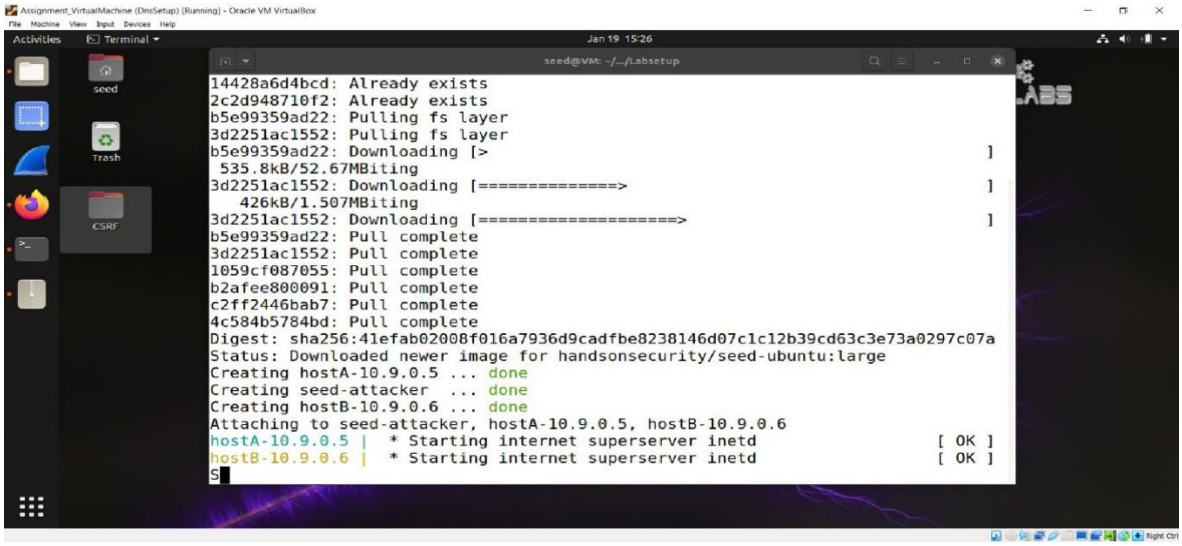
CSC 5290 : Cyber Security Practices
Lab1
Winter 2023
Student name: Kanchan Chopde

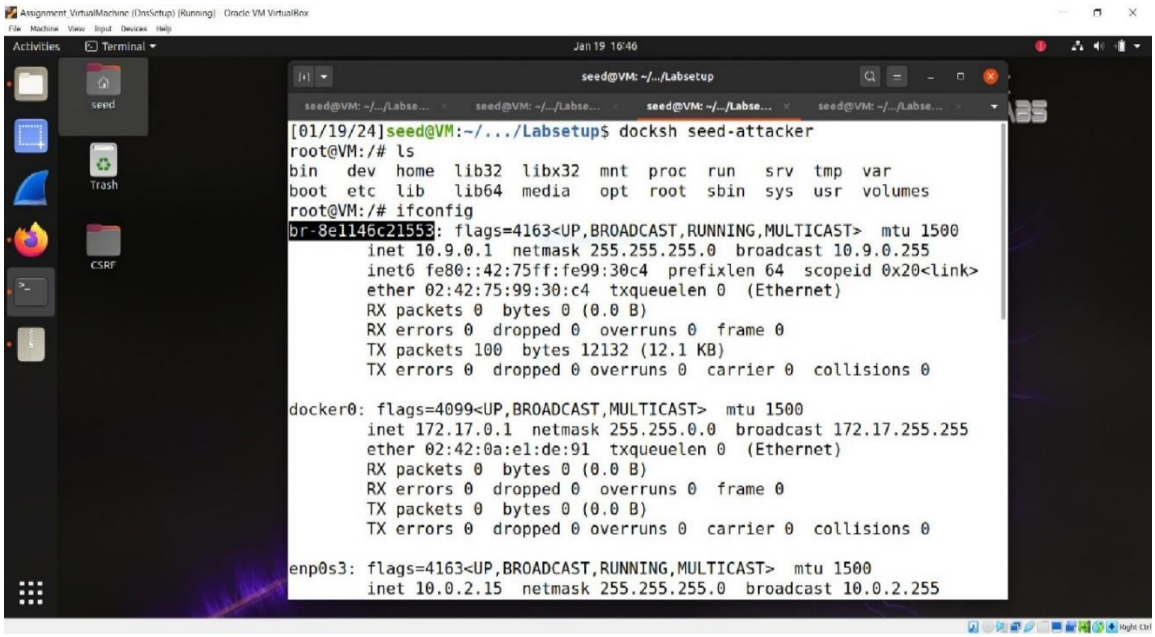
Packet Sniffing and Spoofing

Lab setup :

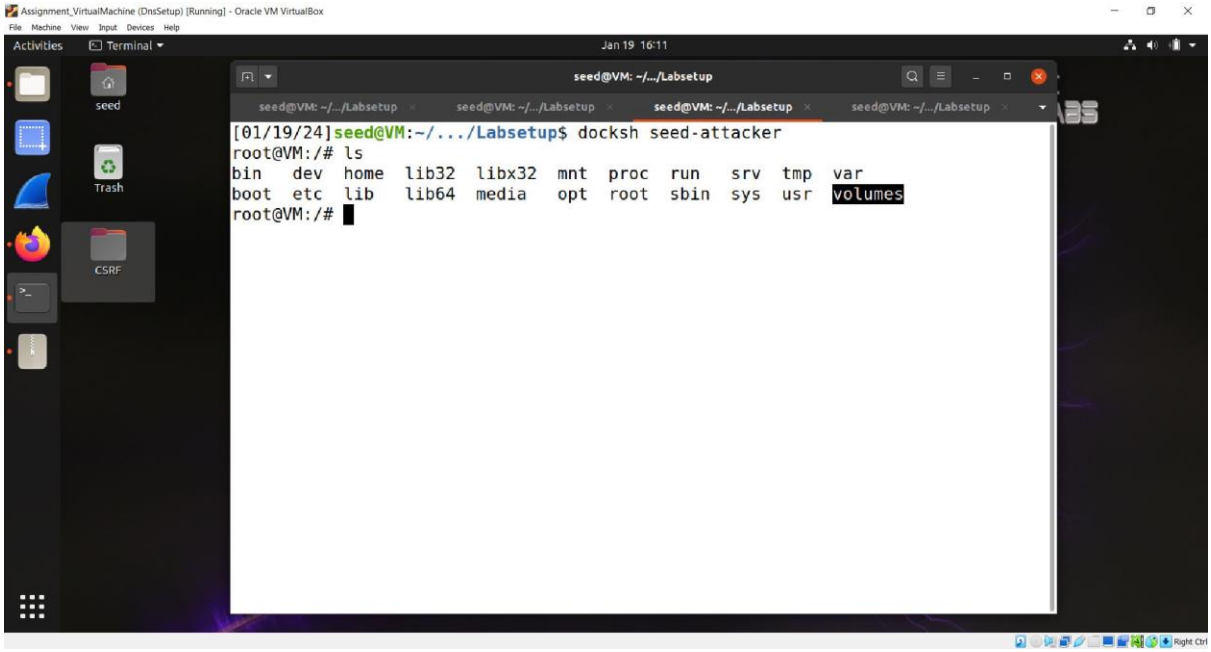
This screenshot shows we are making use of dcup command i.e. docker-compose up to start the services mentioned in docker-compose.yml.
Here it is creating and starting seed-attacker, Host A and Host B.



Below we use ifconfig command to get name of corresponding network interface starting with br- and the ID by docker.

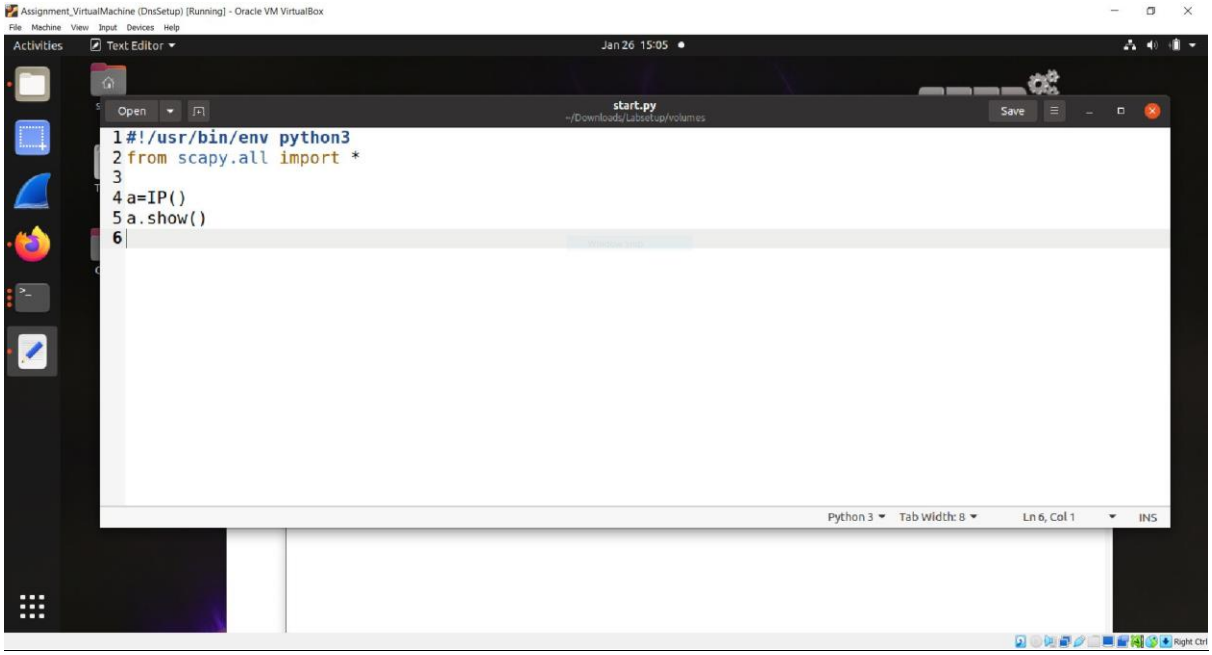


Inside attacker we can see here volumes directory.

