hosts cannot access other internal servers

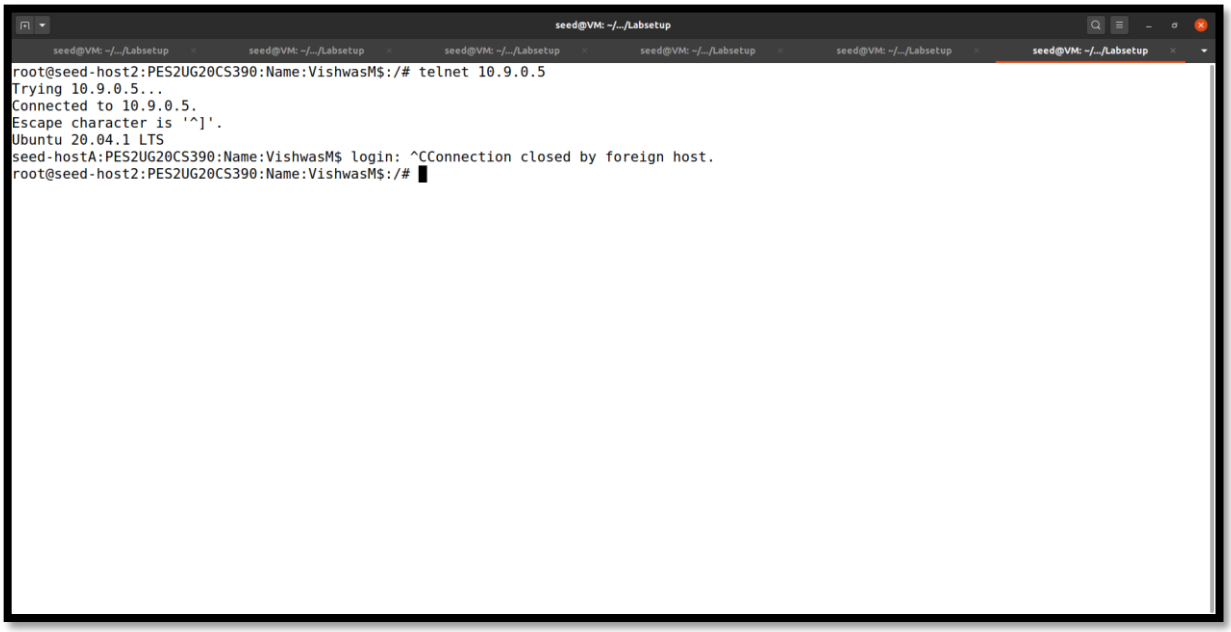
We can see that the connection failed to establish between host and other internal servers.

3. Internal hosts can access all the servers.



The connection is well established between internal hosts and all the servers.

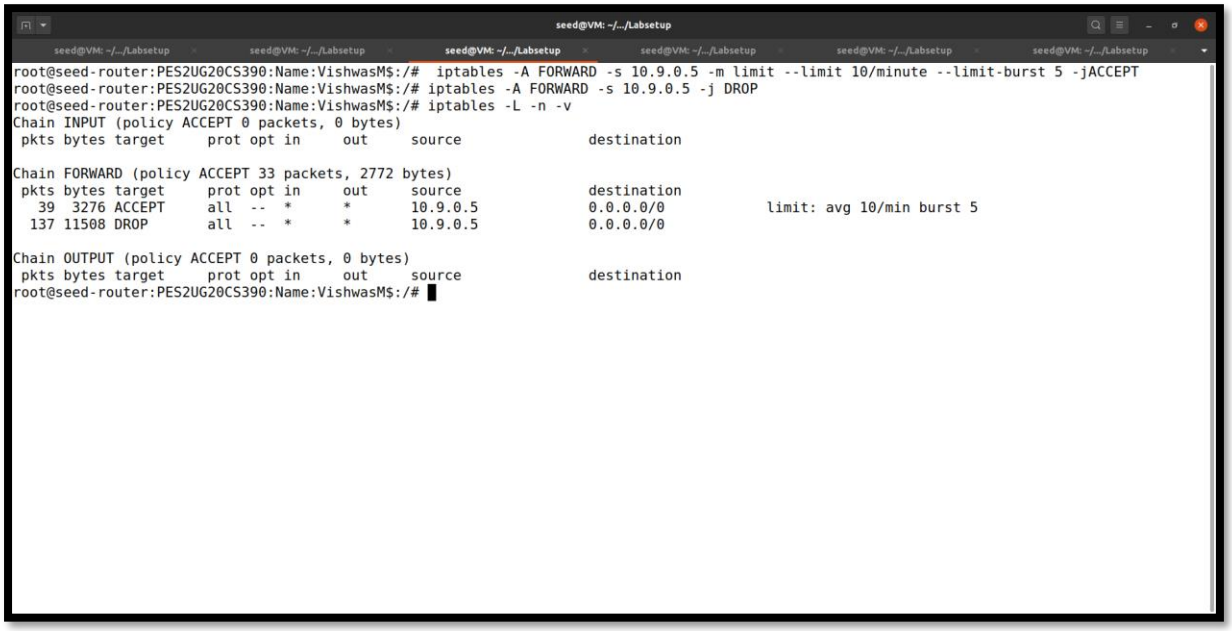
4. Internal hosts can access the external servers.



