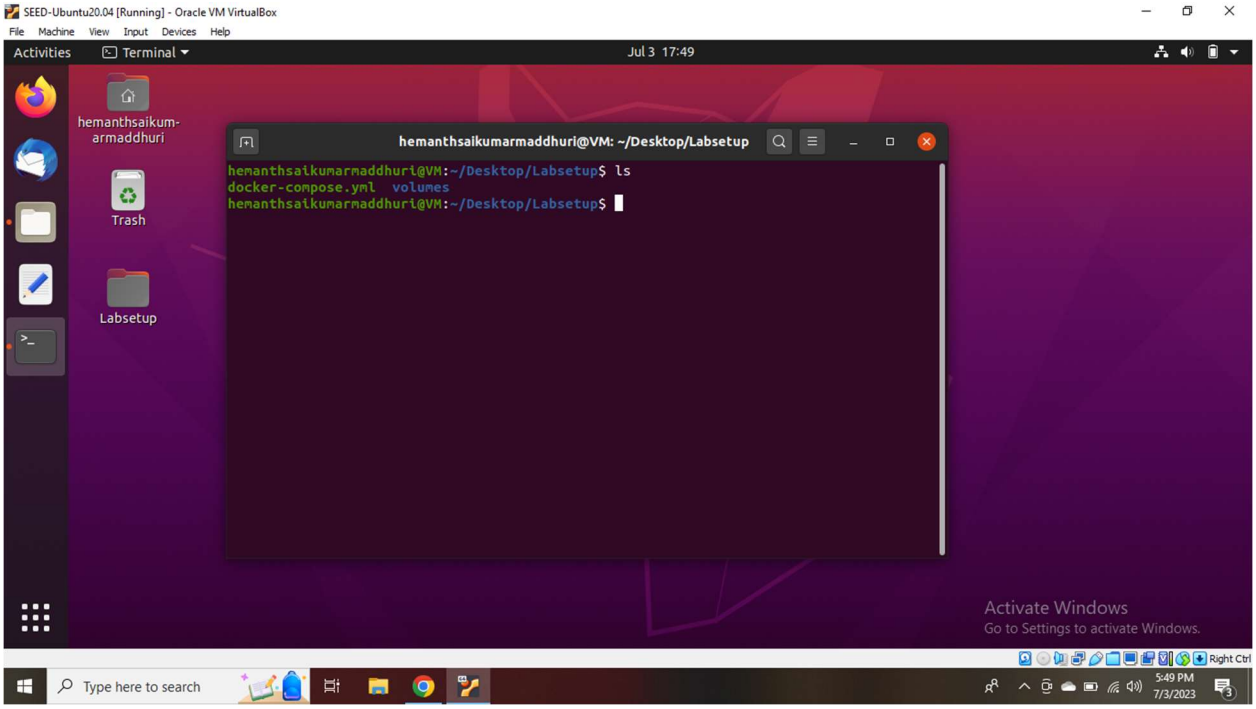
Information and Networking Security

Quiz - 5

Name: Hemanth Sai Kumar Maddhuri **ID:** 999902480

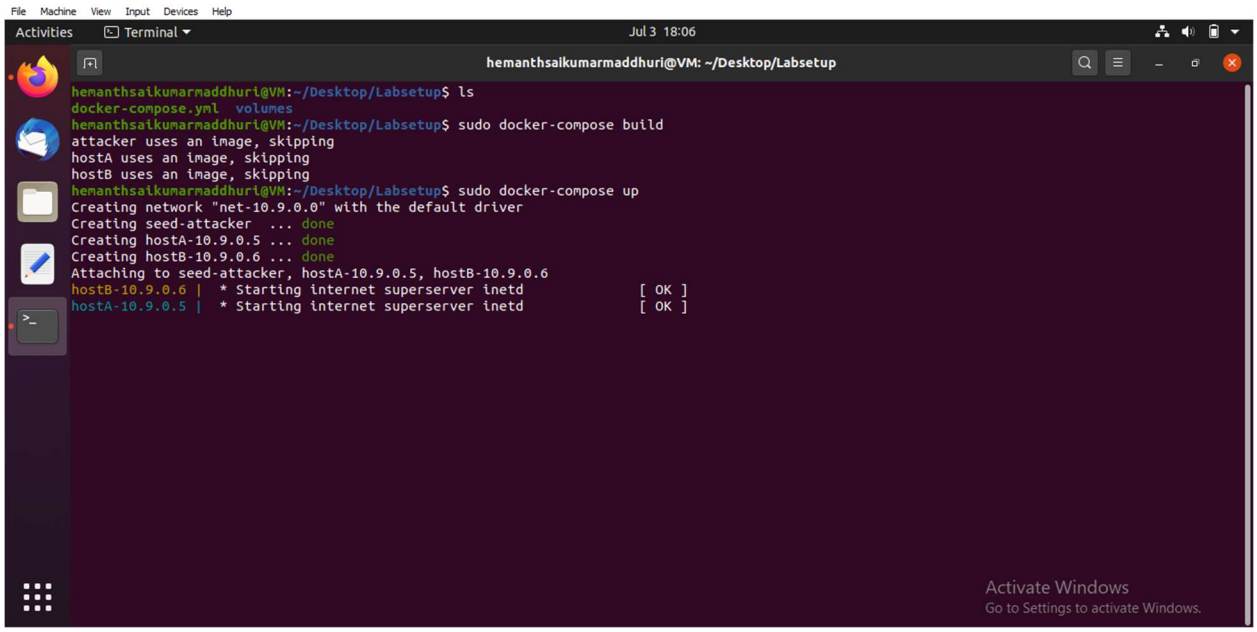
As Instructed I have downloaded Labsetup files required for Packet Sniffing and Spoofing Lab from the following URL https://seedsecuritylabs.org/Labs_20.04/Networking/Sniffing_Spoofing/.