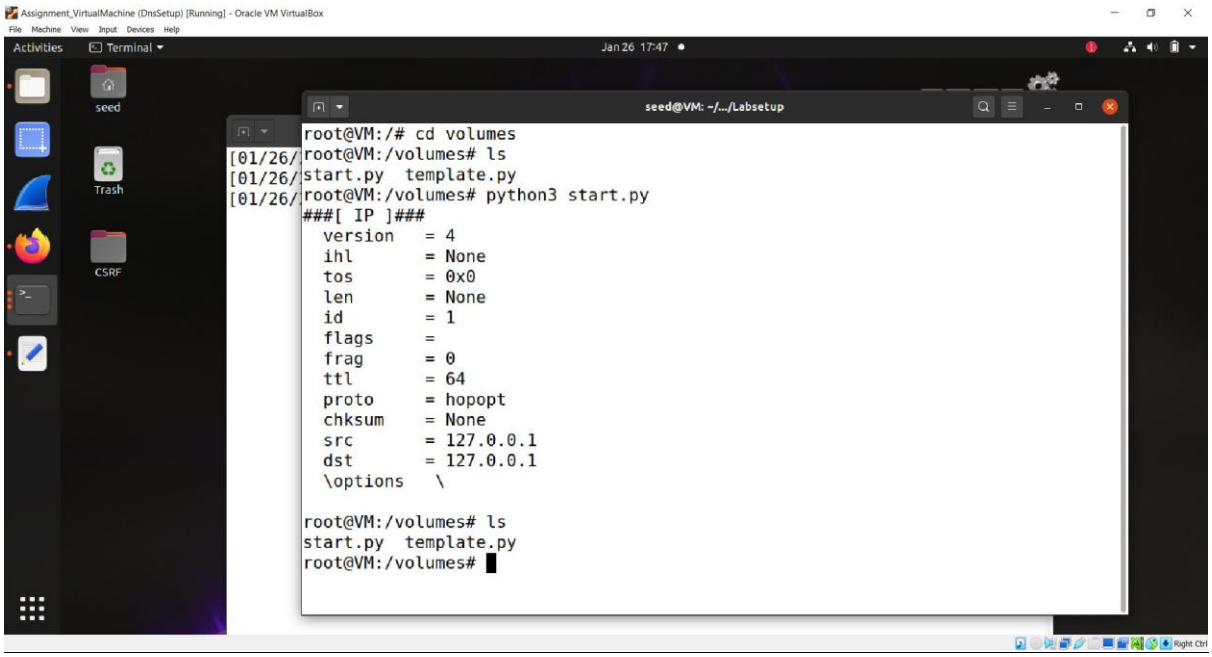


Lab Task Set 1: Using Scapy to Sniff and Spoof Packets

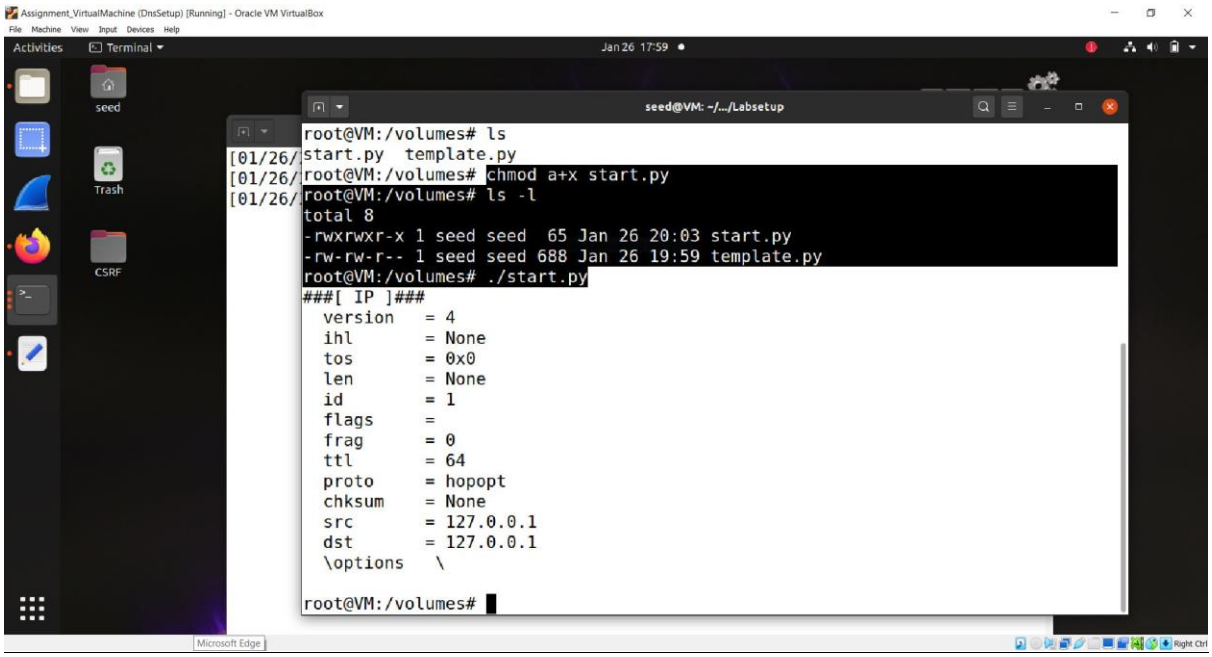
We create file name: start.py file and add below code to it in volumes folder.
make IP object a and using show() display.



To run start.py file following commands are used: python3 start.py
Here we can see output IP object is displayed.



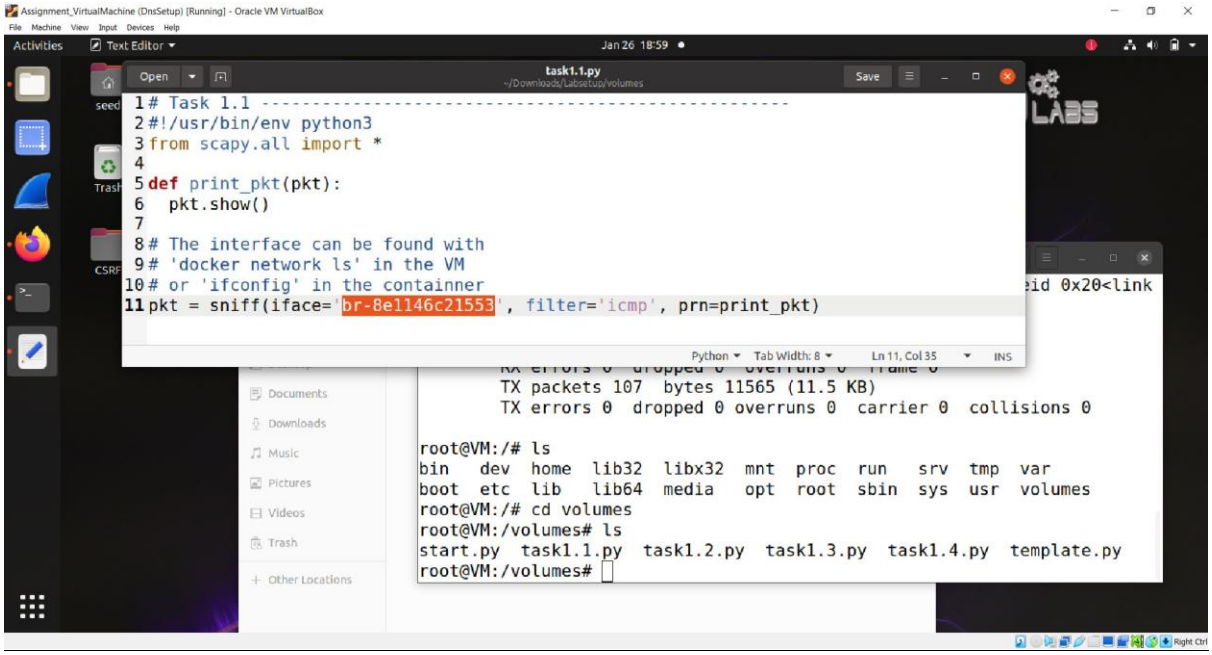
We can also use command: chmod a+x start.py and ./start.py



Task 1.1: Sniffing Packets

Create task1.1.py file and add the code displayed below.