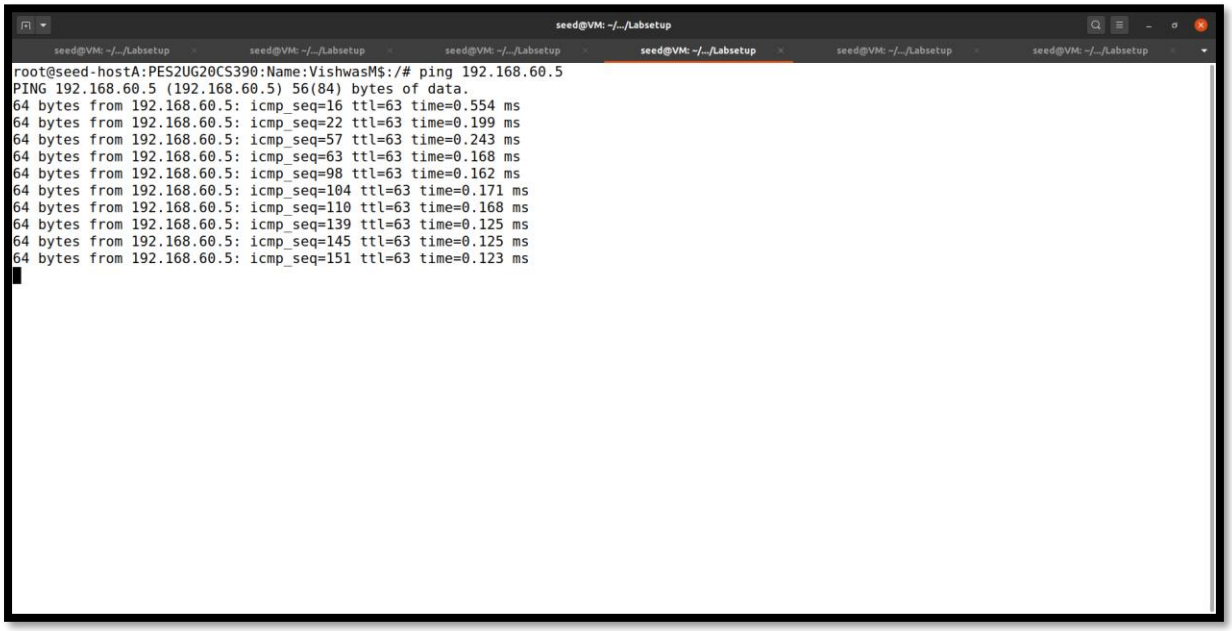
Connection is well established between internal hosts and external servers.

Task 4: Limiting Network Traffic

a) With dropping packets:

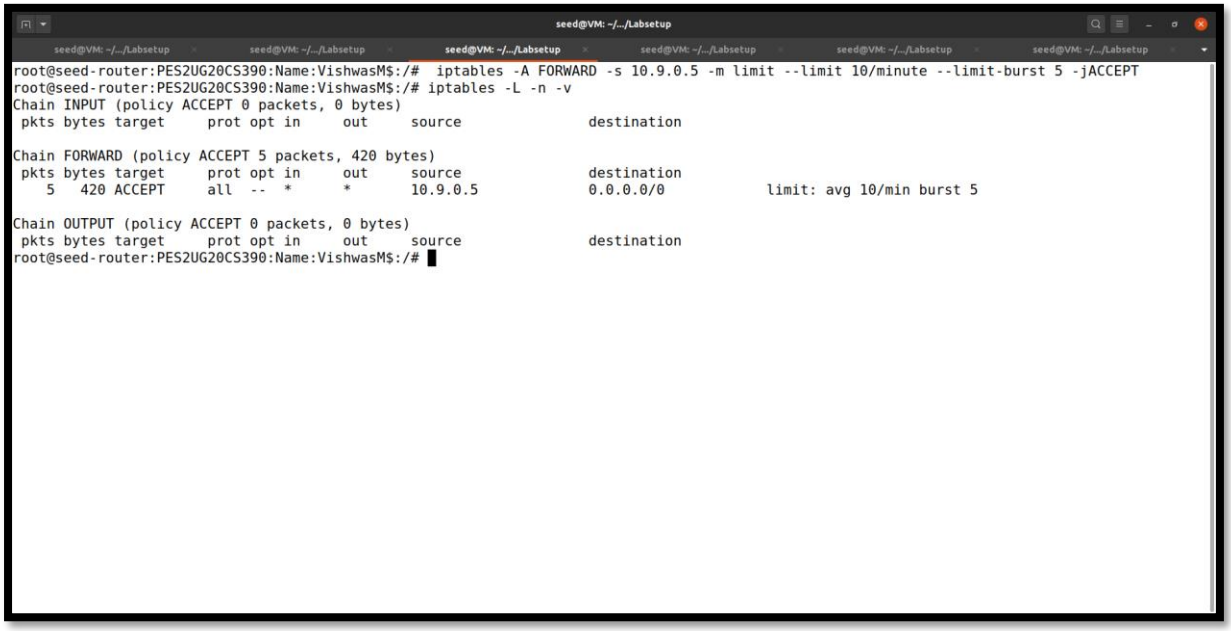


We have set the rules for the firewall here to drop the extra packets.

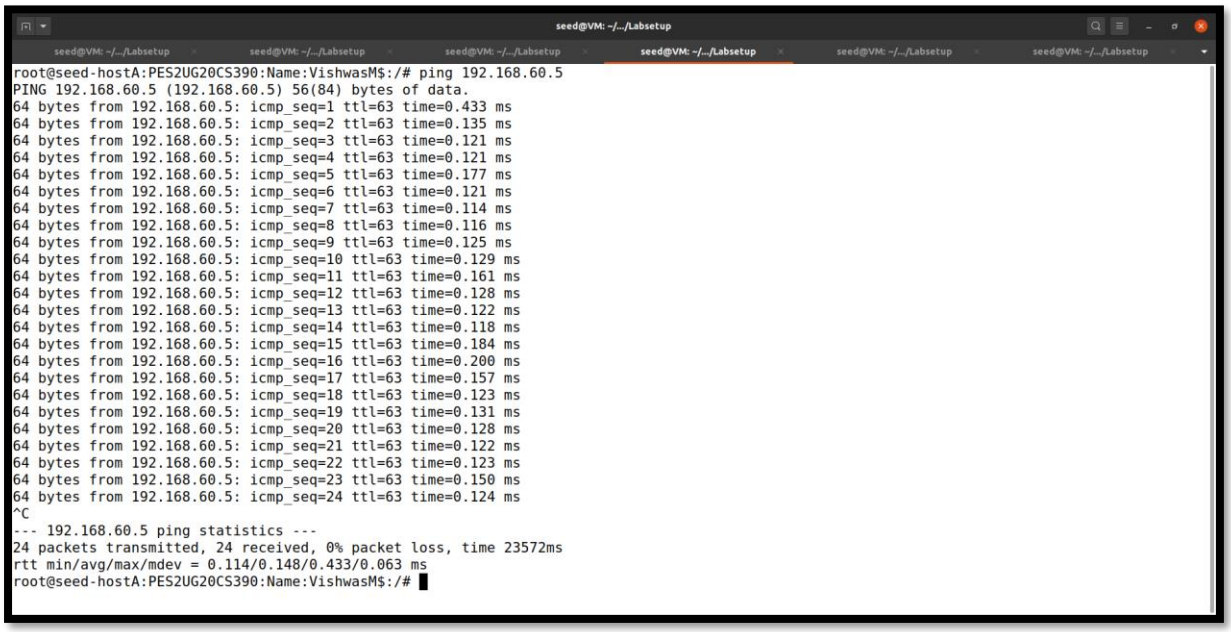


Only 10 packets are allowed to transmit through the firewall per minute, rest of the packets are dropped.