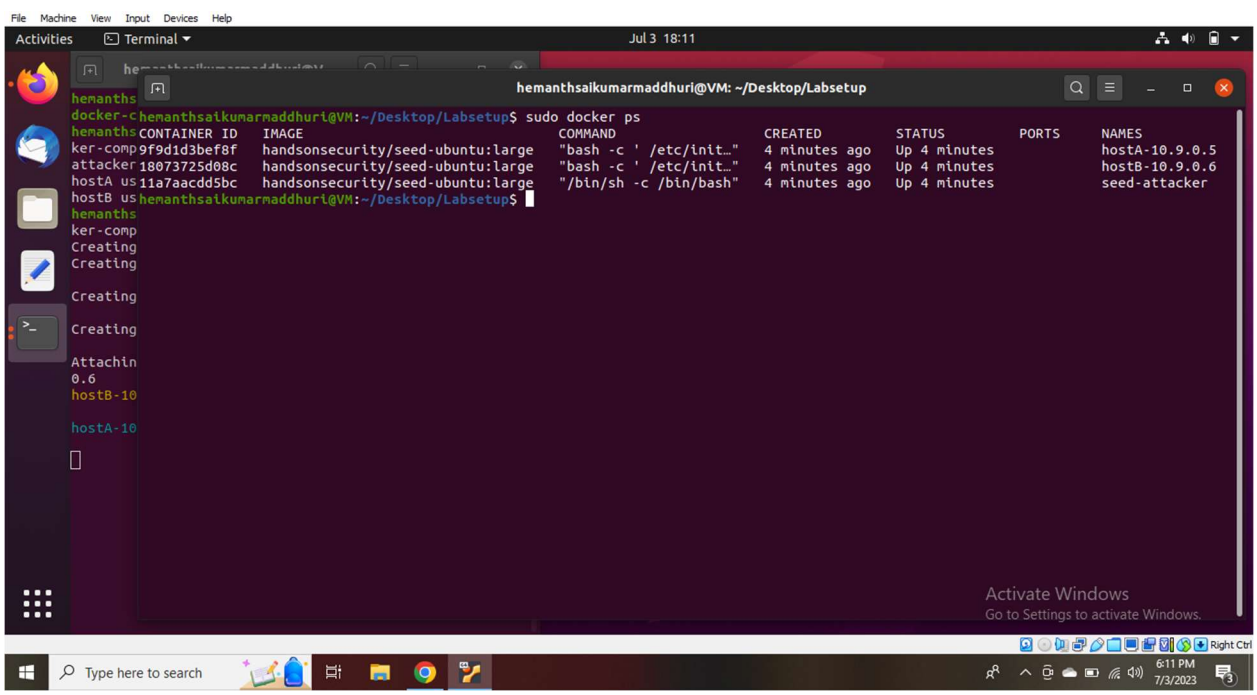
2 Environment Setup using Container

2.1 Container Setup and Commands

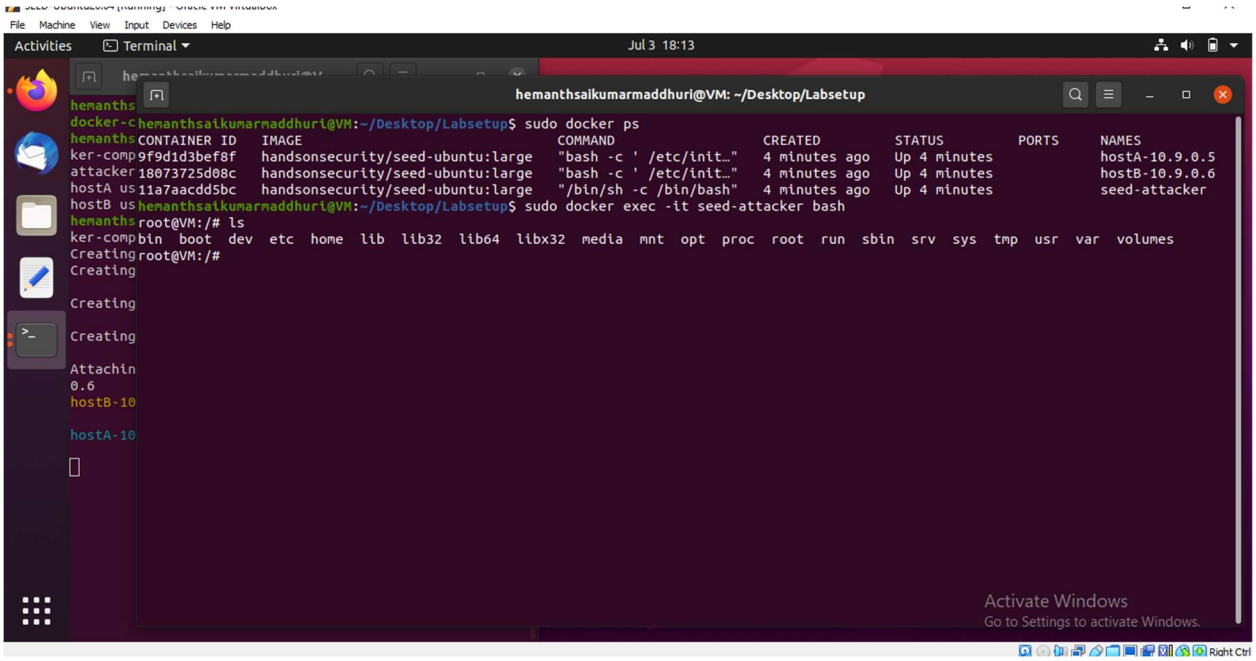
Here we are building the docker and turning it up using the commands shown in the screenshot below.



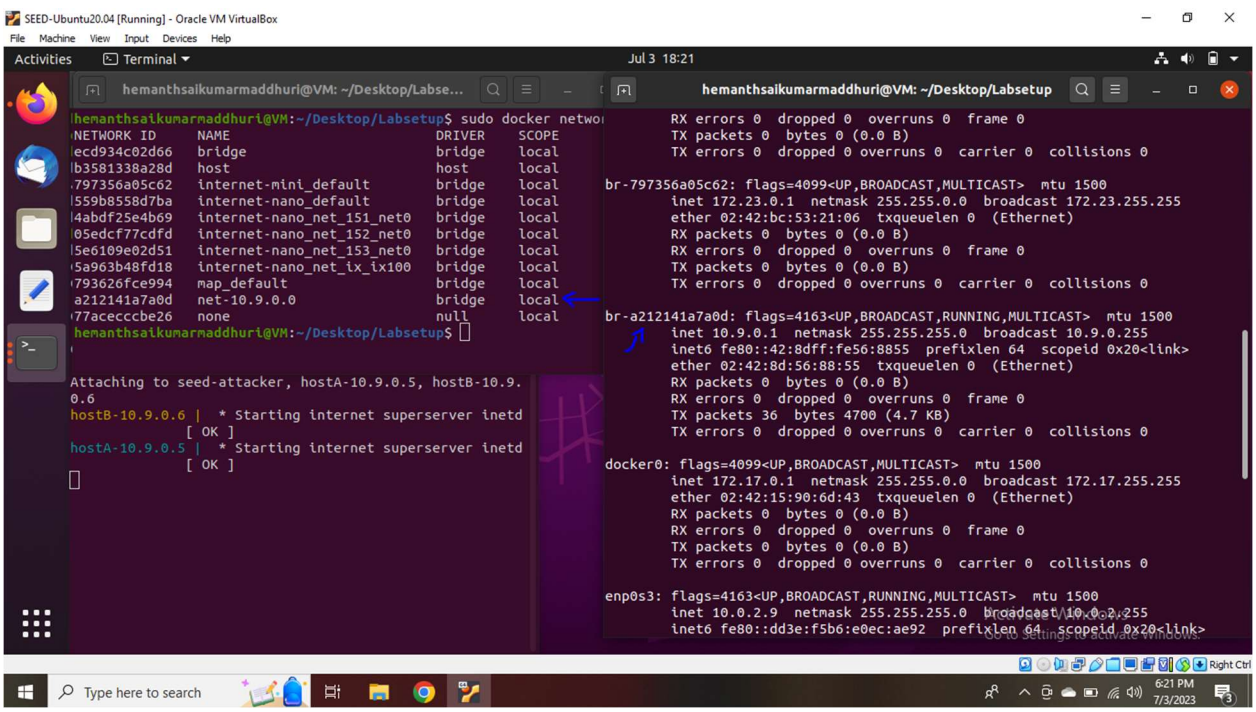
Simultaneously in the other terminal we run the command "**sudo docker ps"** to find out the ID's of the containers.



Using the displayed docker names we now try to get into one of the host machines using the command **“sudo docker exec -it seed-attacker bash”**. As shown below I have logged in and listed the files present in the host machines.

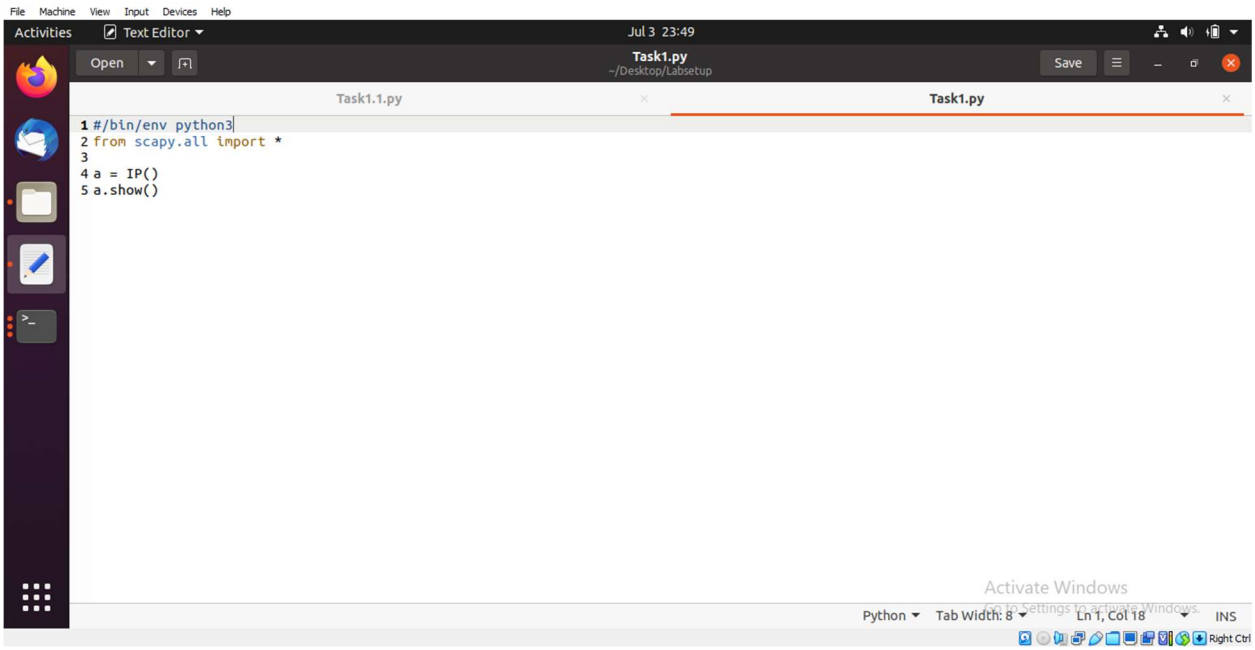


Getting the bridge ID for the host IP and in my case, it is **“br-a212141a7a0d”**.