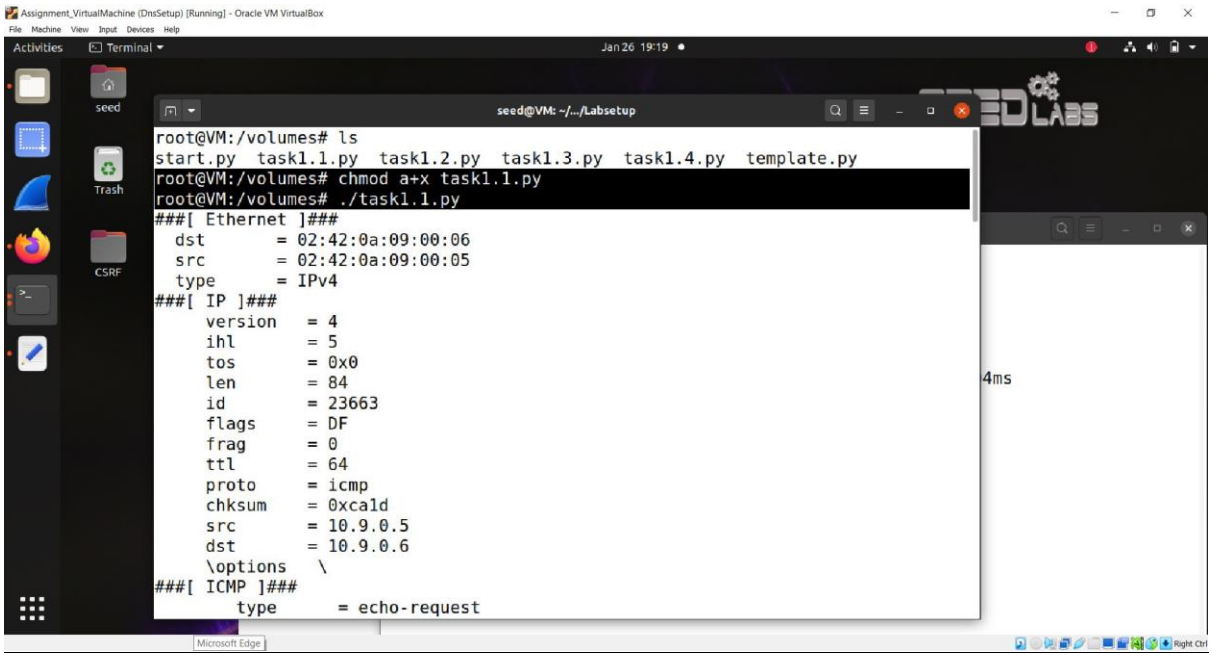
here we use sniff function with iface value as our interface and filter as icmp and function call to print_pkt() which displayed pkt which we created.



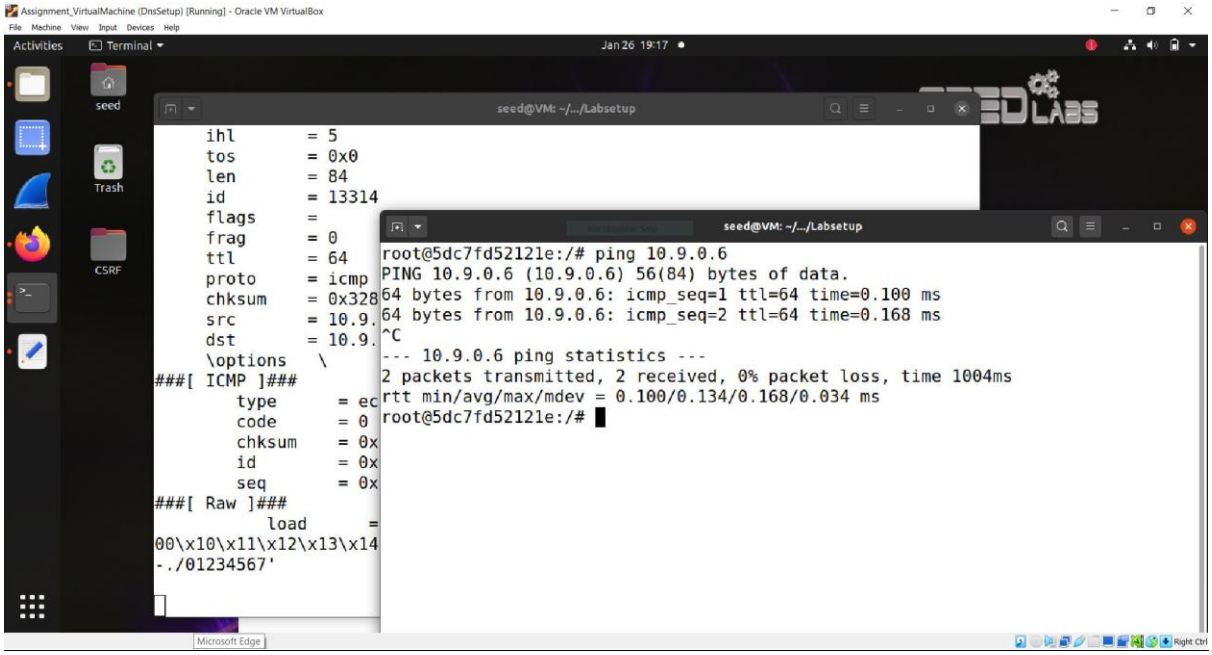
run task 1.1 using command on seed-attacker

chmod a+x task1.1.py

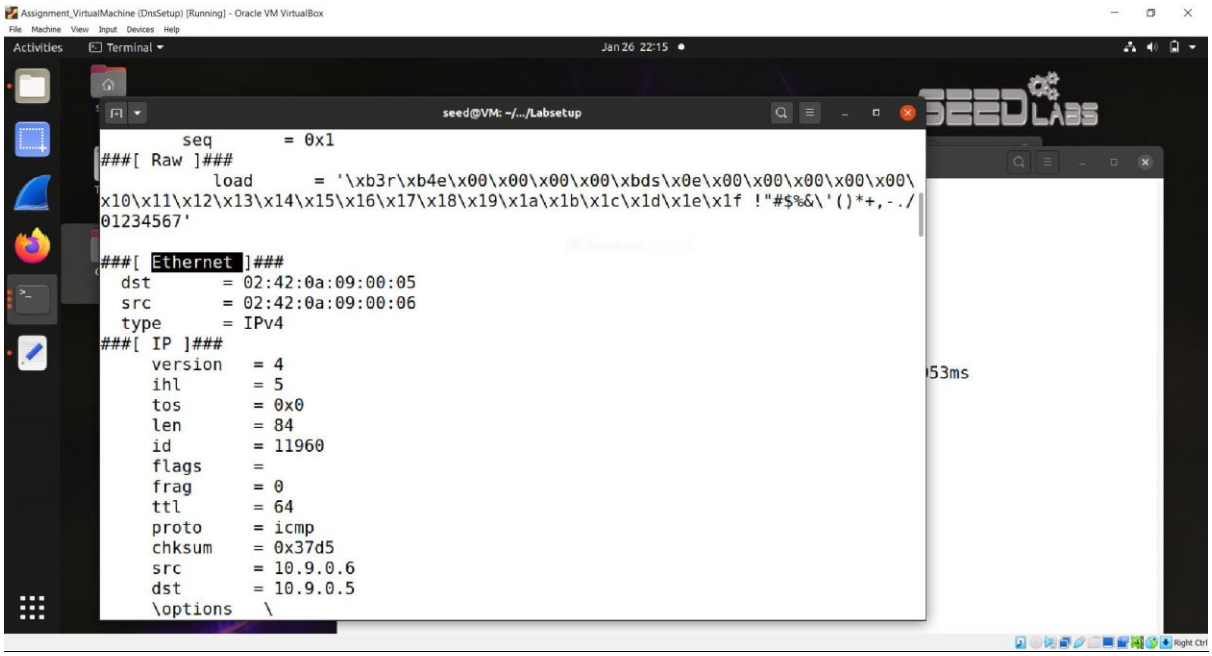
./task1.1.py



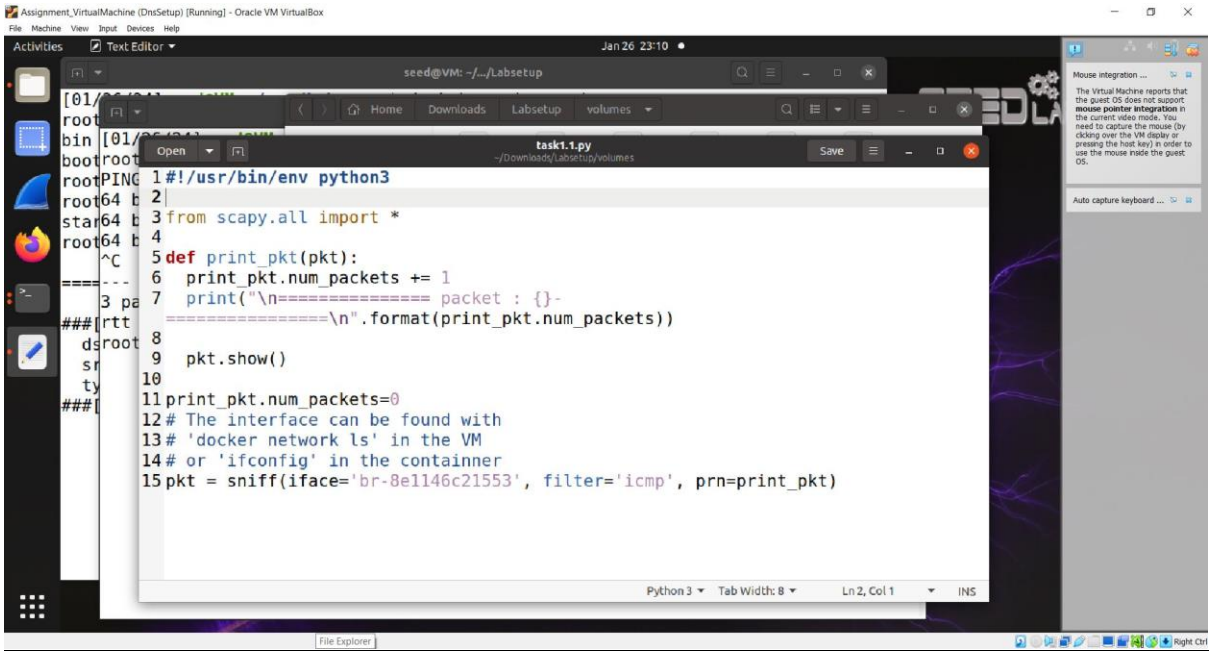
Now , go to host A and listen for host B using ping command.2 received packets can be seen.



