**Packet Sniffing and Spoofing**

Chawin Metah

Department of Computer Science, Purdue University Fort Wayne

ACS 545000: Cryptography and Network Security

Lab 3

Dr. Zesheng Chen

February 2, 2022

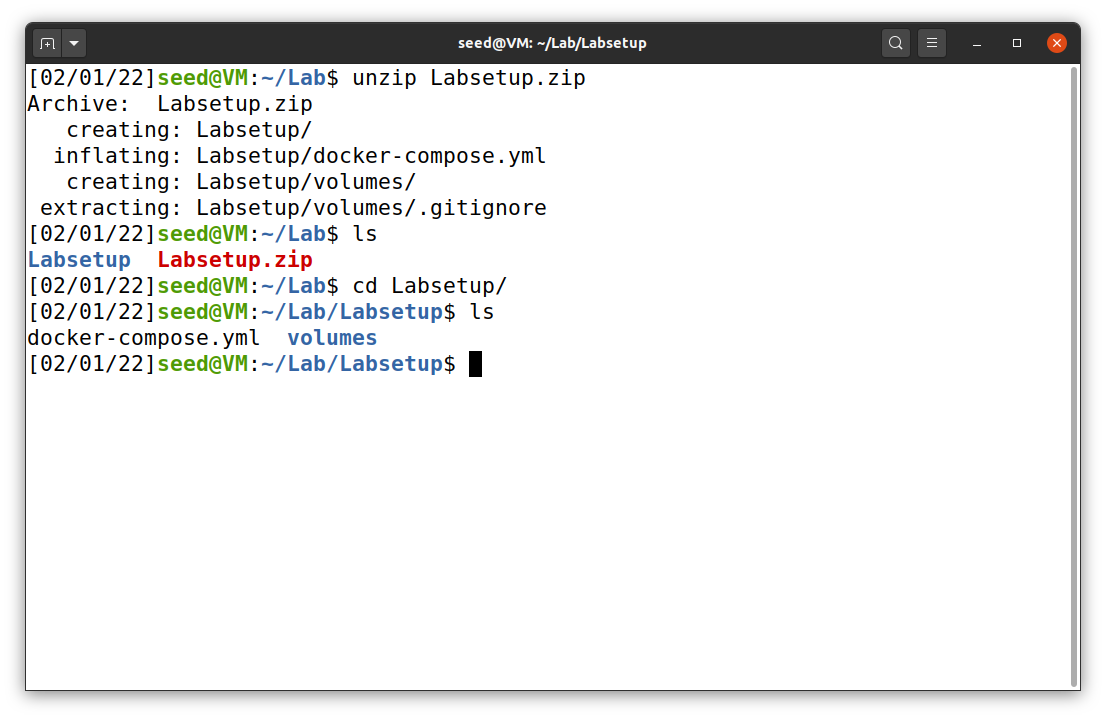
**Task 1.1: Sniffing Packets**

In this lab, we have to download Labsetup.zip from https://seedsecuritylabs.org/‌Labs\_20.04/Networking/Sniffing\_Spoofing/. After using unzip command, we will see two extracted files consist of docker-compose.yml and volume directory, as shown in Figure 1. To check if we got the right file, we have to use the command below (Figure 2). The file indicates that there are three users: attacker, host A, and host B.