b) Without dropping the packets



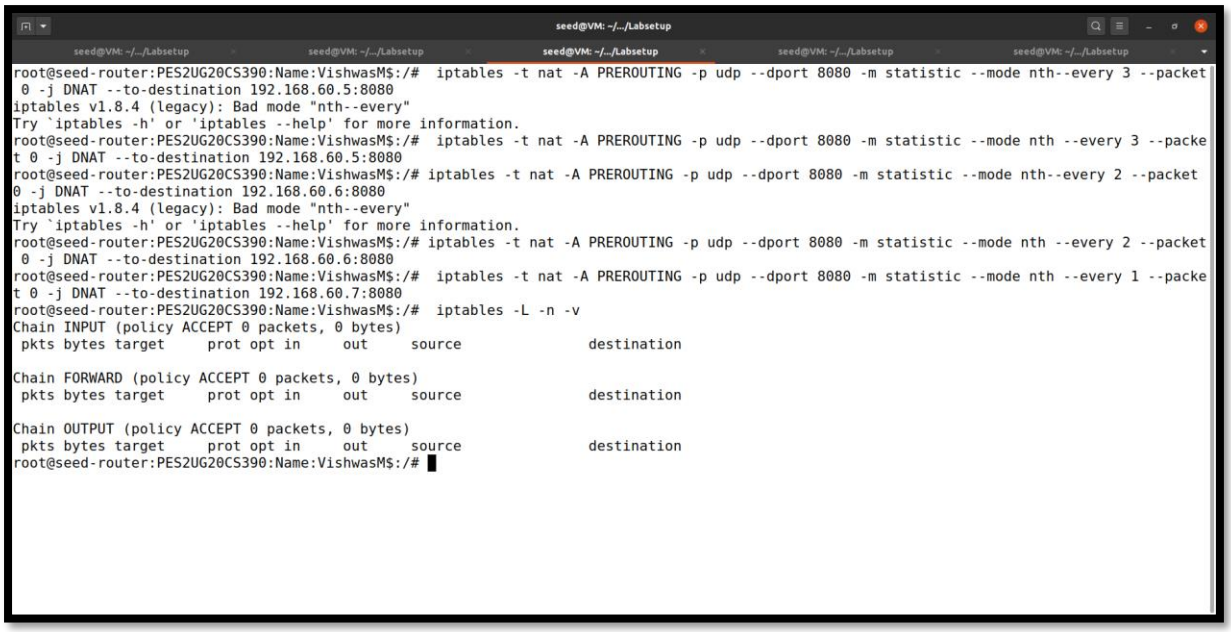
The rule which tells to drop the packets is not uploaded to the firewall.



The packets will not drop the packets which will result in sending the packets to destination server.

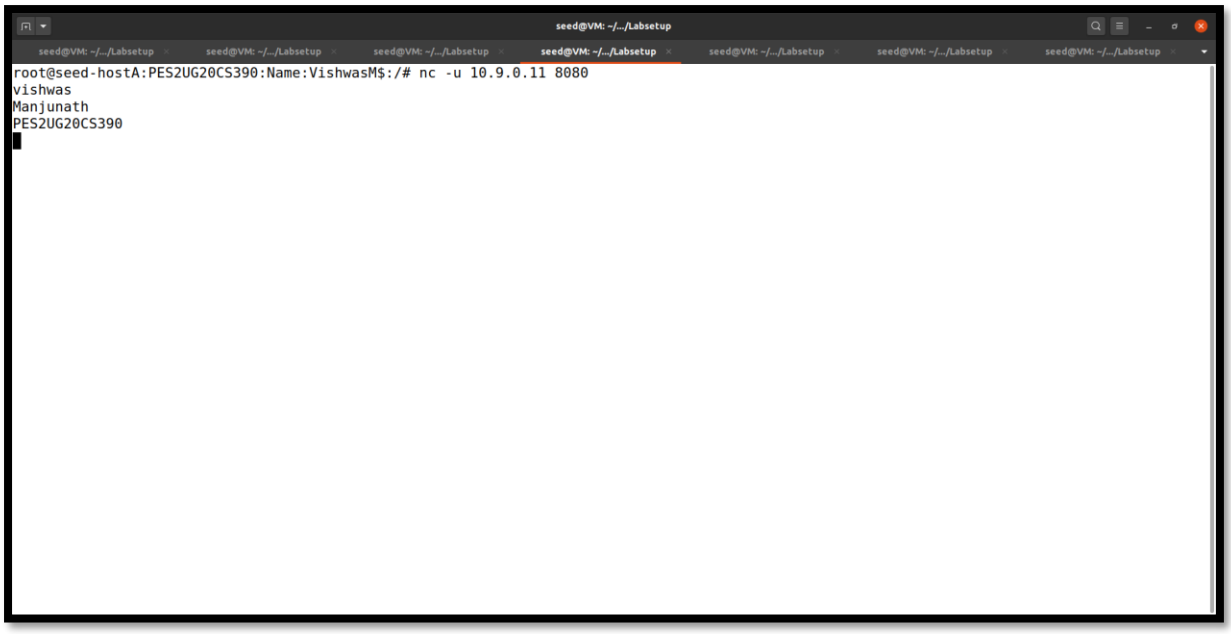
Task 5: Load Balancing

a. Using the nth mode (round-robin):