If we go and check on from where we attacked we get all objects captured.



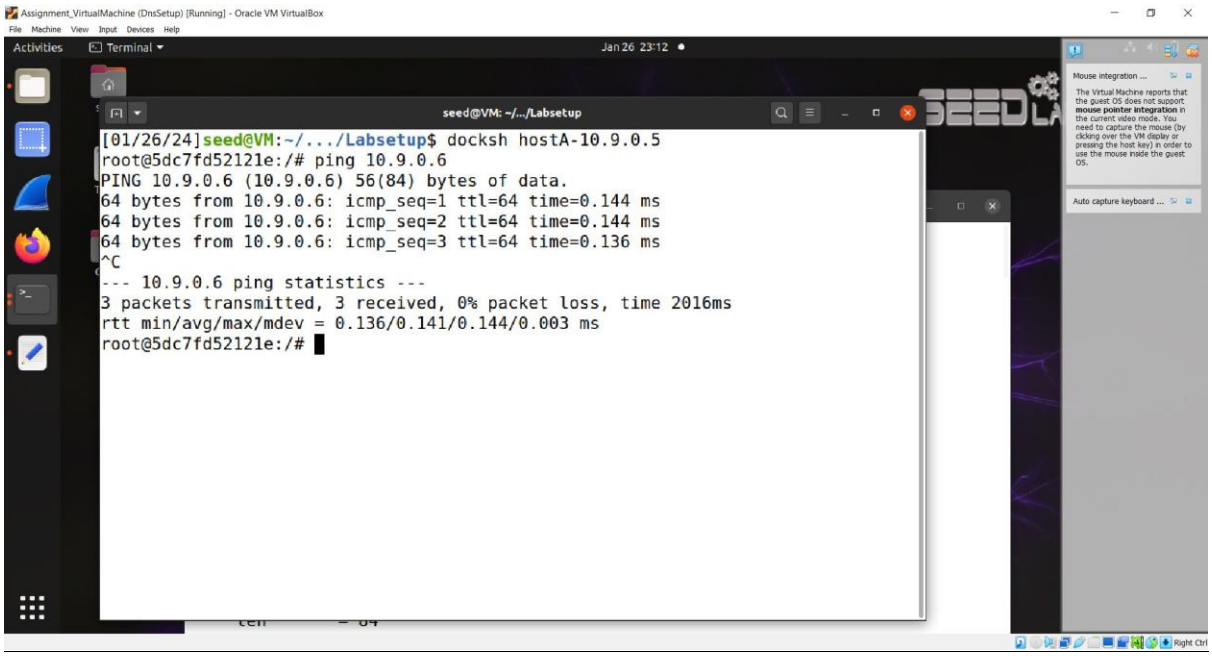
Now we make changes in code to get packet information printed.



again we run it on attacker and listen from host A we ping to hostB:

we observe following :

3 packets transmitted and 3 received.



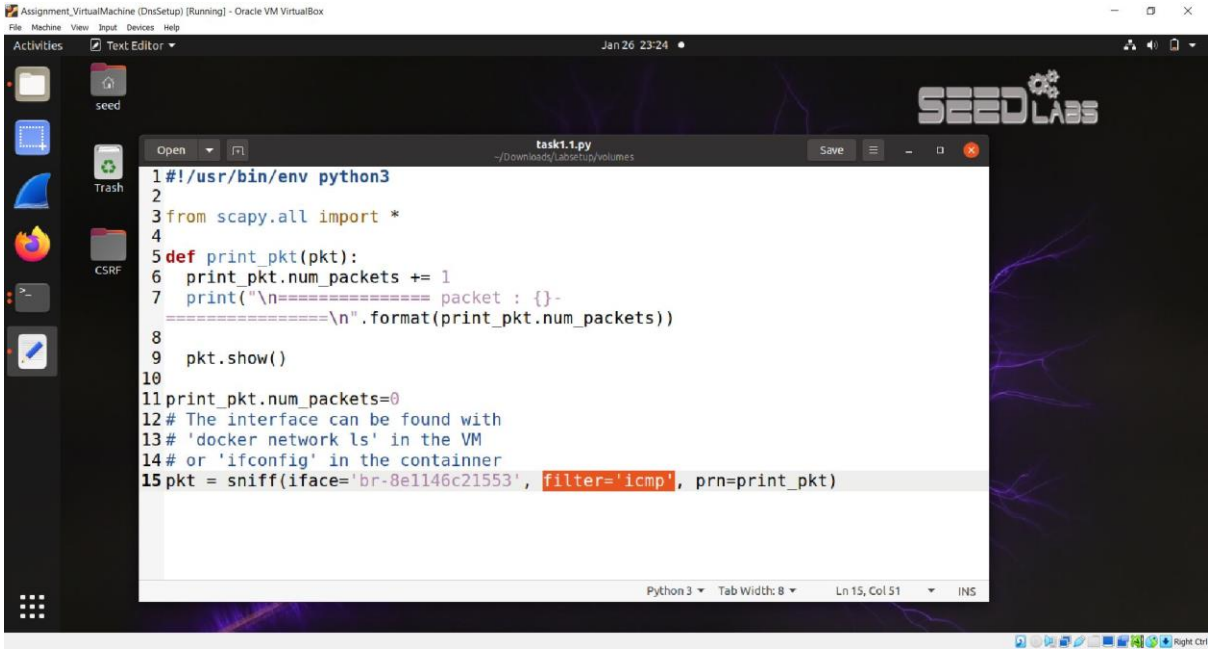
Here, we can see after seeing the results of running modified code to print packet number there are exactly 6 packets in total.

Task 1.1B.

Please set the following filters and demonstrate your sniffer program again (each filter should be set separately):

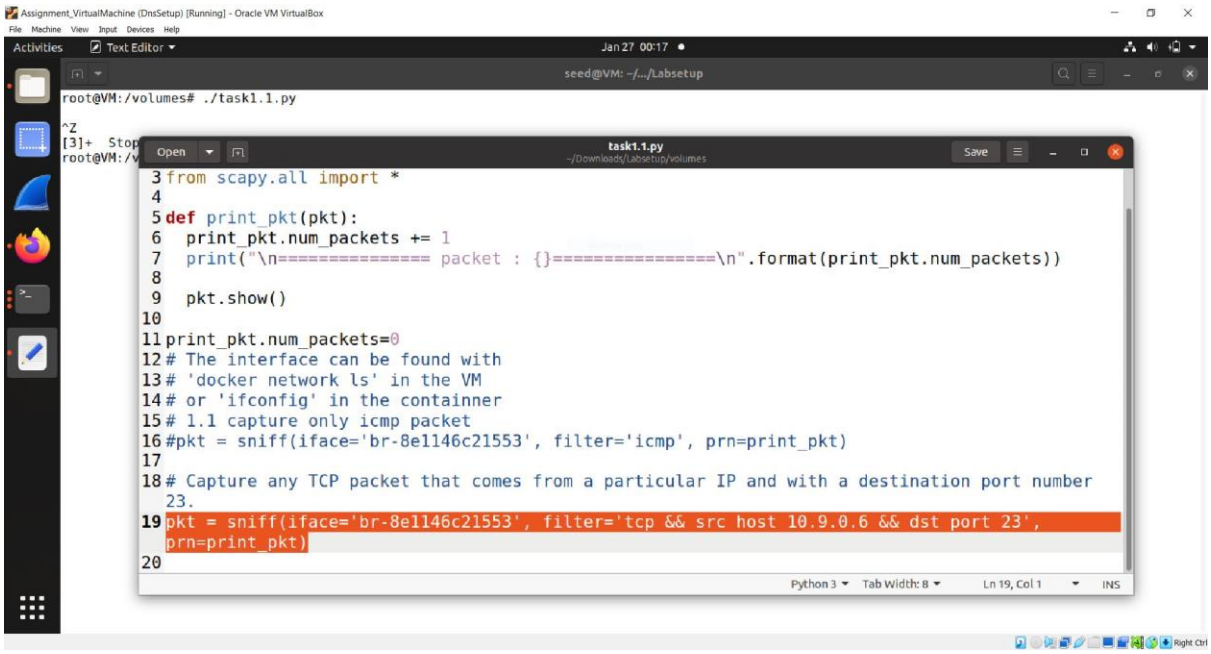
• Capture only the ICMP packet

Task1.1.py file has filter set as “icmp”