cat docker-compose.yml

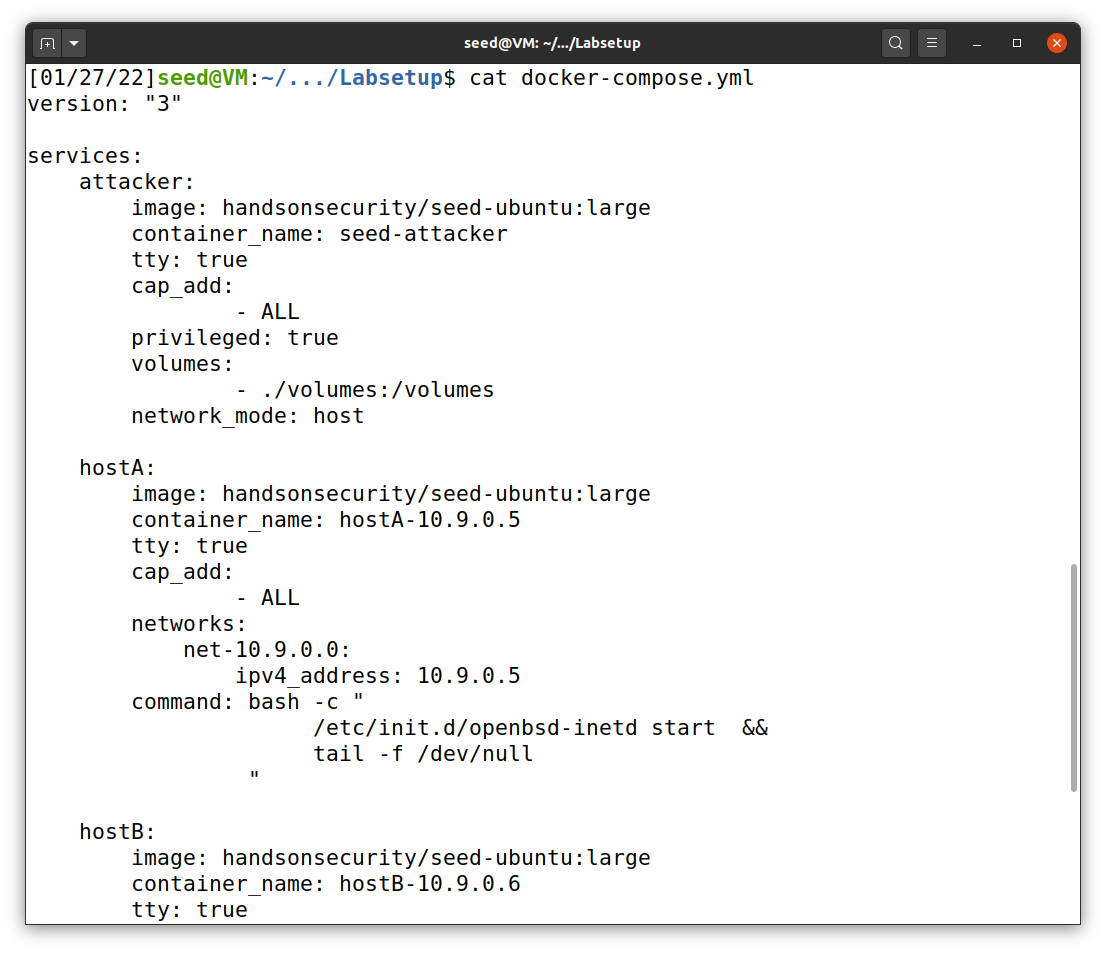
**Figure 1**

*Labsetup.zip*

**

**Figure 2**

*Docker-compose.yml file*



hostB

IP: 10.9.0.6

hostA

IP: 10.9.0.5

Attacker

IP: 10.9.0.1

***Task 1.1A***

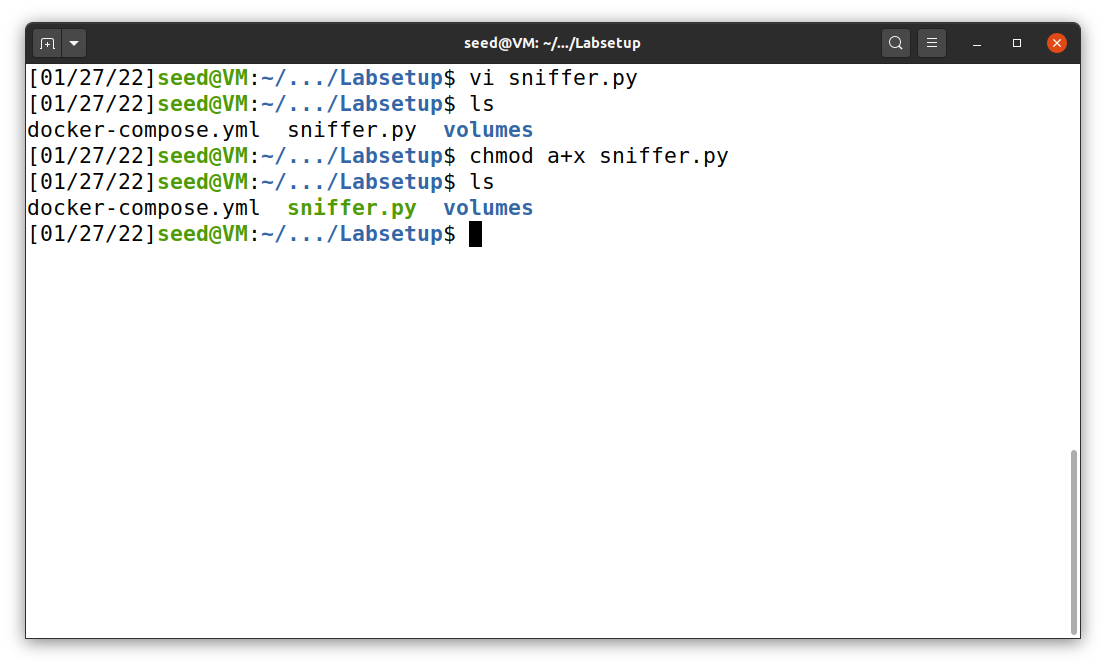
To begin, we first have to create a python file that use scapy with the command below and make it an executable file (Figure 3). The python code of sniffer.py is shown in Figure 4.

vi sniffer.py

chmod a+x sniffer.py

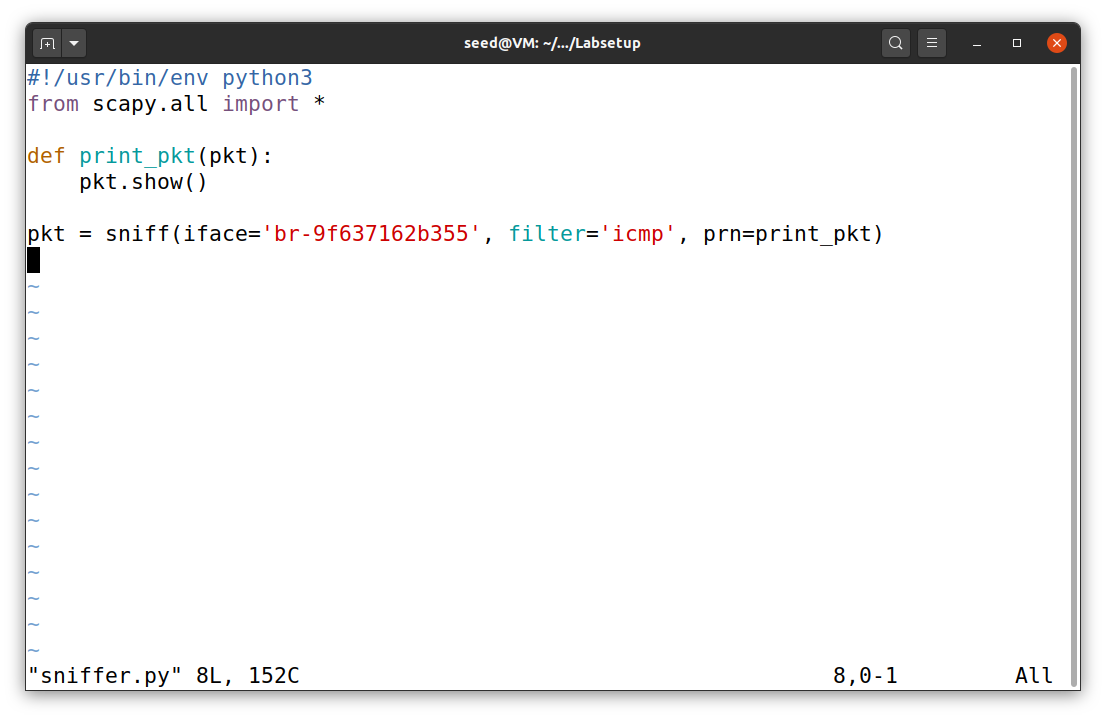
**Figure 3**

*Creation of sniffer.py*



**Figure 4**

*sniffer.py*



This has to be match with the network ID

Before we can run this program, we have to make sure that the interface (network) ID is match. To obtain for the correct ID, after we build and create a network (Figure 5), we can use either ifconfig or docker network ls command (Figure 6 and 7).