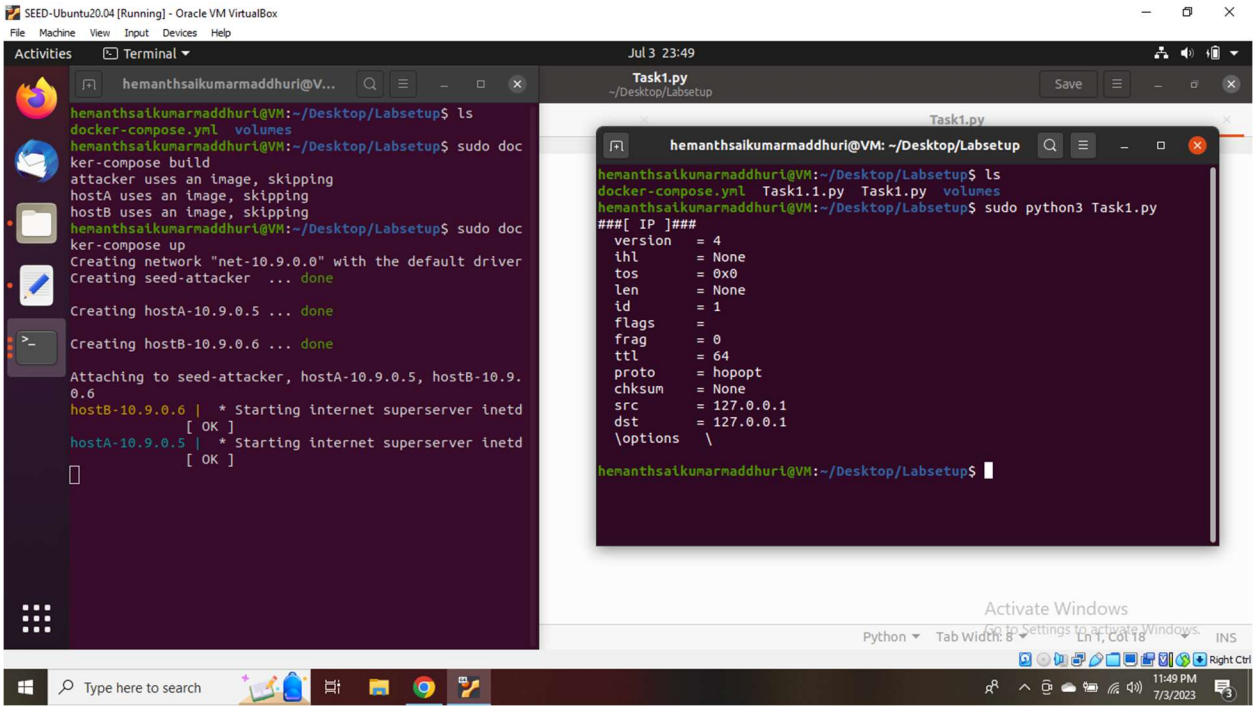


3 Lab Task Set 1: Using Scapy to Sniff and Spoof Packets

The python code for Task 1 is as shown in the figure.

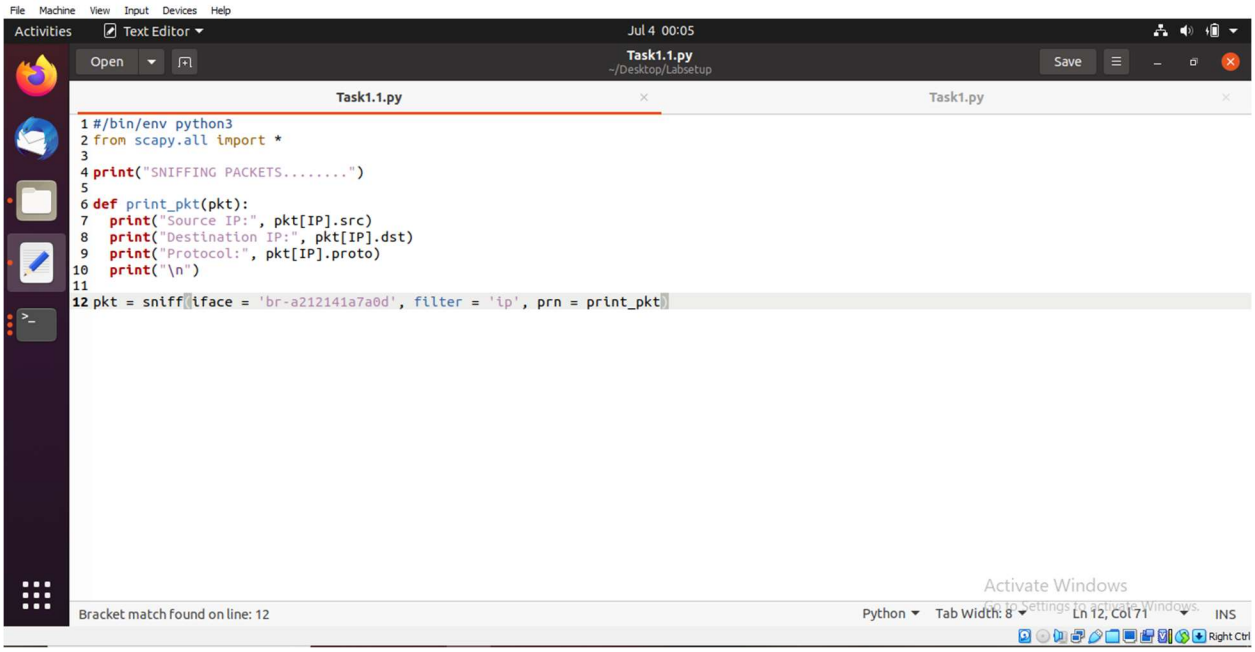


Here we are running the code for Task 1 using the command **"sudo python3 Task1.py"** the information of IP is printed to the command line after the execution of Task1.py.



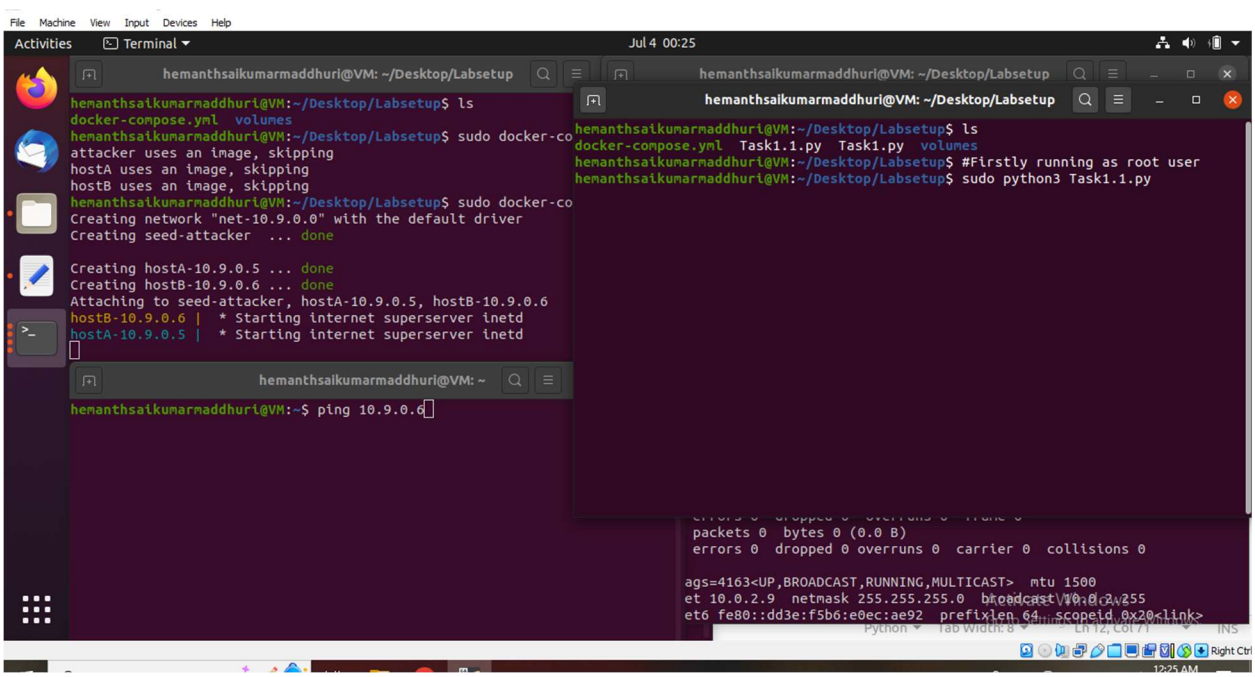
3.1 Task 1.1: Sniffing Packets

The python code for Task 1.1 is as shown below.