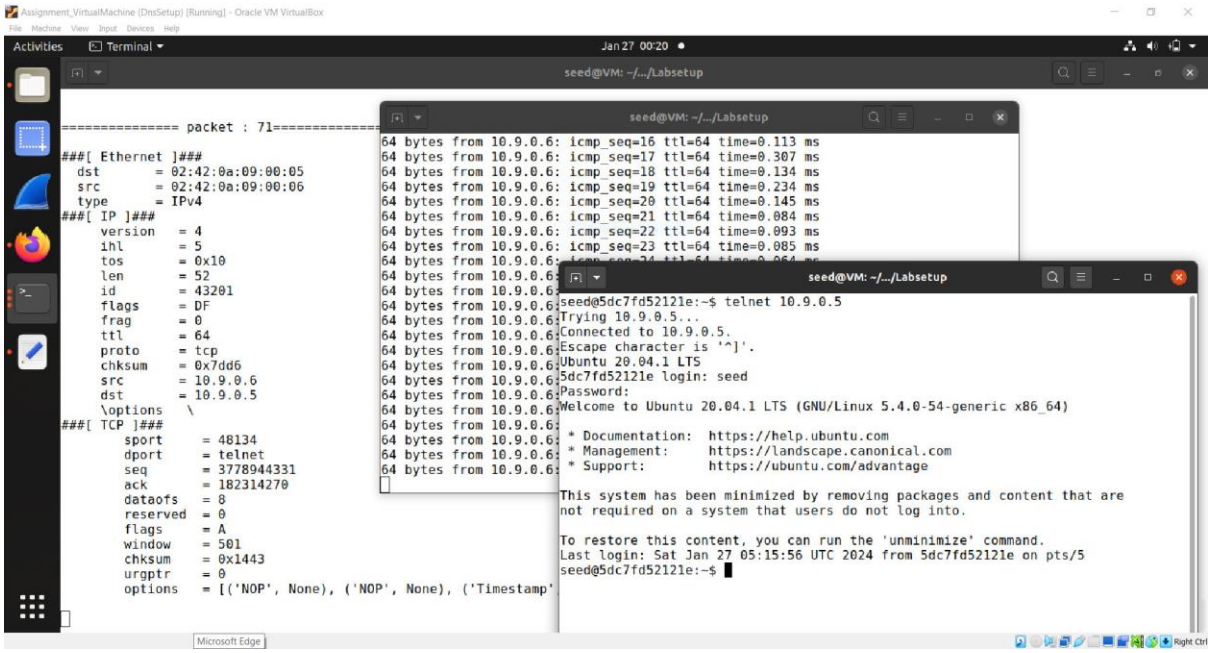


• Capture any TCP packet that comes from a particular IP and with a destination port number 23.

Here we set tcp protocol, src host as 10.9.0.6 and destination port is 23.
After checking found port 23 corresponds to telnet.

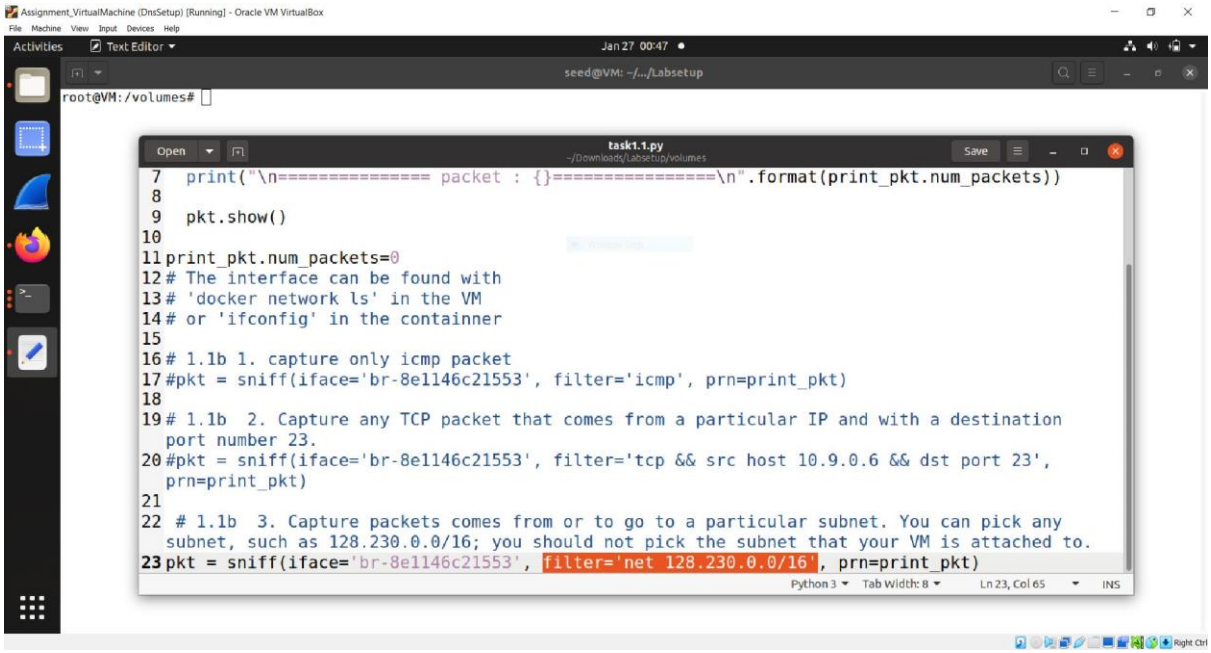


Here we can see seed-attacker gets src= 10.9.0.6 and dst=10.9.0.5
Host B uses telnet 10.9.0.5 and attacker gets information of TCP protocol with Ips and dport as telnet which is port no. 23

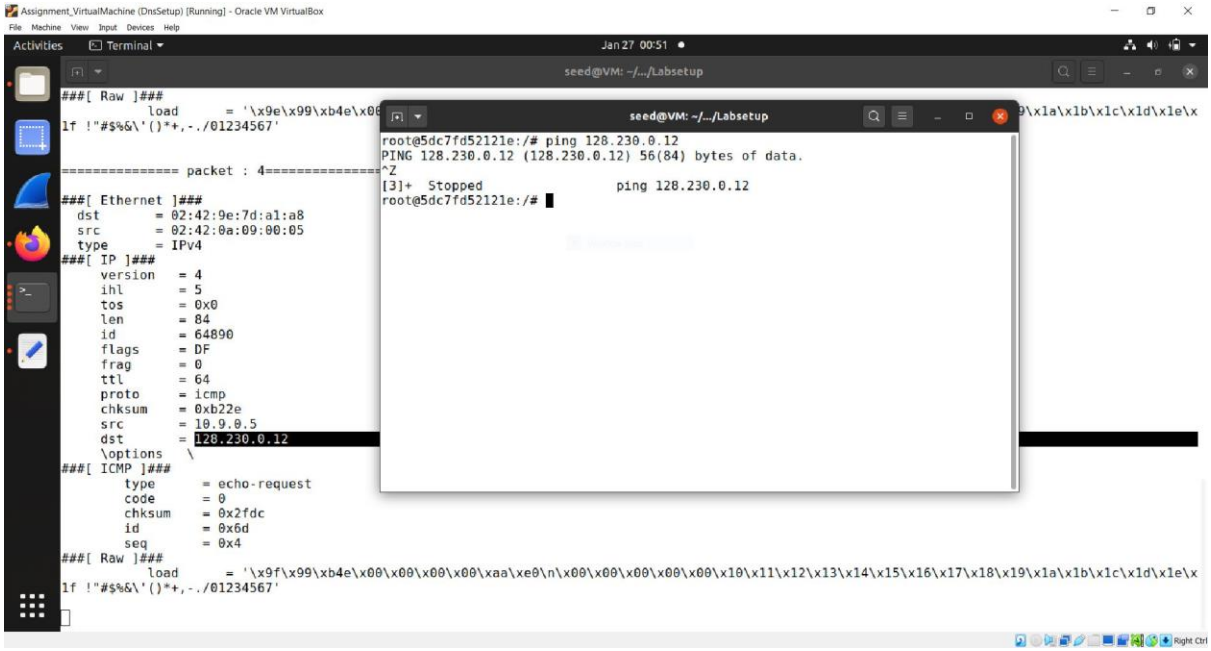


• Capture packets comes from or to go to a particular subnet. You can pick any subnet, such as 128.230.0.0/16; you should not pick the subnet that your VM is attached to.

Added “net 128.230.0.0/16” in the filter to start capturing from the particular subnet mentioned.

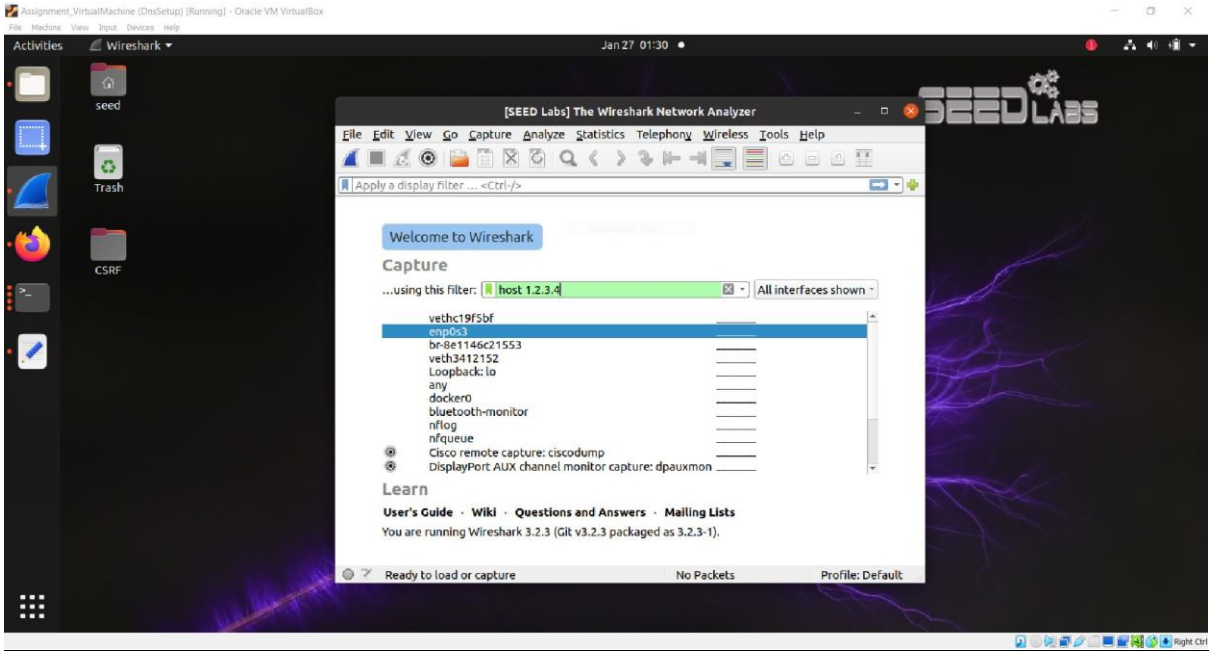


Now we run from attacker this script. And from host A we ping subnet.
We can see ping 128.230.0.12 from Host A and attacker is able to capture packets from Host A 10.9.0.5 to destination we pinged i.e. 128.230.0.12