**Figure 5**

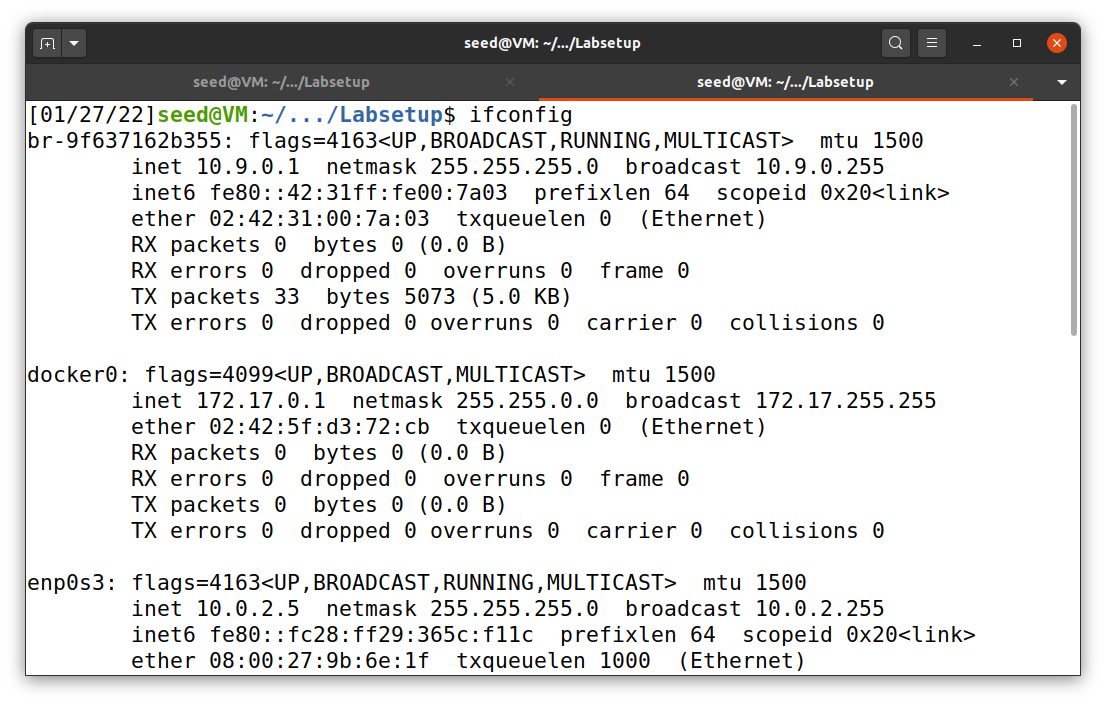
*Establish a docker network*

**Text

Description automatically generated**

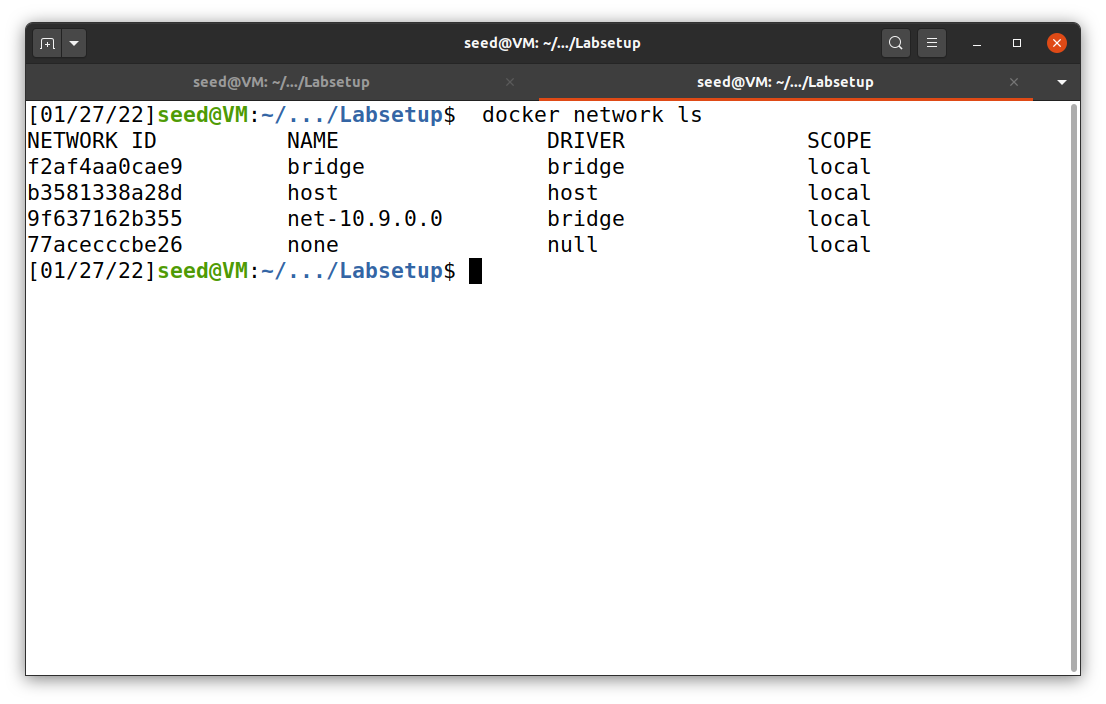
**Figure 6**

*ifconfig command*

****

**Figure 7**

*docker network ls command*

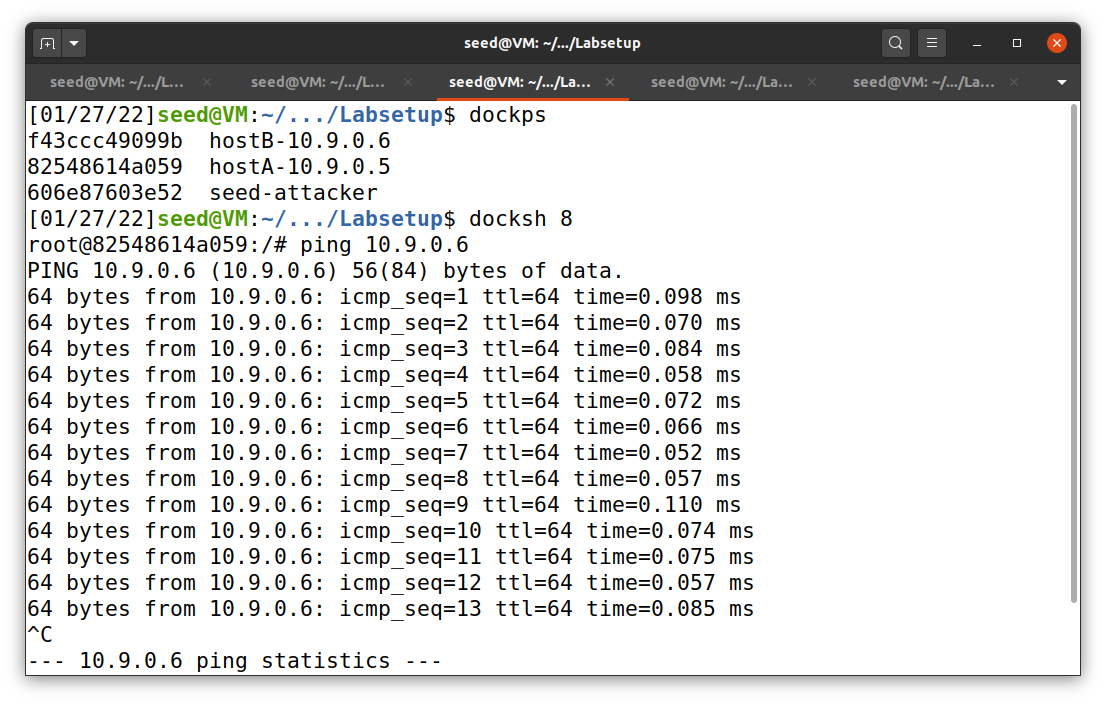
****

This is our network