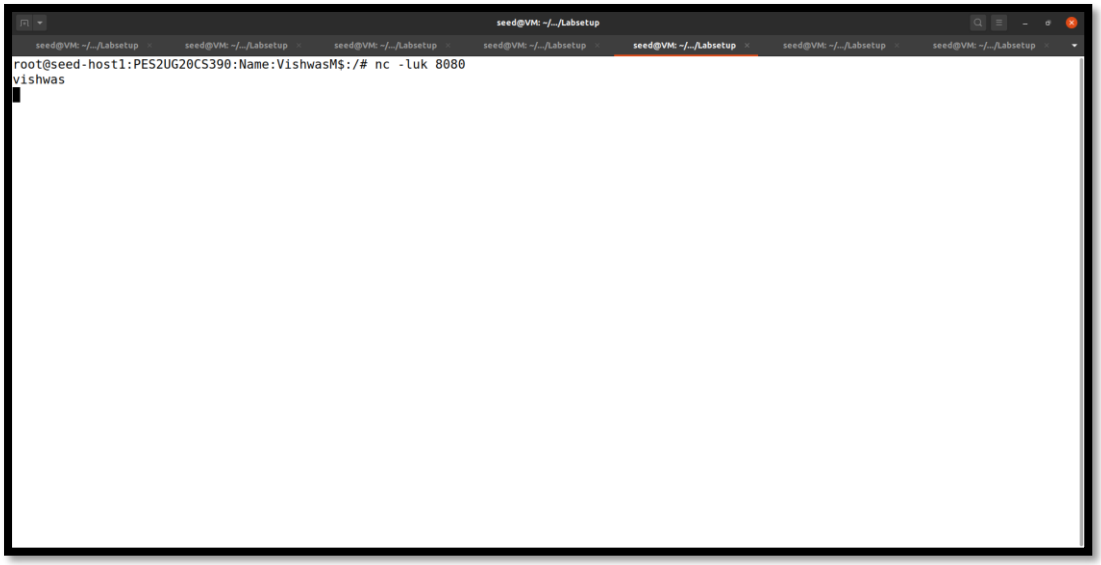
We are updating the rules in the firewall.

Host A:



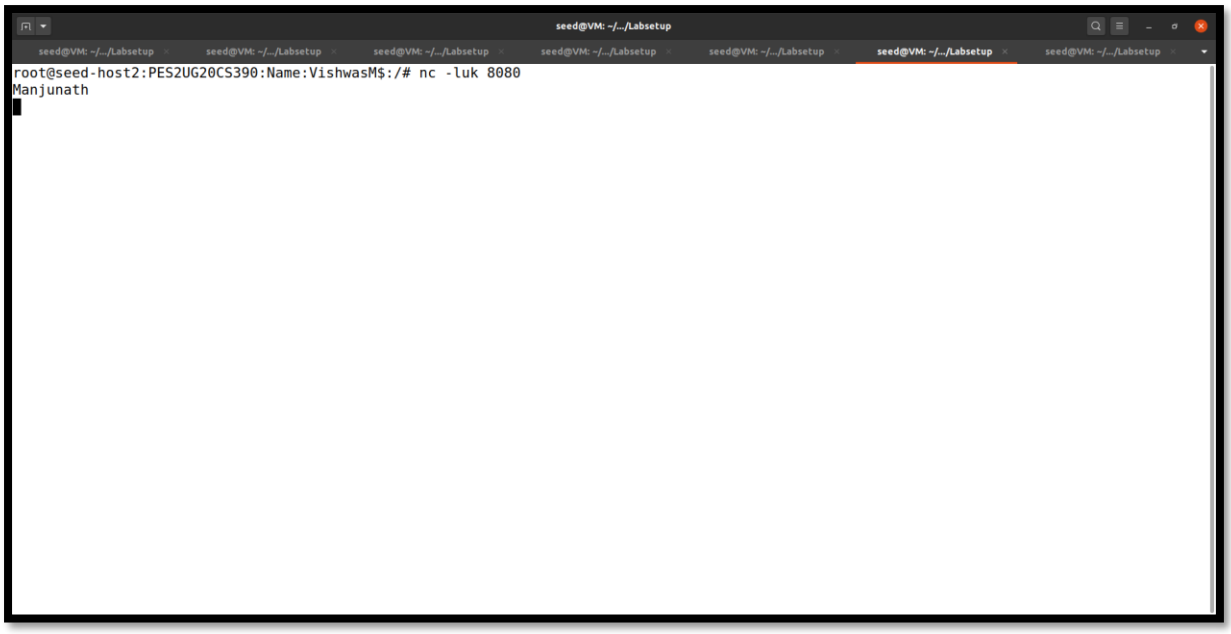
We have leave 30 sec gap between each word else all the 3 words will be considered as a single packet and go to the same host.