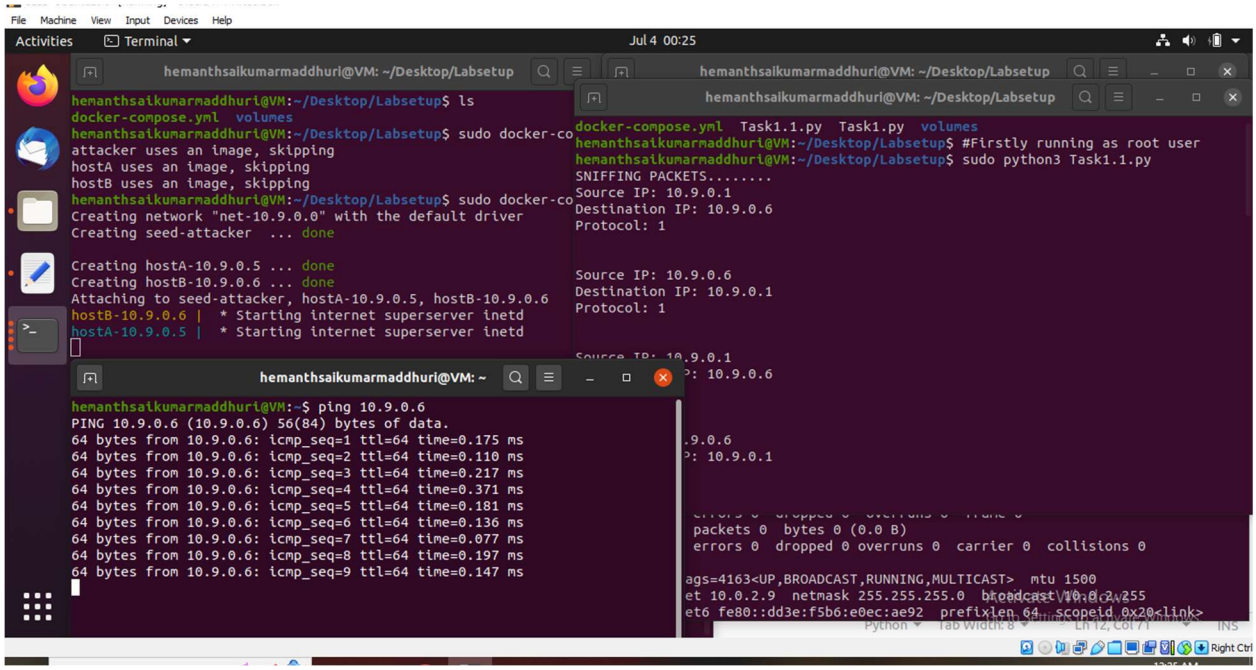


Task 1.1 A

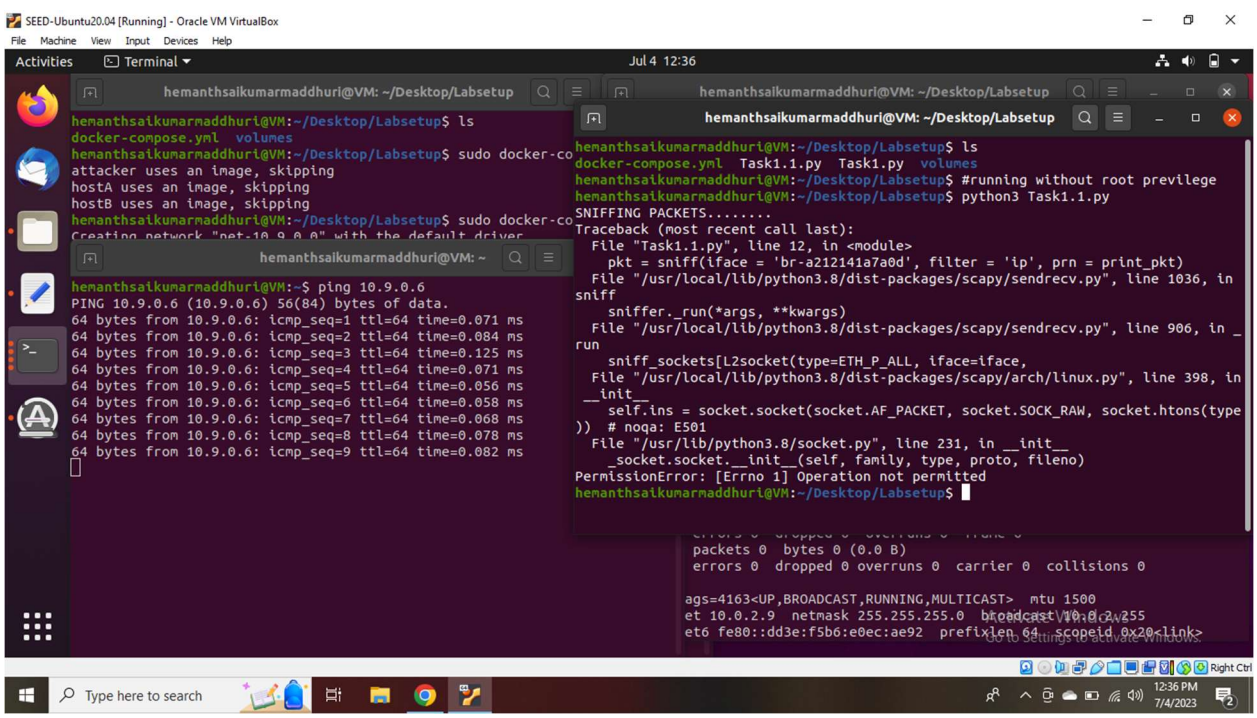
Here we are running the python file required for Task 1.1 as a root user, using the command **"sudo python3 Task1.1A.py"** and in the other terminal we are trying to create some traffic by using the command **"ping 10.9.0.0"**.



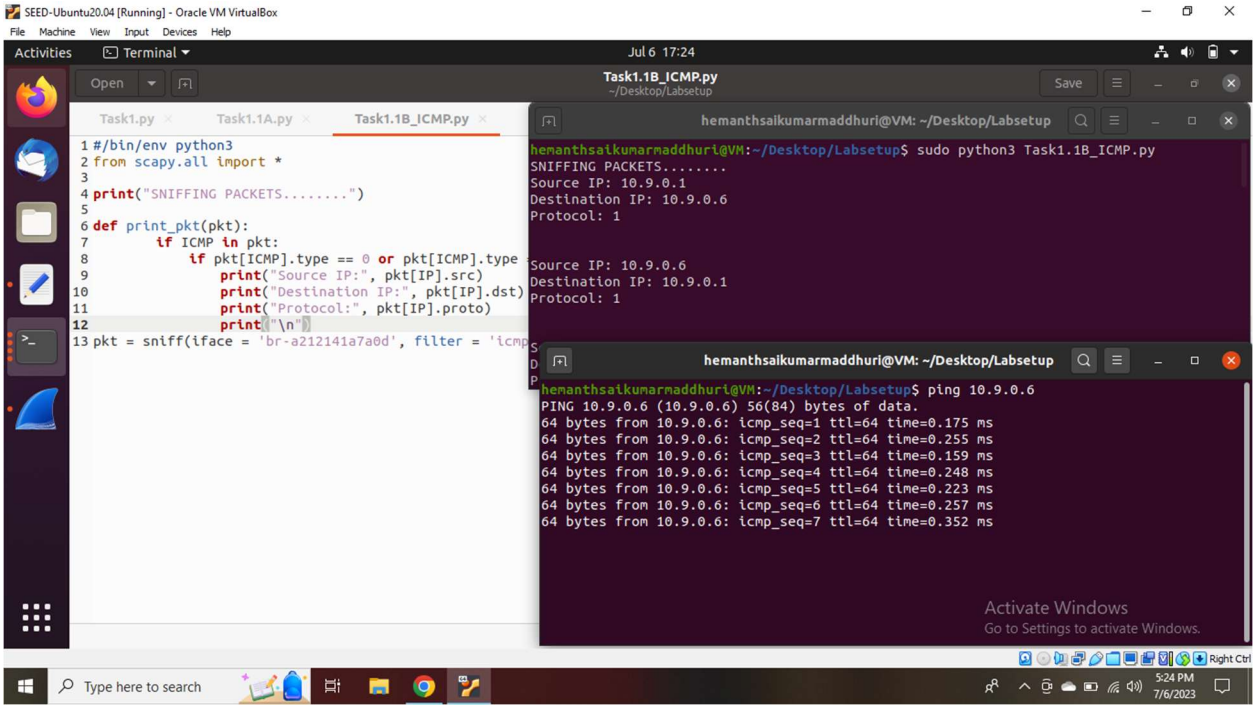
After the execution of the above commands, we see the output as shown in the screenshot below. We can see that the packet information such as source IP, destination IP and protocol used is being printed to command line when we run python file as root user.



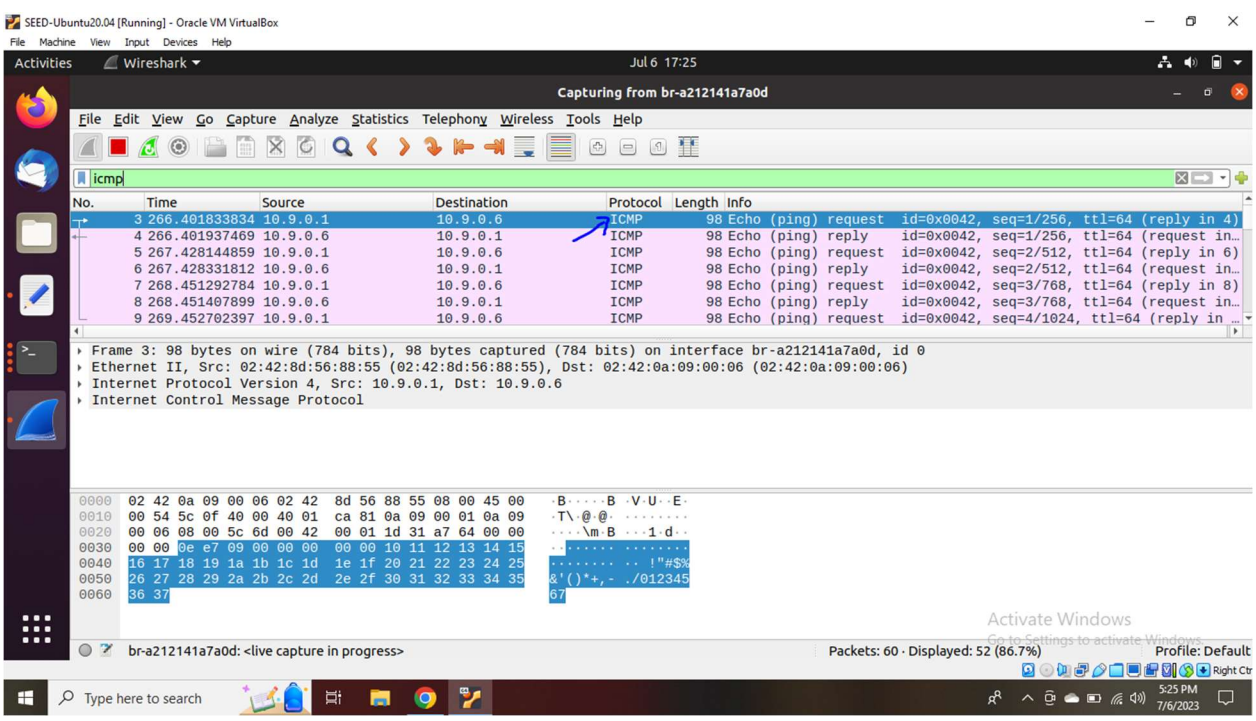
I have observed that when we try to run the python file required for Task 1.1A as non-root user it prints out that **"operation is not permitted"**. Here we can get into the execution but unable to sniff as non-root user.