After that, we can run sniffer.py and hook up to one of our servers. I chose to dock with hostA (10.9.0.5) by using docksh command, followed by hostA’s ID (Figure 8). Then, I used ping to find the latency between host A and B, as shown in Figure 8.

**Figure 8**

*Connection to host A*

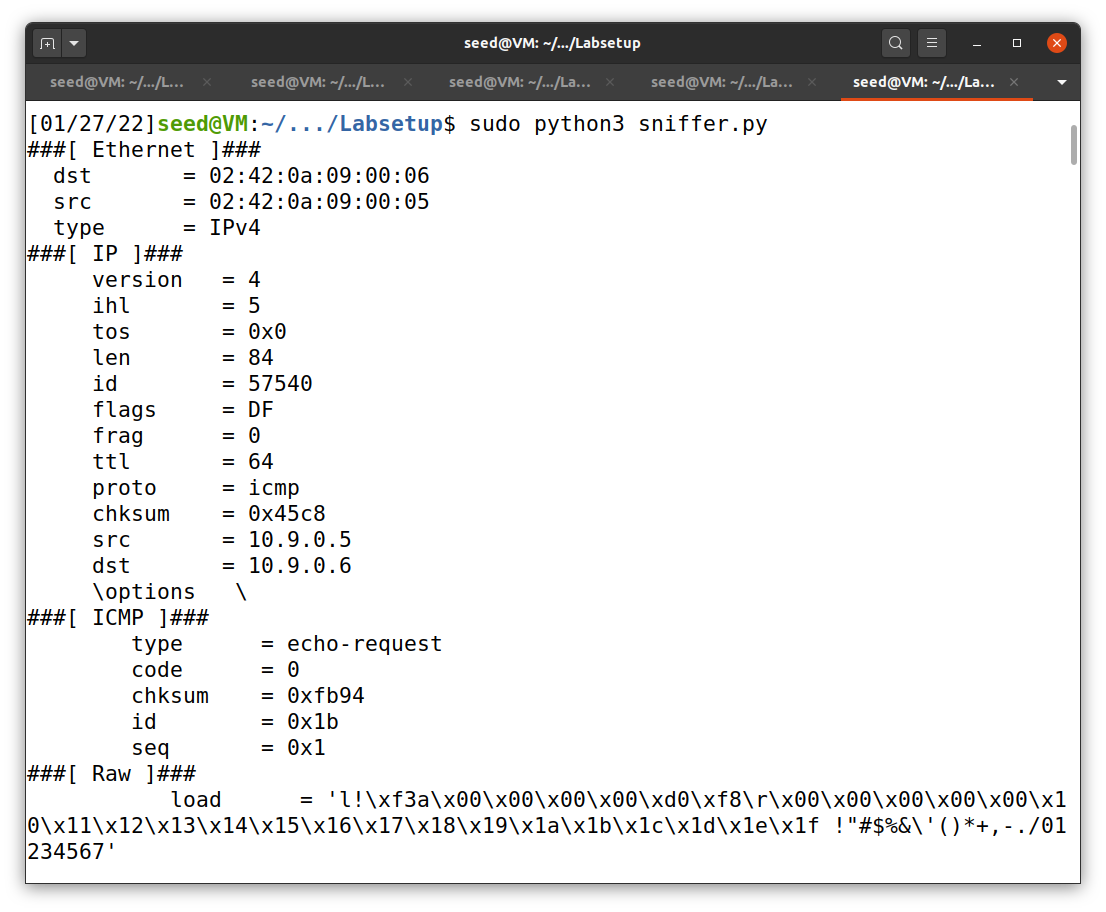
****

hostA ping hostB

After running sniffer.py with super user, the result can be seen in Figure 9.