Host 1:



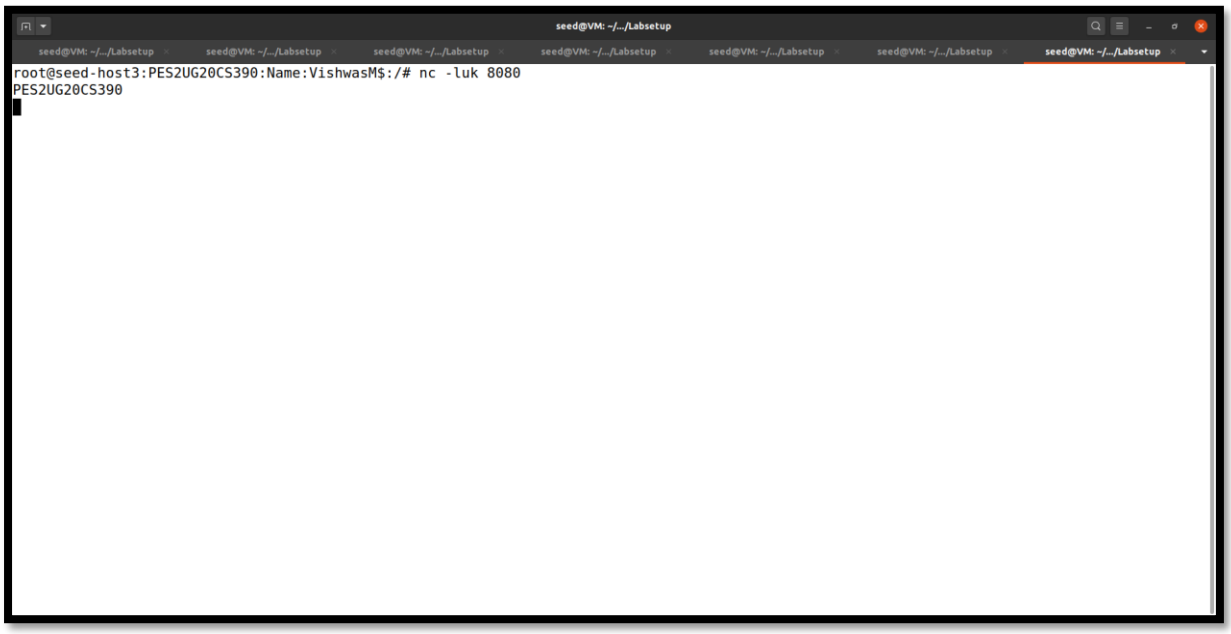
We can see that the first word is coming to host 1.

Host 2:



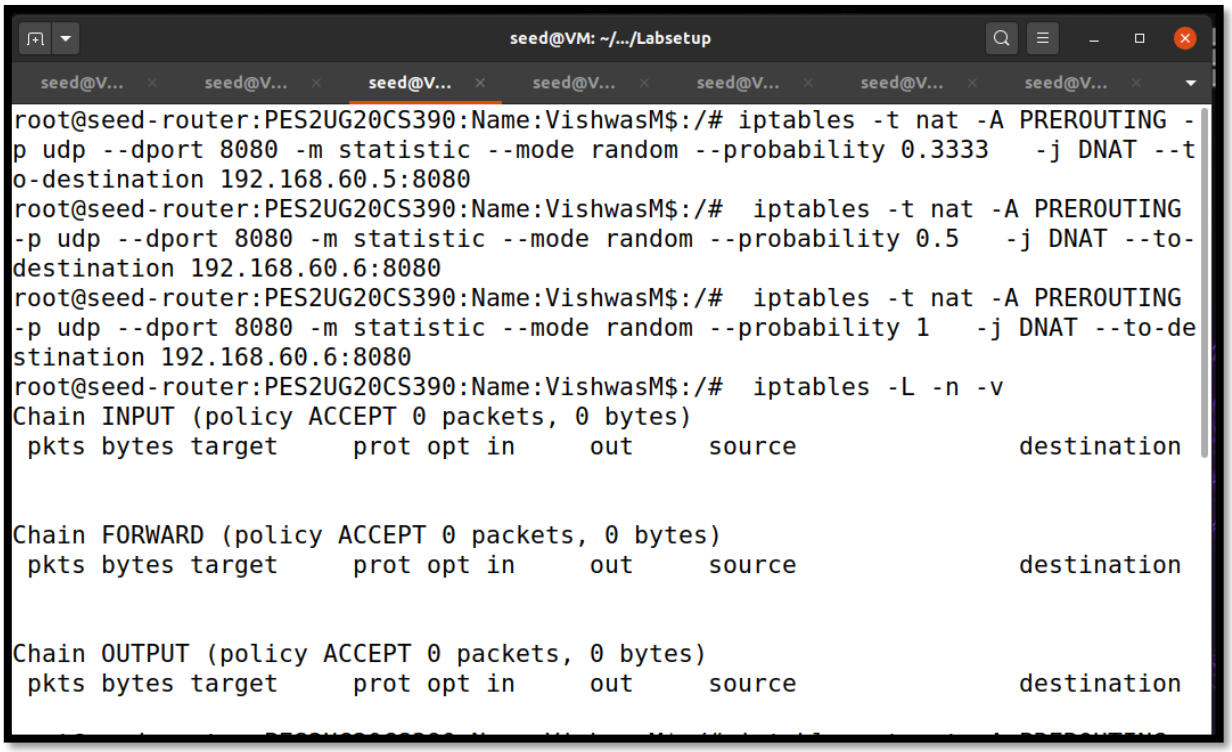
We can see that the 2nd word is coming to host 2

Host 3:



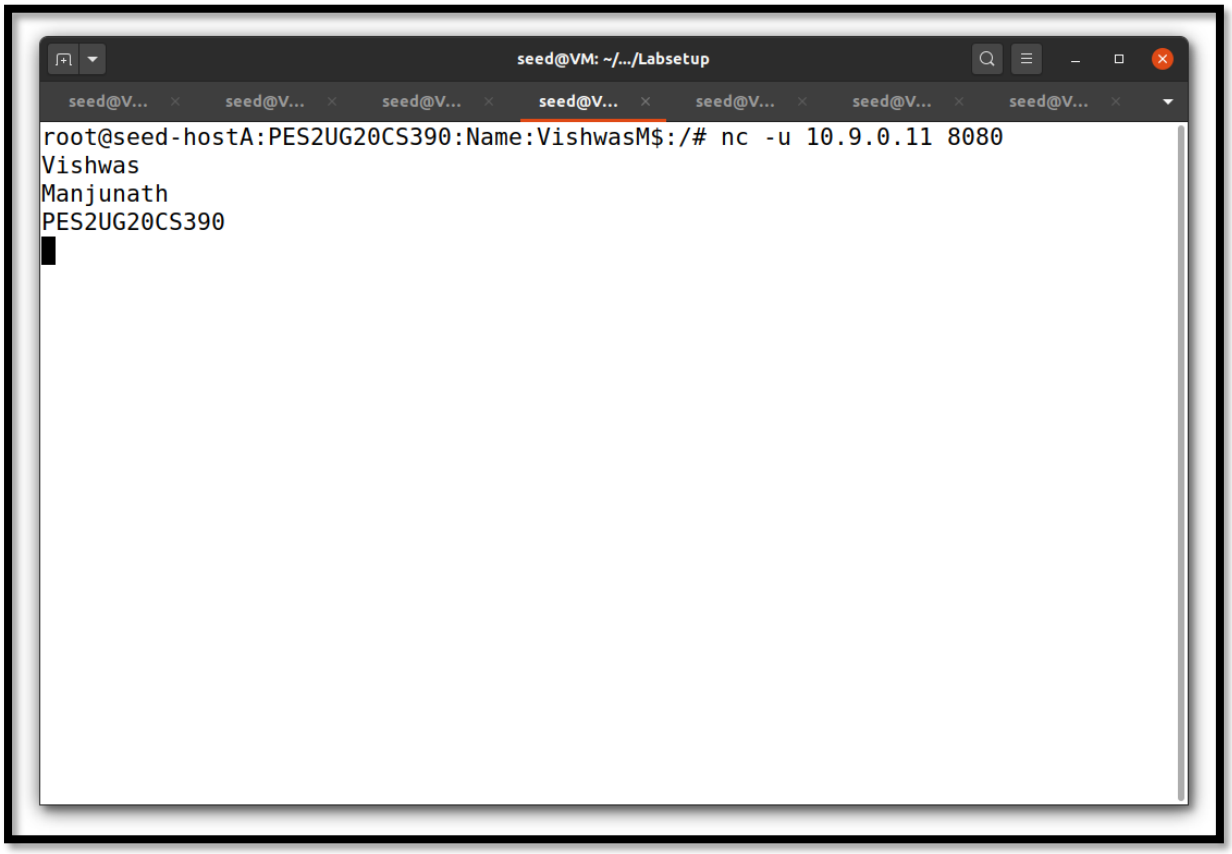
We can see that 3rd word is coming to host 3.