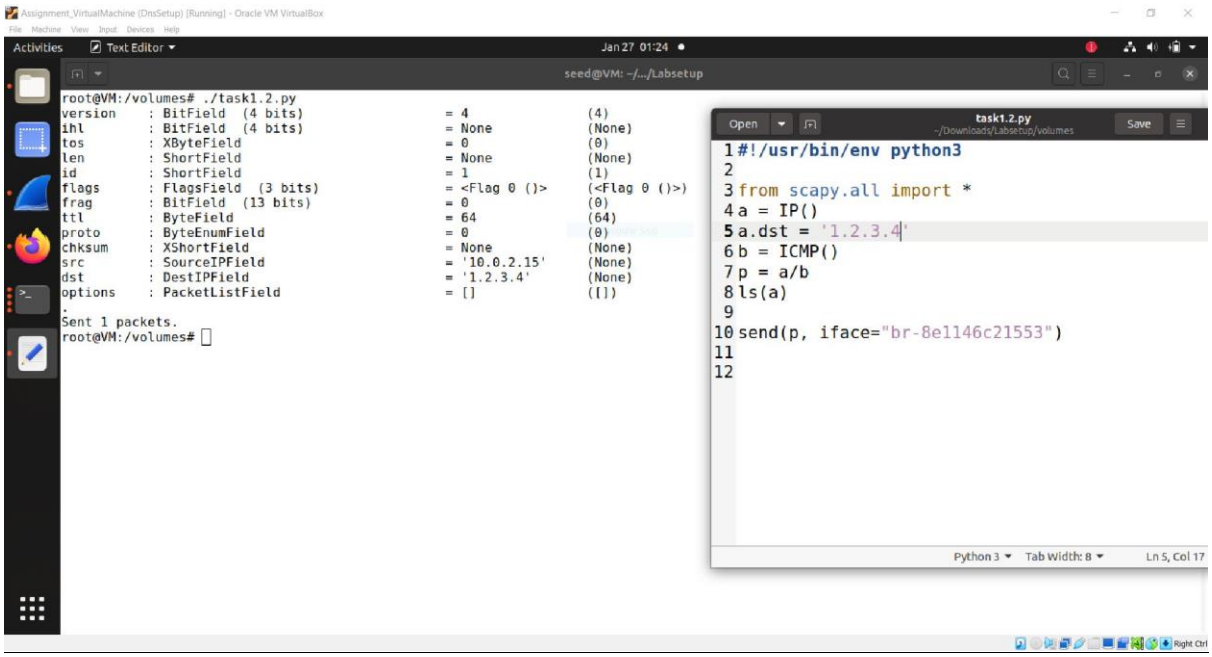


Task 1.2: Spoofing ICMP Packets

We are capturing from our interface 10.0.2.15 (enp0s3)and host 1.2.3.4



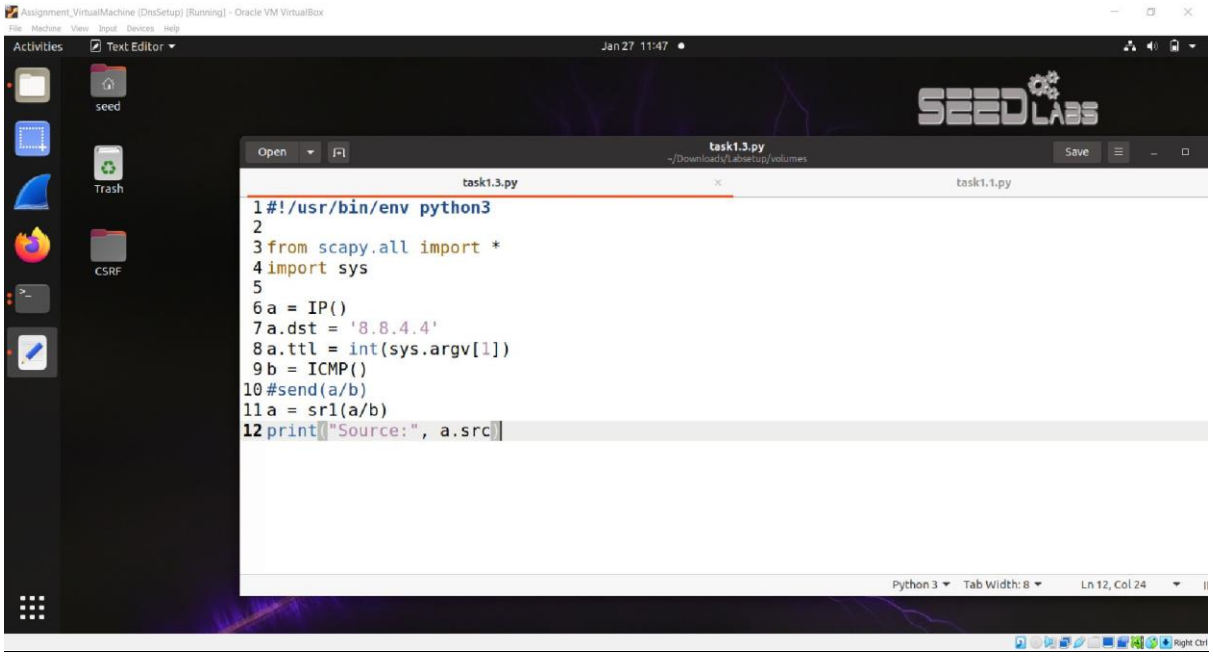
Now we add interface name in send () and destination as 1.2.3.4



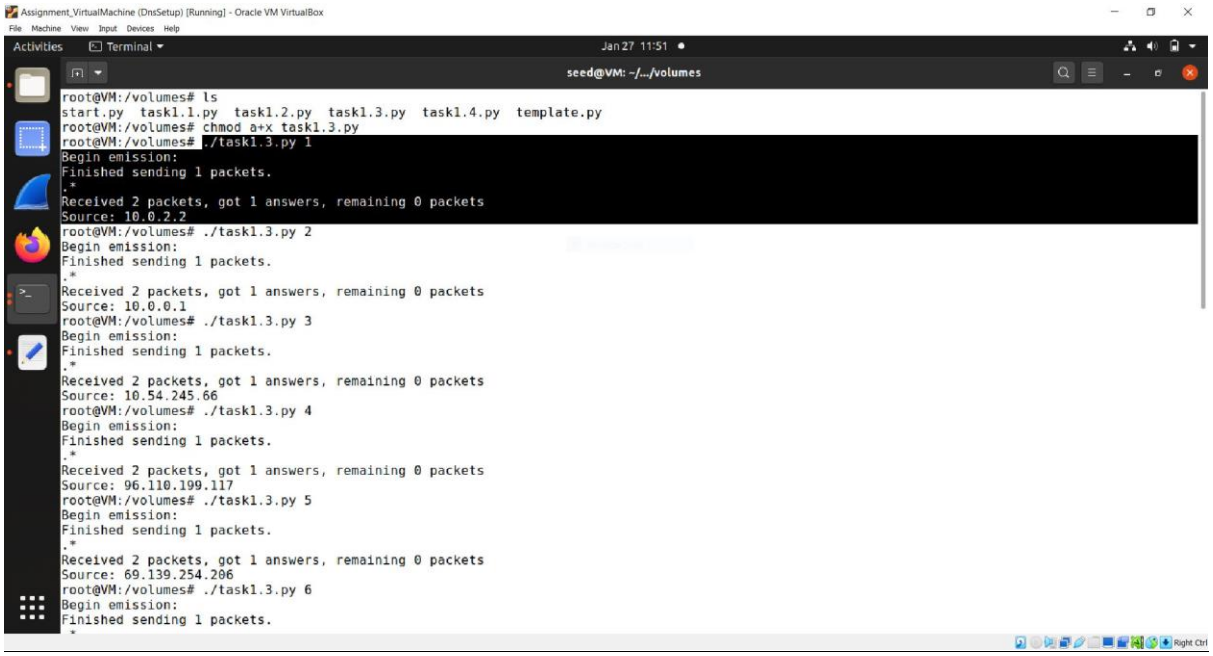
Task 1.3: Traceroute

Here for tracing route we write this program: set destination IP = 8.8.4.4 and take from command line TTL (Time to Live). We make use of sr1() from scapy which returns one packet that answered the packet sent. In this way, we can track the route of packets until we get destination.