**Figure 9**

*Sniffing result*

**

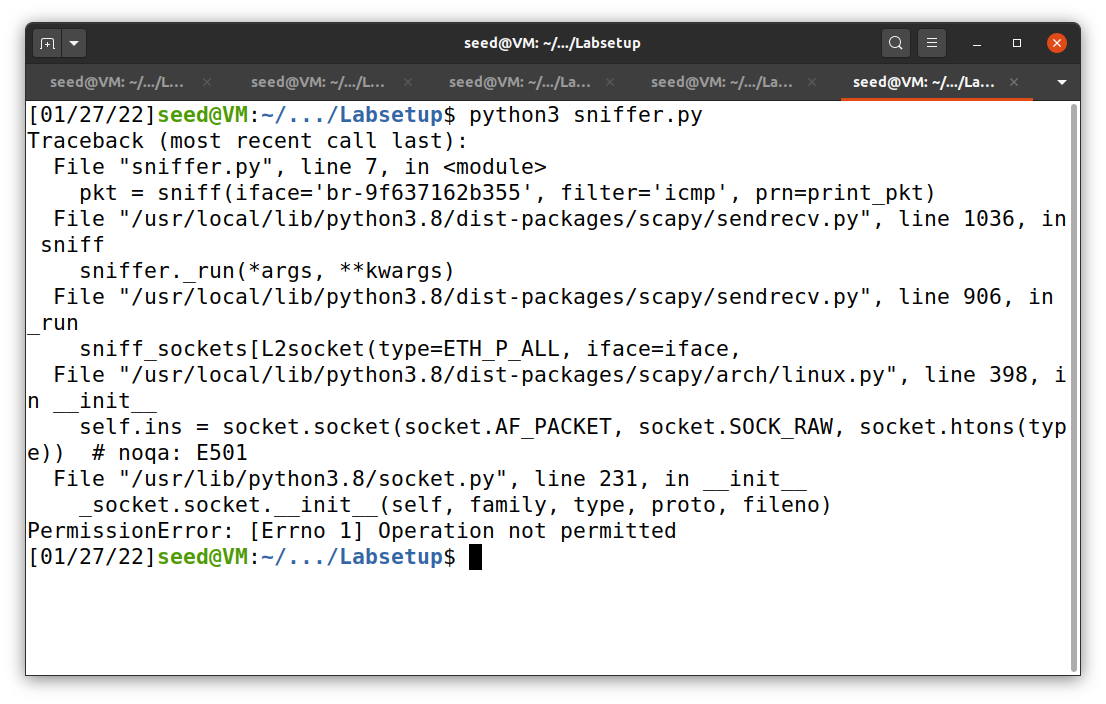
Terminal that runs sniffer.py

When we want to run a python program with scapy, it requires a super user (root privilege) access to execute. The example command is shown below. If we do not use sudo, the execution will be denied with an error message show up (Figure 10).