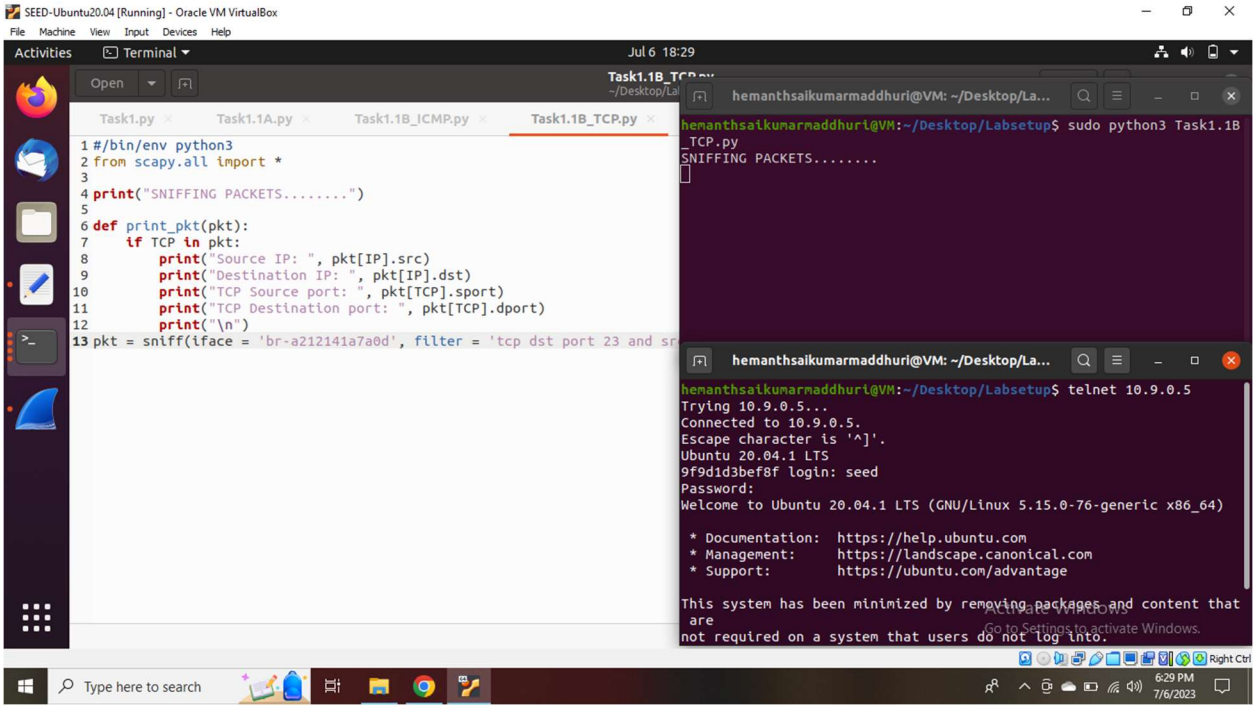
In the screenshot below, we are running Task1.1B_ICMP.py to capture only the ICMP packets. I am running the code in one of the terminals while creating traffic on the other one.



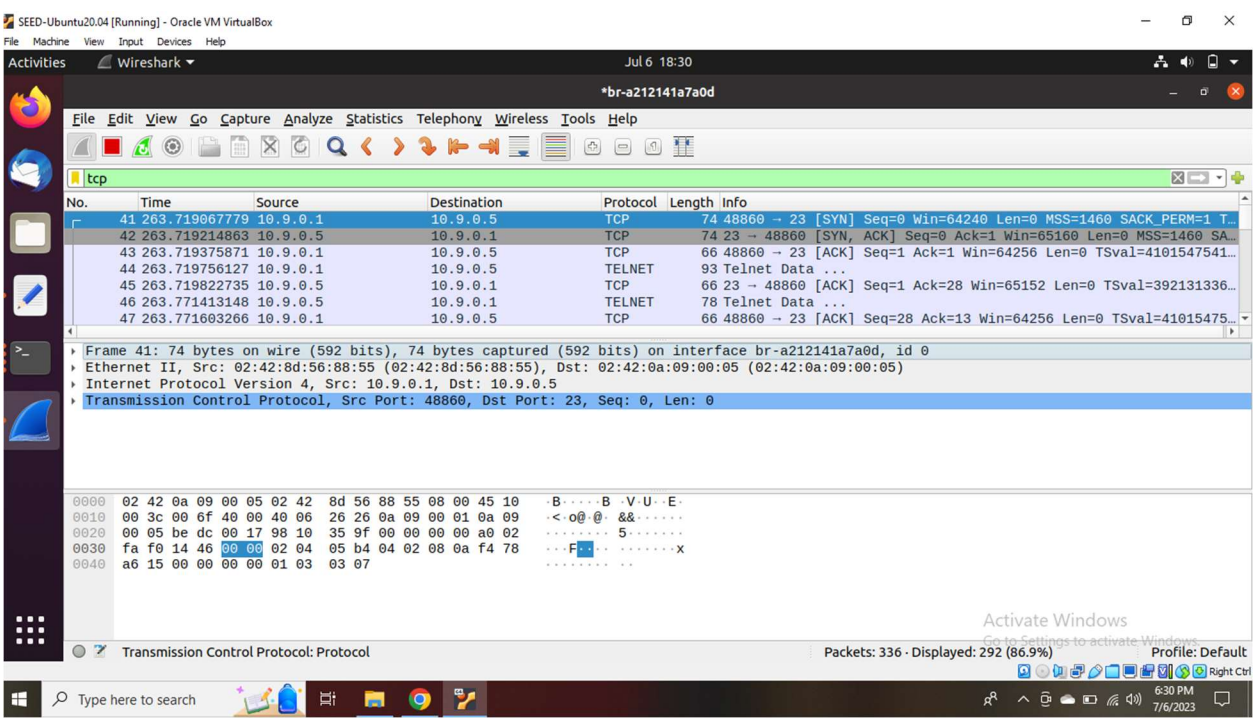
Screenshot from the Wireshark is attached here to show that we have captured ICMP packet successfully.



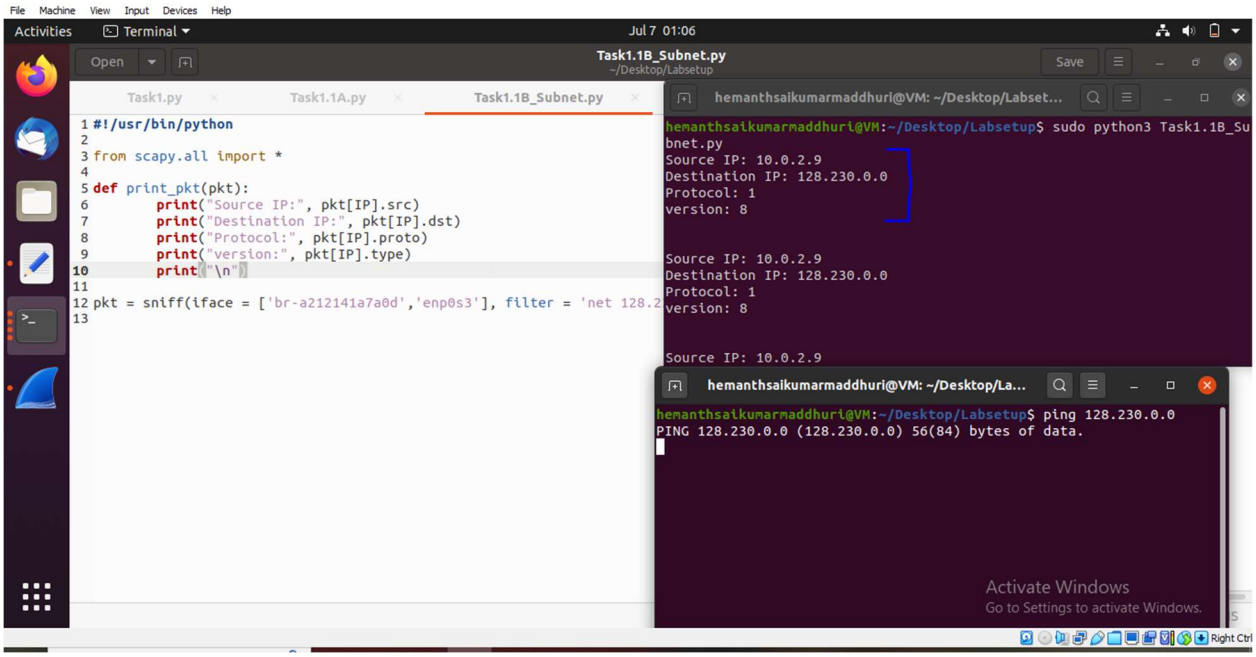
In the below screenshot, we are running the Task1.1B_TCP.py python file in one of the terminals and other terminal we are creating traffic using command **“telnet 10.9.0.5”**.