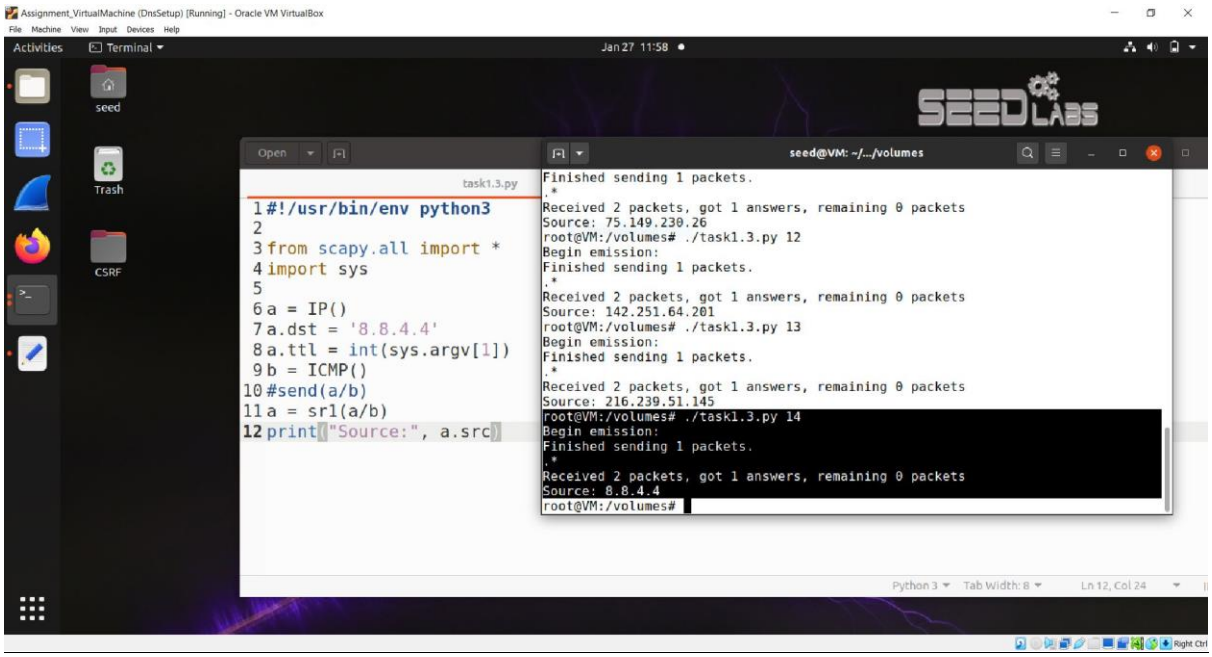
Also print the source, so we can verify the route.



After running the code we see: src is 10.0.2.2 and we keep giving argument from command line until we reach destination.

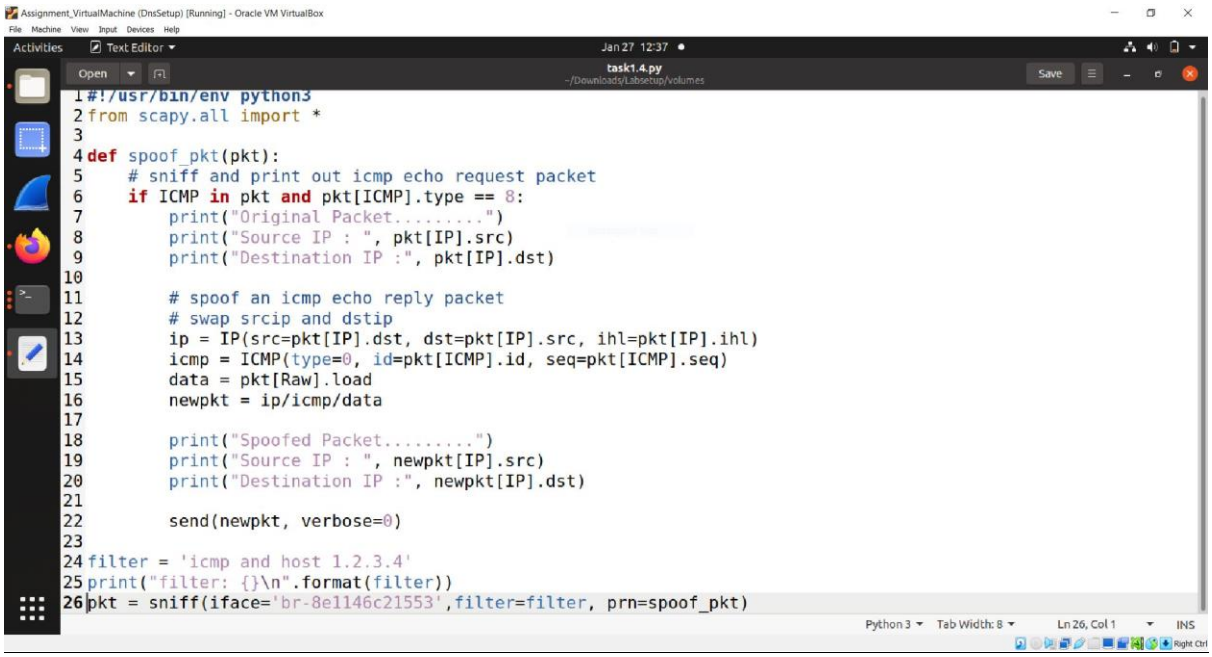


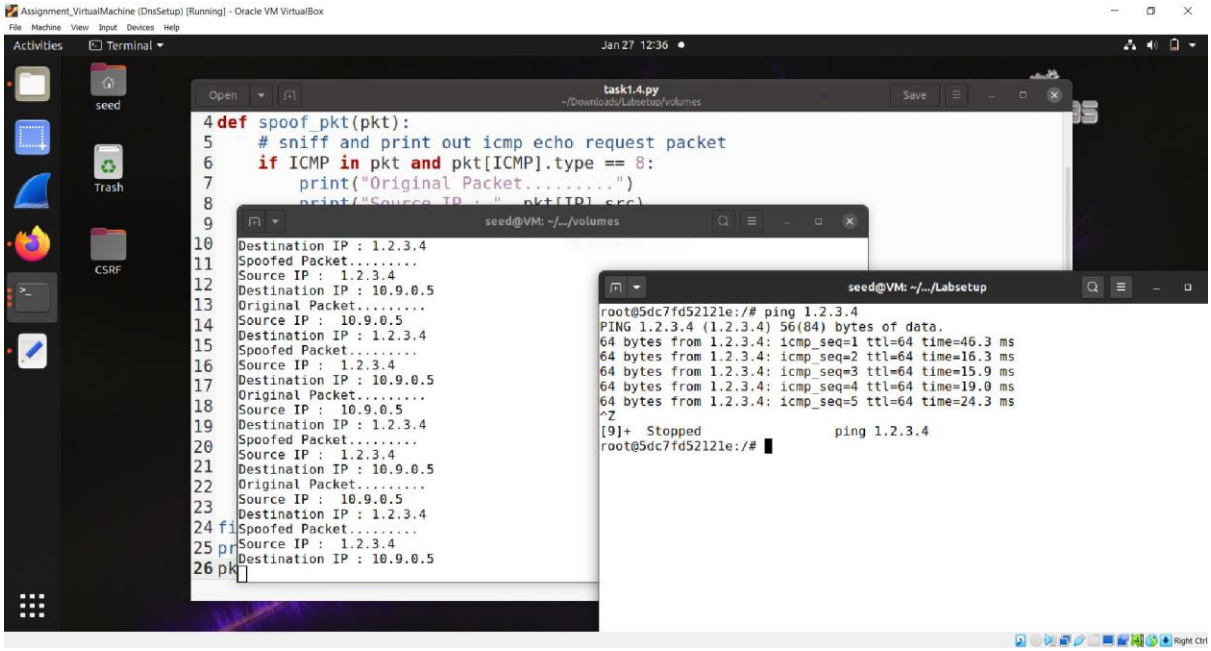
We can see it reached destination after giving 14 as TTL.