sudo python3 sniffer.py

**Figure 10**

*sniffer.py without sudo*

**

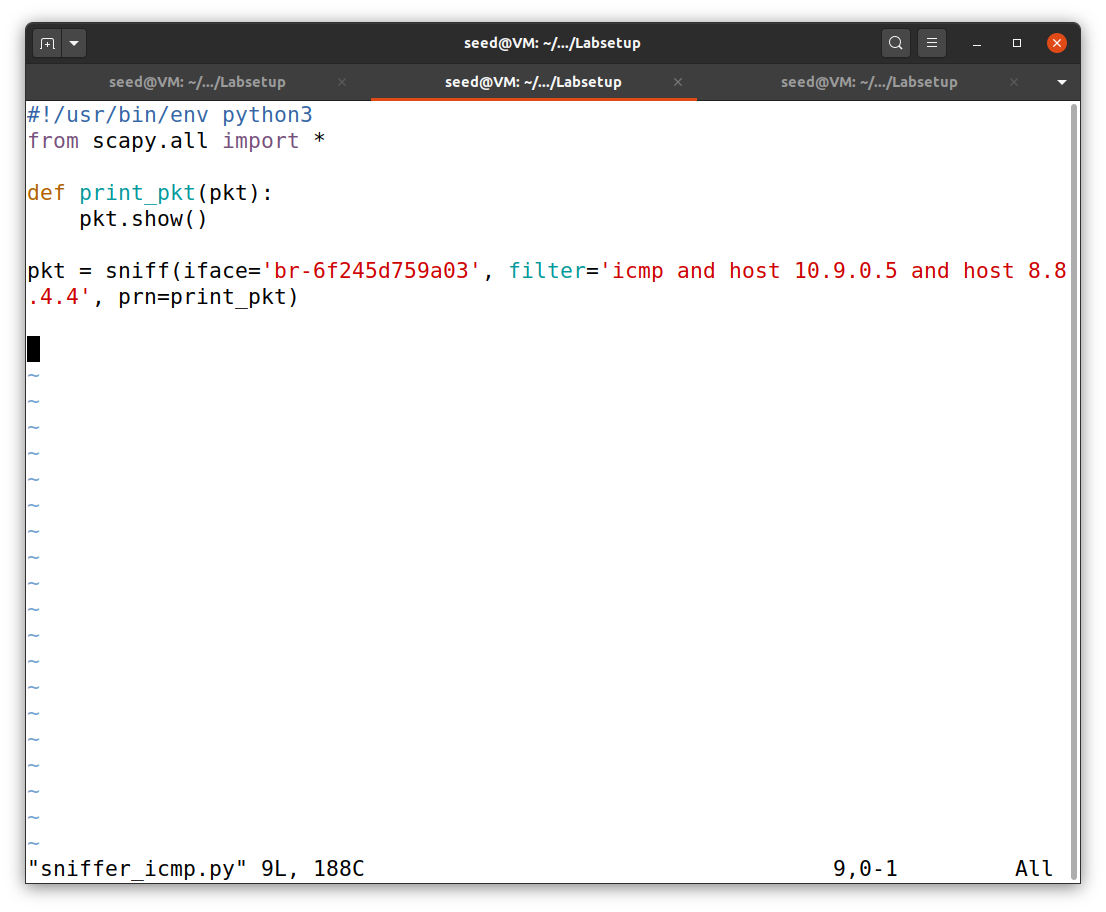
***Task 1.1B***

The first task of task 1.1 B is to capture only ICMP packets. However, we already did this in task 1.1 A. So, I configured it to sniff out only from one specific host (host A) to another (google.com with IP 8.8.4.4) with the code shown in Figure 11.

Note: I did this task after I reset the machine. Network (bridge) ID may be different from previous task.

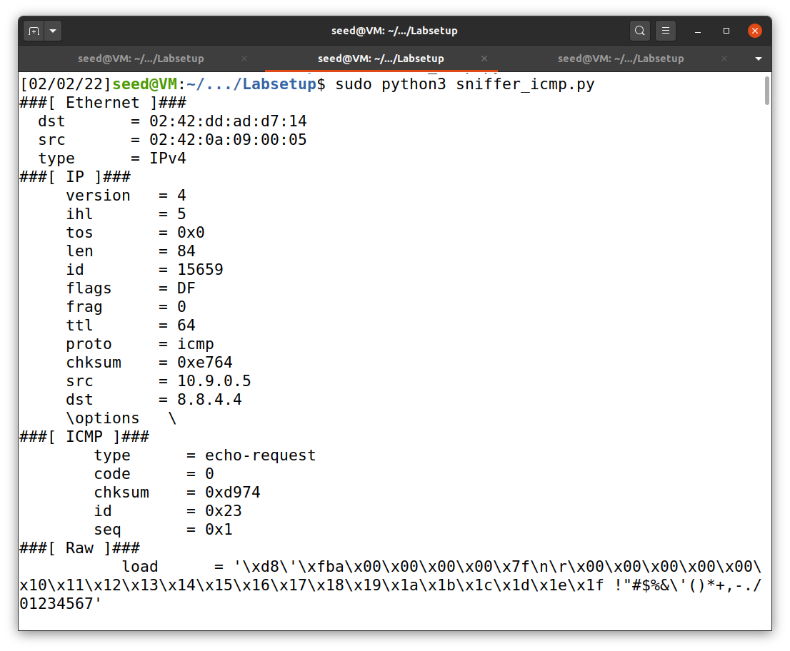
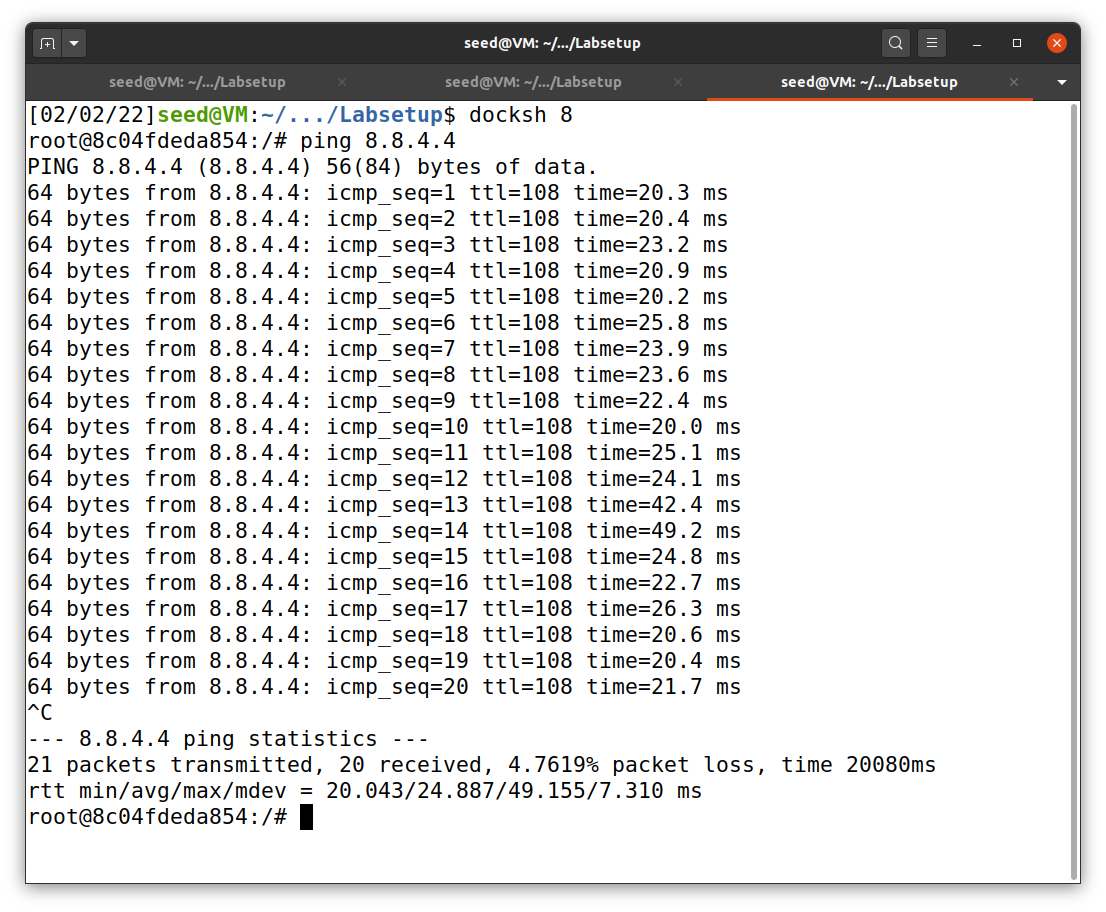
**Figure 11**