The same is captured using the Wireshark application as shown below.

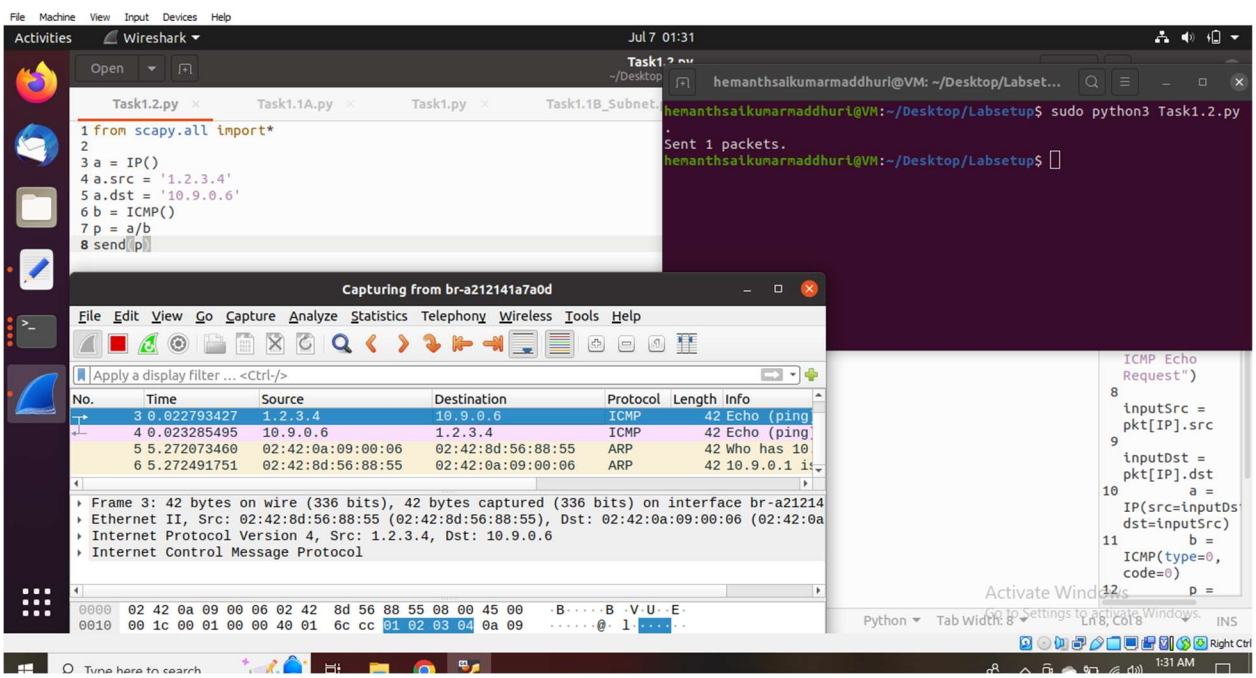


In the below screenshot, we are running the Task1.1B_Subnet.py python file in one of the terminals and other terminals we are creating traffic using command **“ping 128.230.0.0”**.



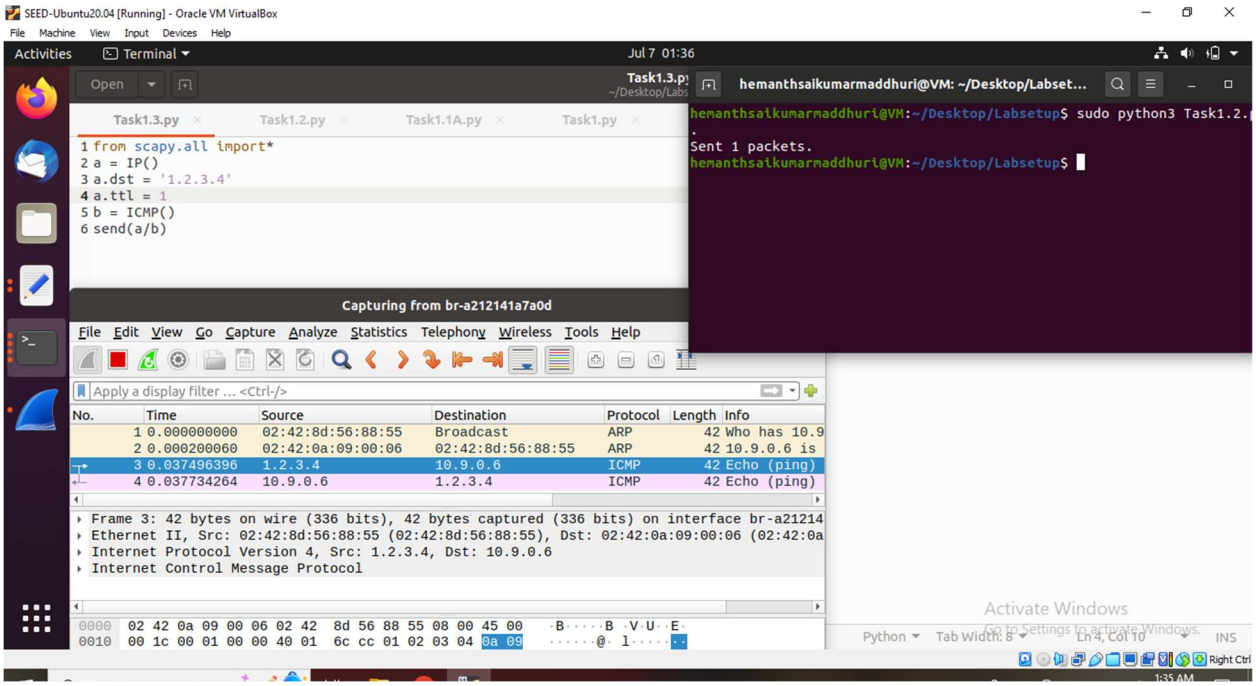
3.2 Task 1.2: Spoofing ICMP Packets

Here we are trying to spoof an ICMP echo request packet with an arbitrary source IP address. As shown in the screenshot below I have run the python file and we can see that output as **“sent 1 packet”** similarly in the Wireshark application too.