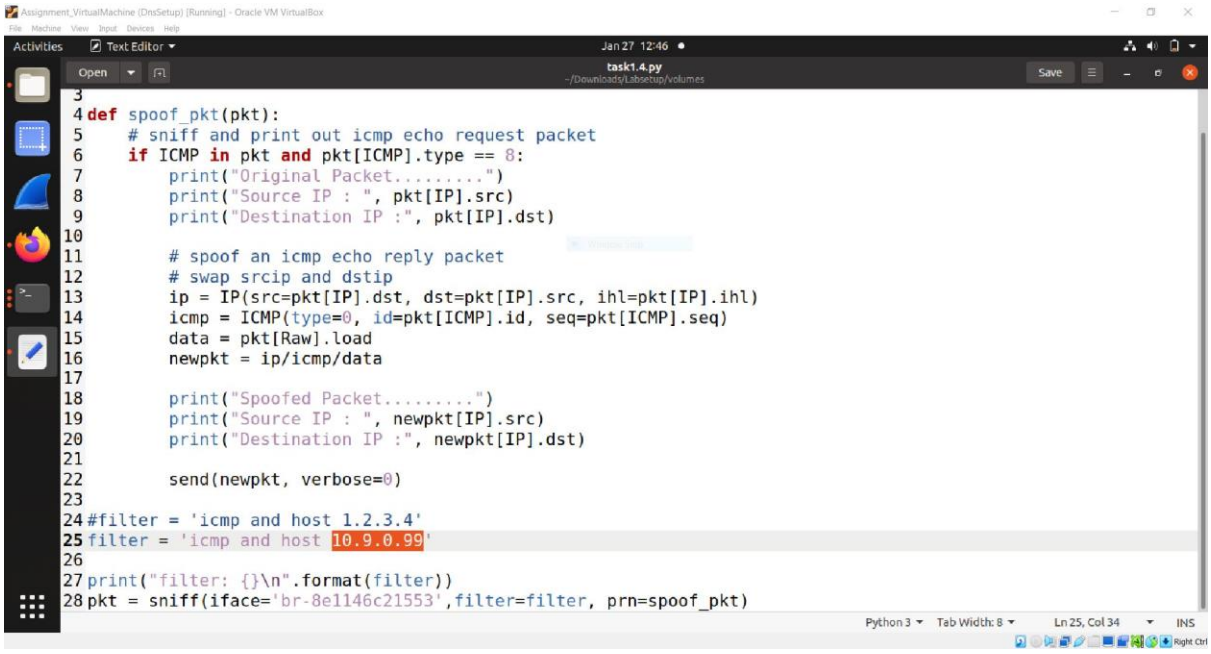
Task 1.4: Sniffing and-then Spoofing

ping 1.2.3.4 # a non-existing host on the Internet

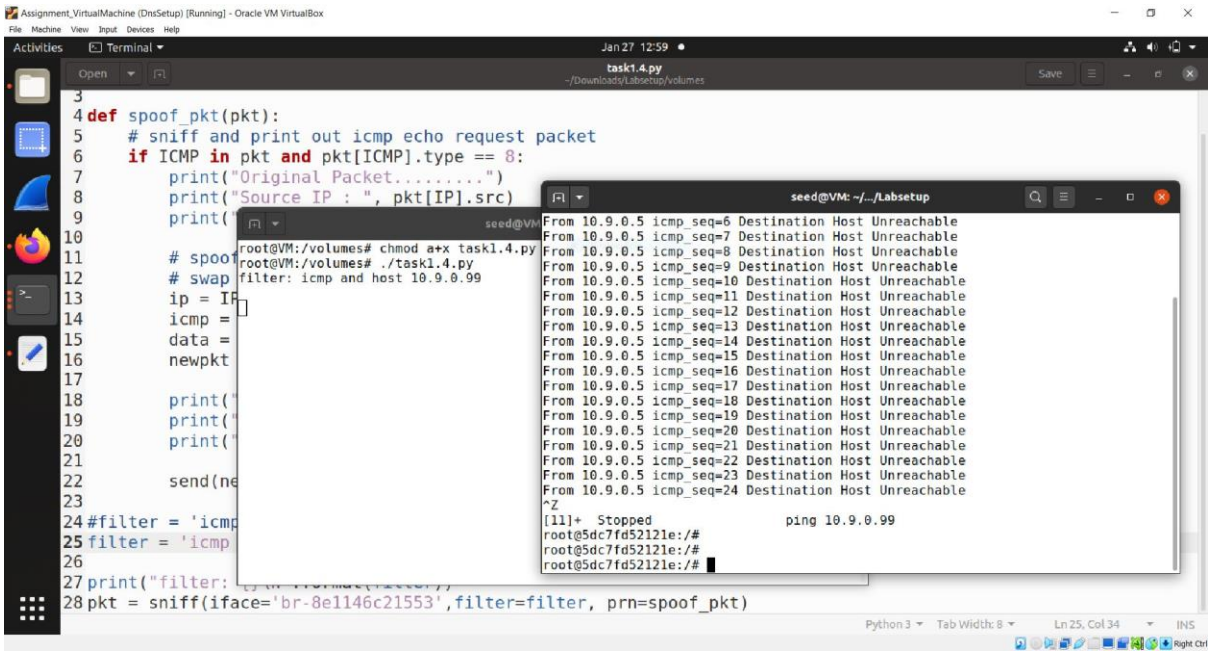




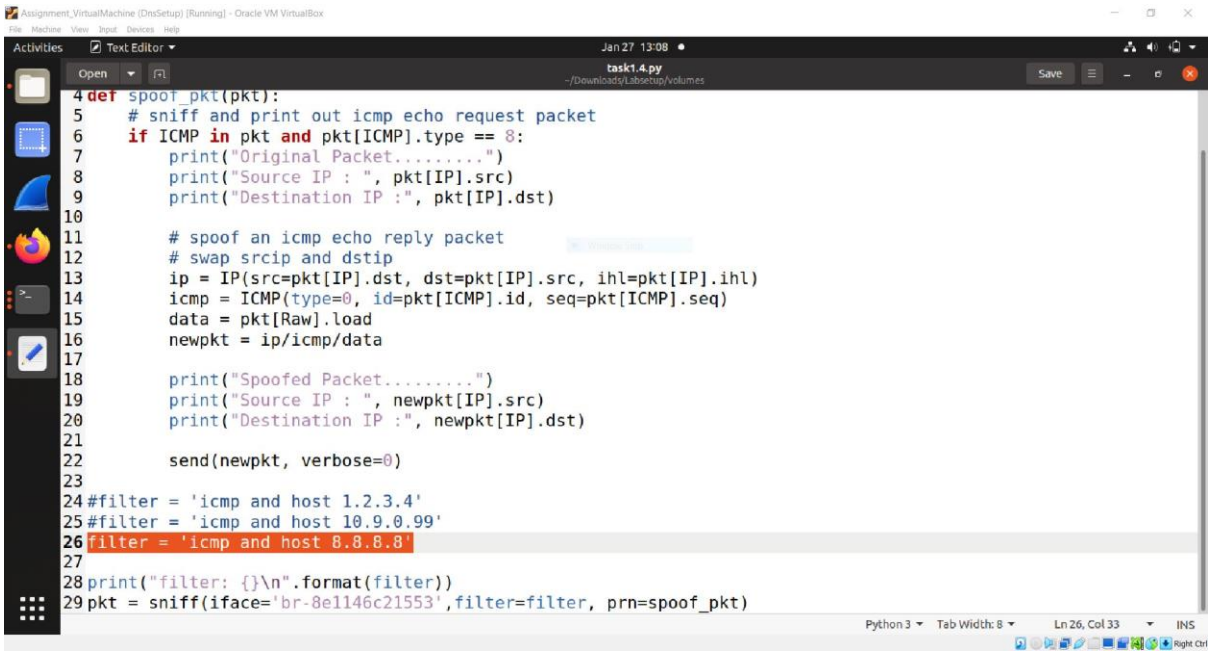
ping 10.9.0.99 # a non-existing host on the LAN



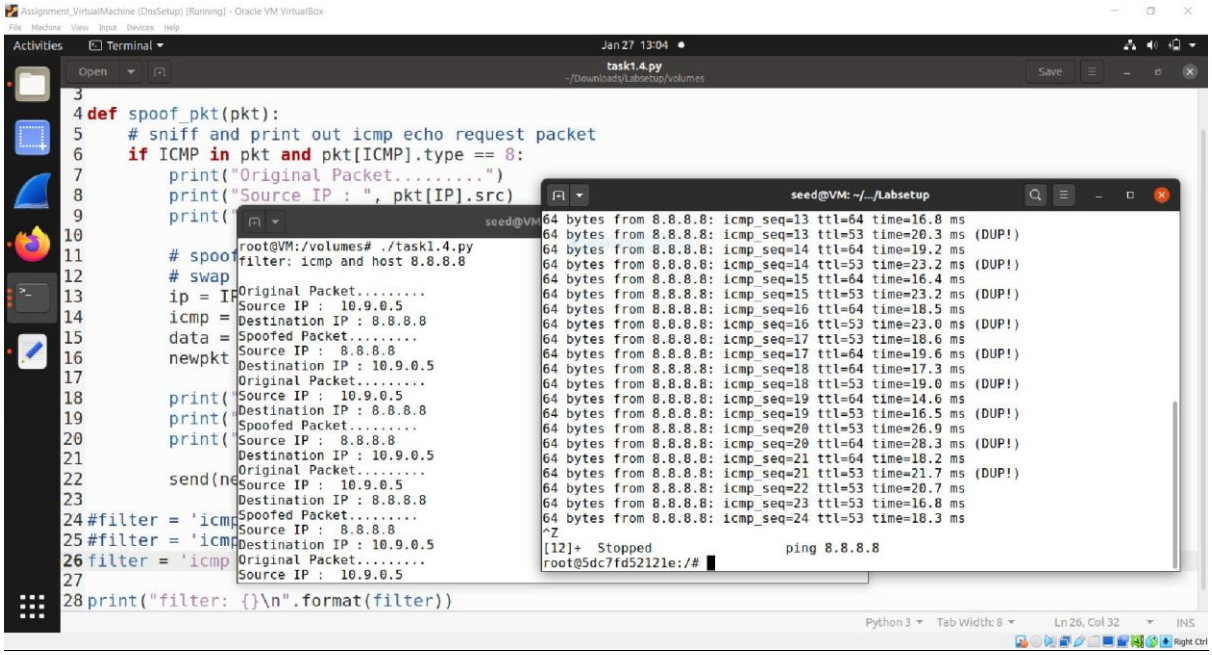
results: Destination Host is not reachable in this case



ping 8.8.8.8 # an existing host on the Internet
Code changes to added IP 8.8.8.8 as host



We get results as duplicate packets received i.e. DUP! we do not know which one is spoofed at host side but at attacker we have printed information.



Task 2.1A: Understanding How a Sniffer Works

Question 1. Please use your own words to describe the sequence of the library calls that are essential for sniffer programs. This is meant to be a summary, not a detailed explanation like the one in the tutorial or book.

For sniffer program in c using pcap api following is the sequence of library calls:

1) using pcap_open_live() function to capture packet .
2) using pcap_compile() to compile filter expression to convert it into BPF code (Berkely filter packet)
3) using pcap_setfilters() to set the compiled filters to capture only specific packets based filter expression.

4) using pcap_loop() to capture packets with few parameters stating when to stop loop or otherwise.

Question 2. Why do you need the root privilege to run a sniffer program? Where does the program fail if it is executed without the root privilege?

Without root privileges it will not execute and will display operation not permitted error.

Question 3. The value 1 of the third parameter in pcap open live() turns on the promiscuous mode (use 0 to turn it off). Can you tell the difference when this mode is on and off ?

Promiscuous mode allows a network interface to receive and process all incoming traffic, regardless of whether it is intended for the interface's MAC address or not.

To turn on promiscuous mode in a sniffer program, you set the third parameter of pcap_open_live() to 1. To turn it off, you set the parameter to 0