*sniffer\_icmp.py*

**

**Figure 11**

*The result from capturing only icmp packets*

**

hostA ping google.com

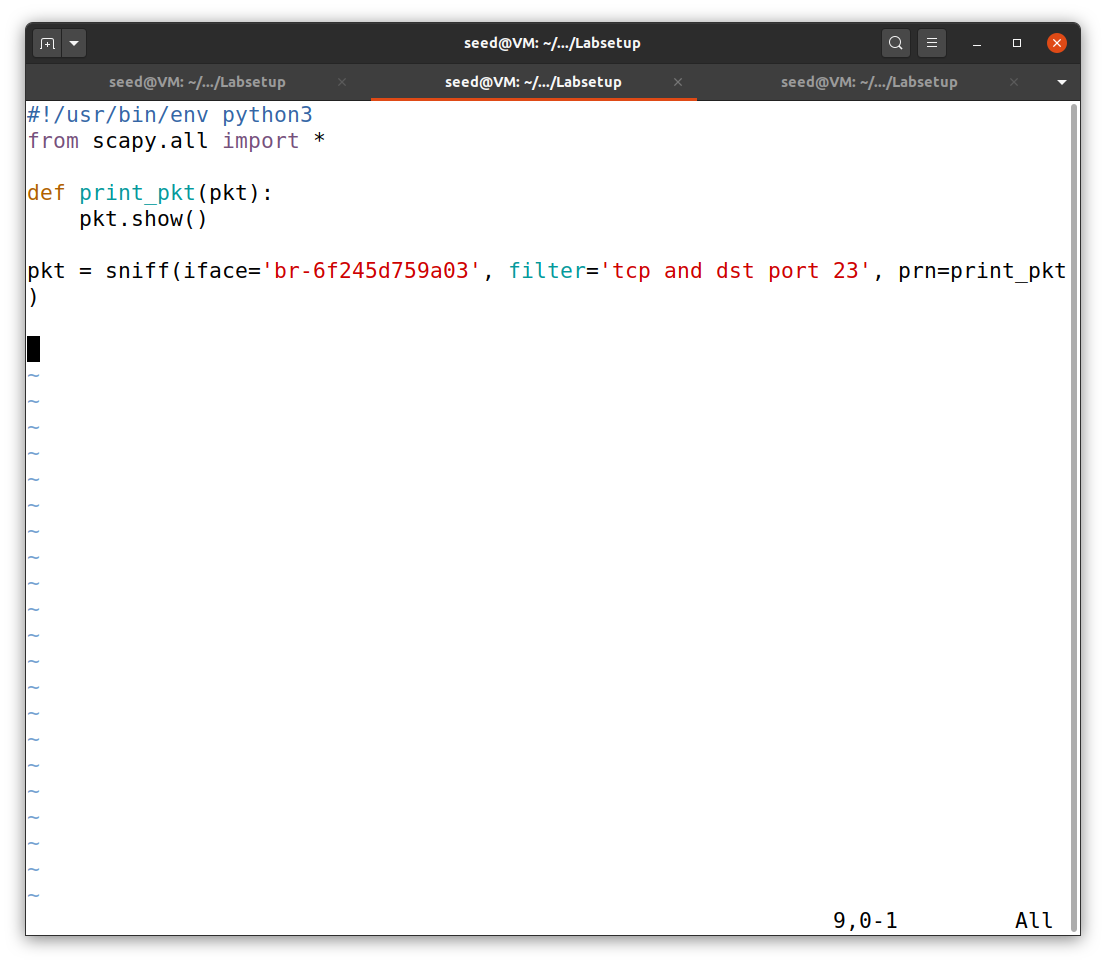
hostA

Terminal that runs sniffer\_icmp.py

Then, we will capture any TCP packet that comes from a particular IP and with a destination port number 23. The code from Figure 12 was used. The result when host A telnet host B (communicate via port 23) is shown in Figure 13.