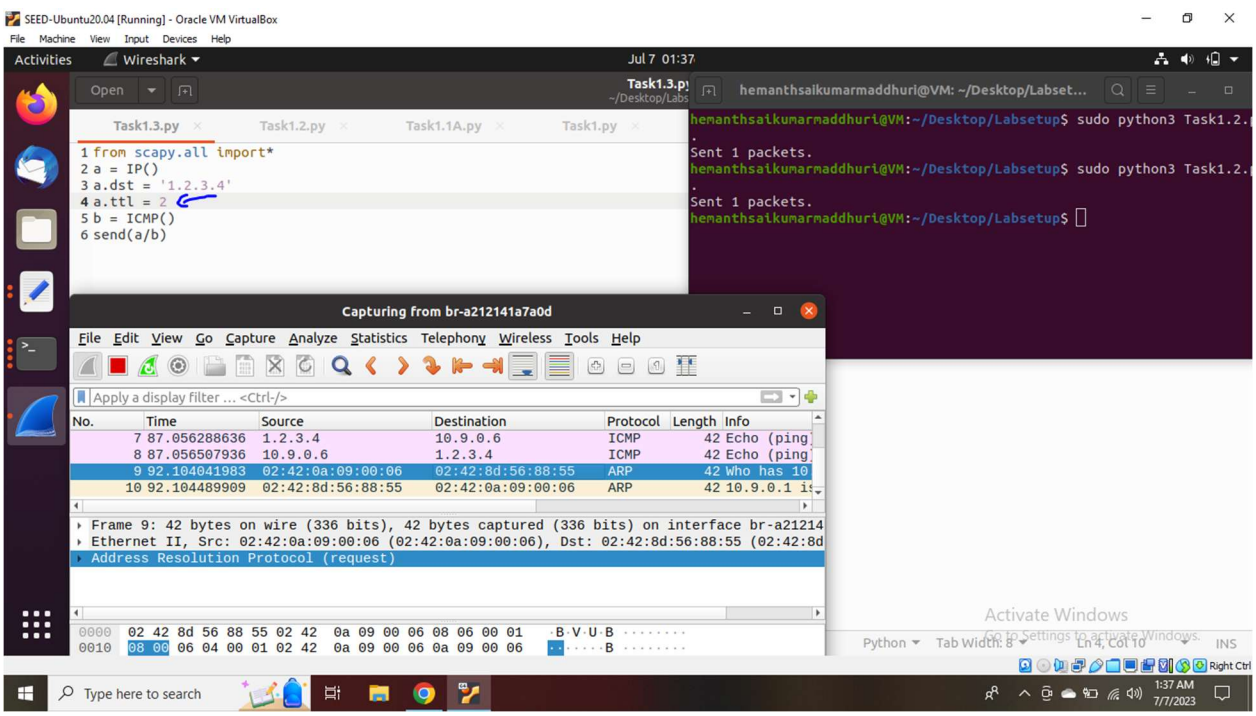


3.3 Task 1.3: Traceroute

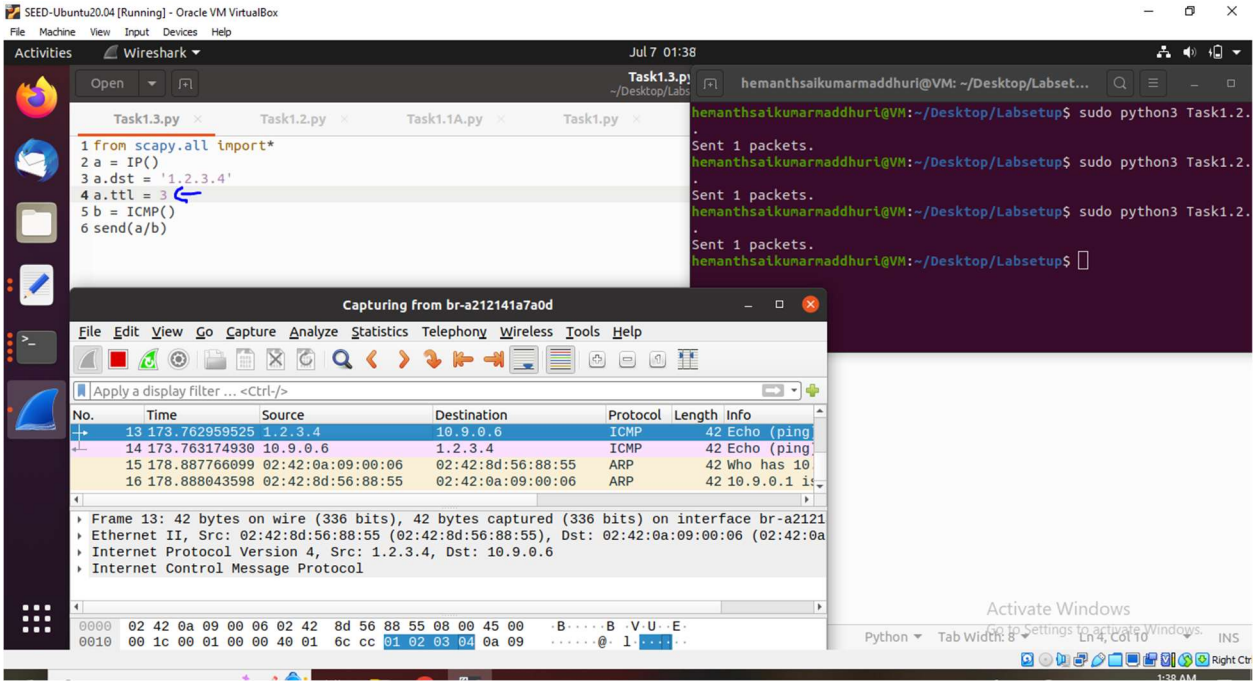
Here we are trying to estimate the distance between the routers. So I have run the below code with TTL value set to **"1"** and in this instance the packet is being delivered.



As instructed, I have next set the TTL value to **"2"** whereas even in this case the packet was delivered and the same can be found in Wireshark application too.

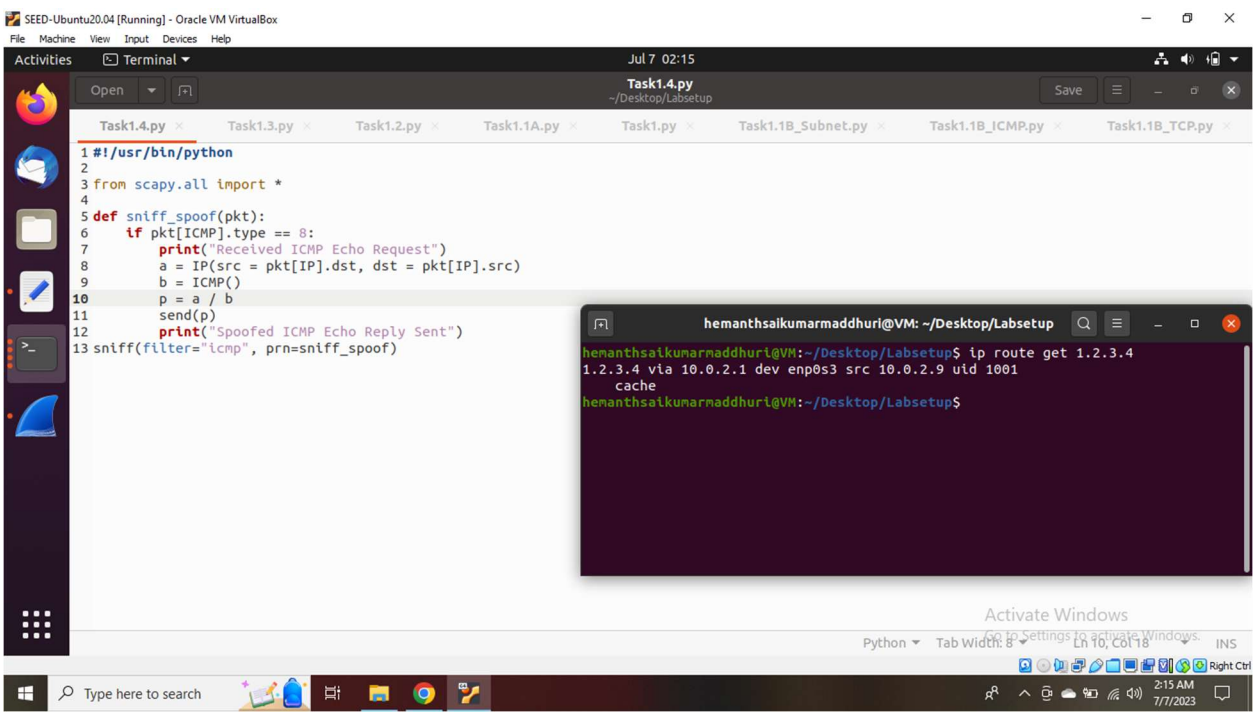


As we did in the previous steps, I have set the TTL value to **“3”** here even in this case the packet was delivered successfully and I hope the distance between routers in my case can be 1, 2 or 3.