b. Using random mode: The following rule will select a matching packet with the probability P. You need to replace P with a probability number.



We have updated the rules of the server.

Host A:



We have leave 30 sec gap between each word else all the 3 words will be considered as a single packet and go to the same host.

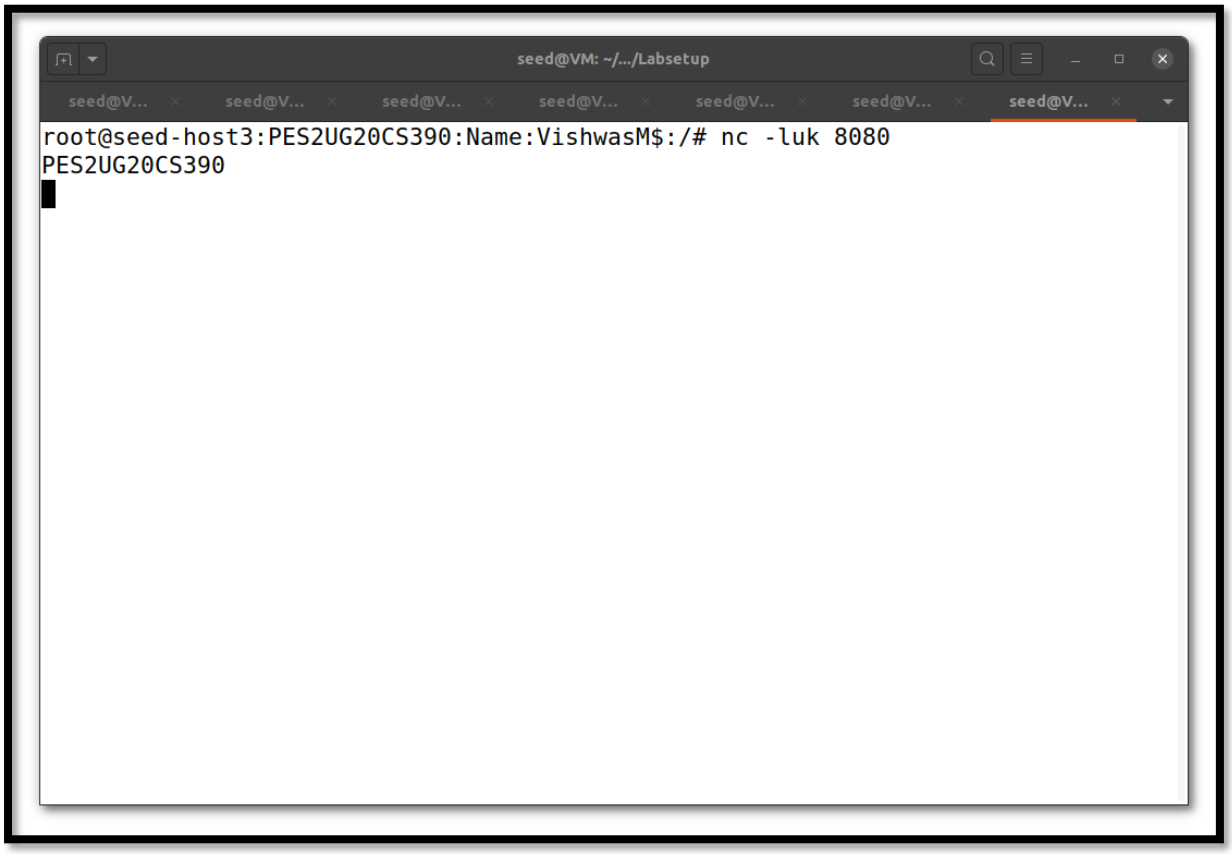
Host 1:

We can see that the first word is coming to host 1.

Host 2:

We can see that the second word is coming to host 2.

Host 3:



We can see that the third word is coming to host 3.