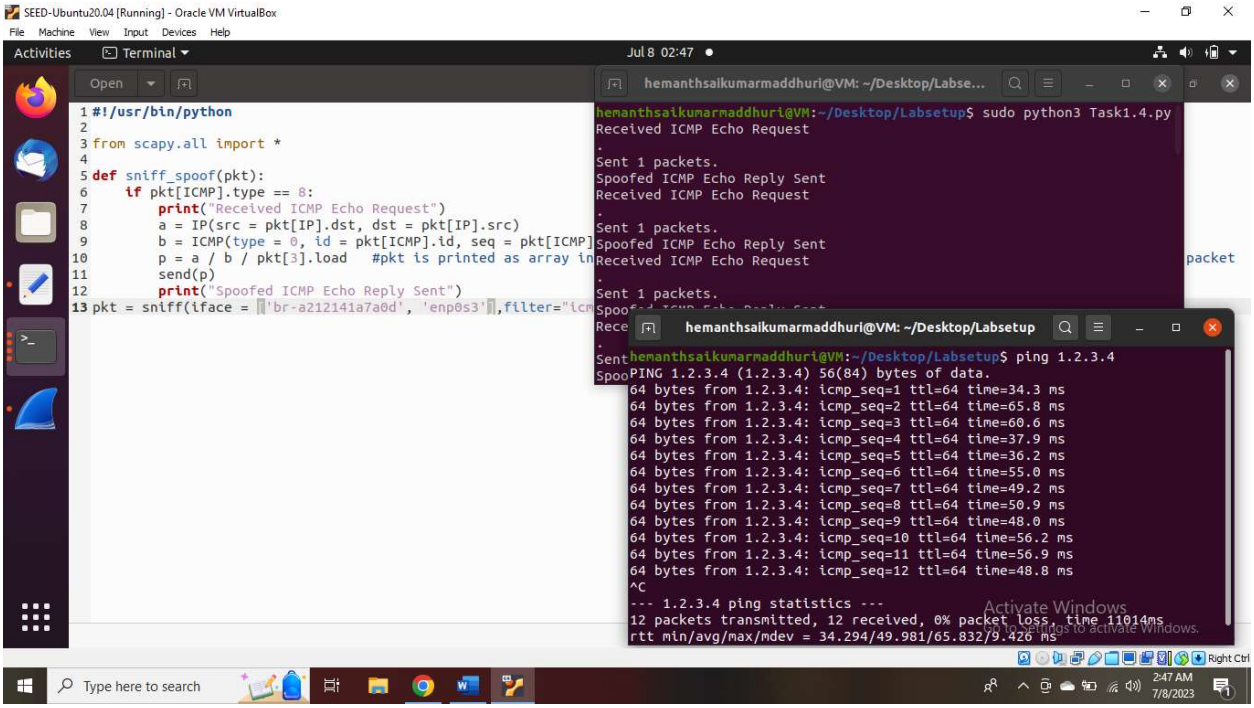


3.4 Task 1.4 Sniffing and then Spoofing

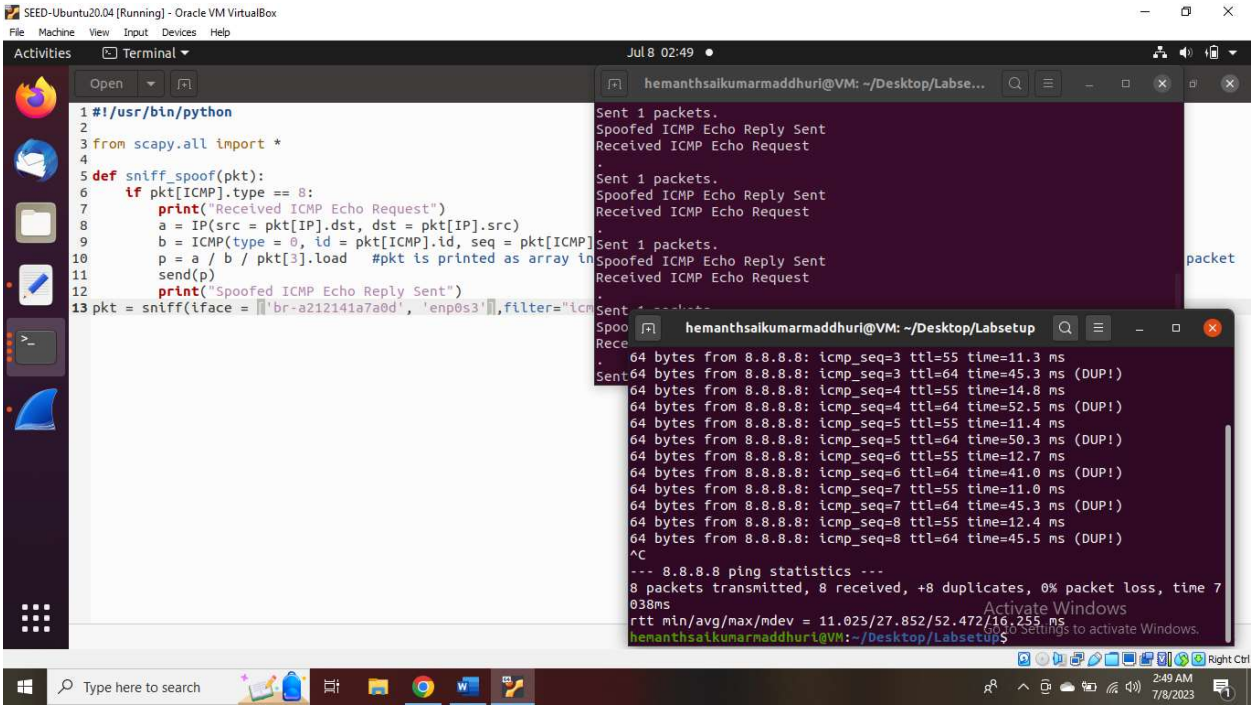
As per given hint I have tried the command **“ip route get 1.2.3.4”** and understood its characteristics.



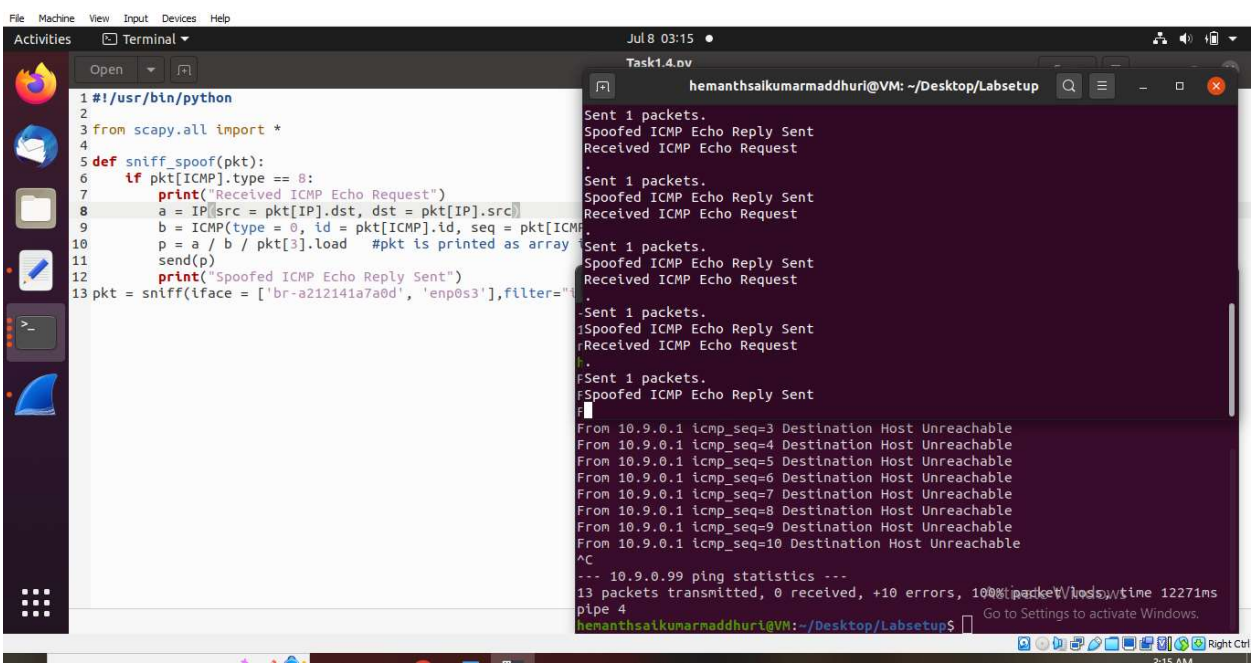
Here in this task, we are going to perform both sniffing and spoofing the same which we performed individually earlier. As shown in the screenshot below I have run the python file in one of the terminals and at the same time trying to ping the ip 1.2.3.4 using command **"ping 1.2.3.4"**. As we run the python file, we can see that the packet is sent no matter what the response is received or not and the packet is spoofed in successfully as we can see that number of packets sent and received are same.



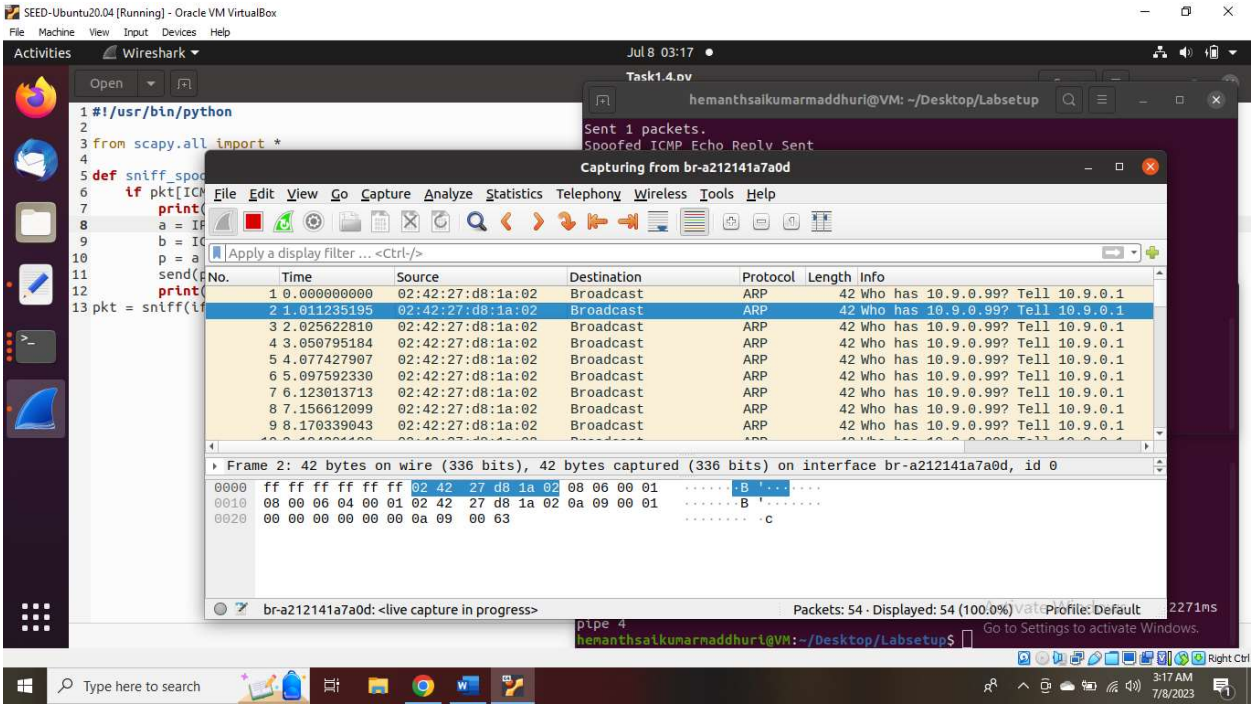
While the above process is still going on, in one of the terminals we run the command **"ping 8.8.8.8"** as instructed in the lab manual and we can clearly see that the packet transmission is successful, and we have successfully sniffed and spoofed the packet.



As instructed, we are trying to ping 10.9.0.99 using command **"ping 10.9.0.99"** from one of the terminals and I have observed that when to try to ping 10.9.0.99 it is unreachable as it does not exist.



The Wireshark Screenshot for the same is attached below.



Summary

In this lab I have learnt how to sniff and spoof data to host or target system. I have learnt about a new library of python named Scapy with is useful for network security. Also, I have observed various things like the packet that we print to command line is printed as array, I have tested with giving names and numbers, yet I got the same result while debugging. Overall it was a good lab and understanding the topics like spoofing and sniffing is very important according to my opinion.