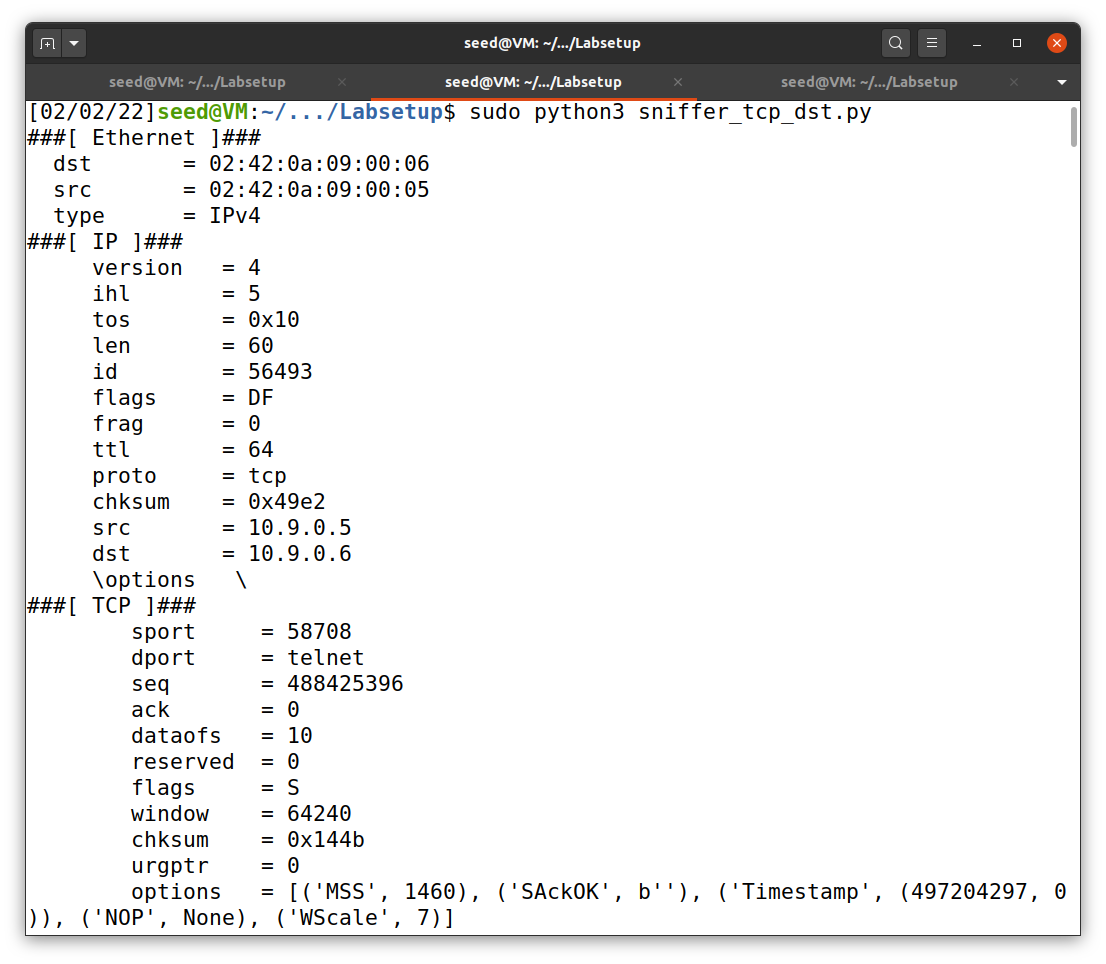
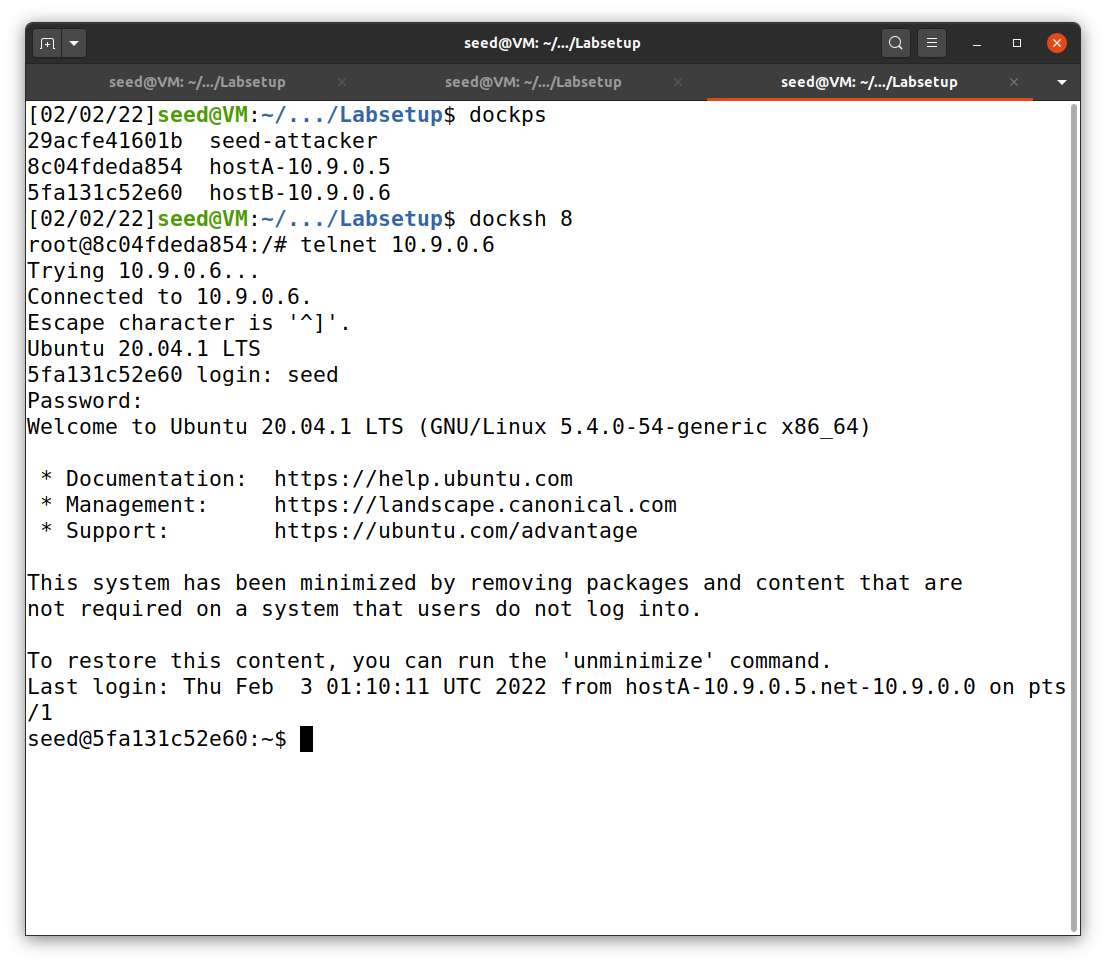
**Figure 12**

*sniffer\_tcp\_dst.py*

**

**Figure 13**

*The result from capturing tcp packets on port 23*

**

hostA

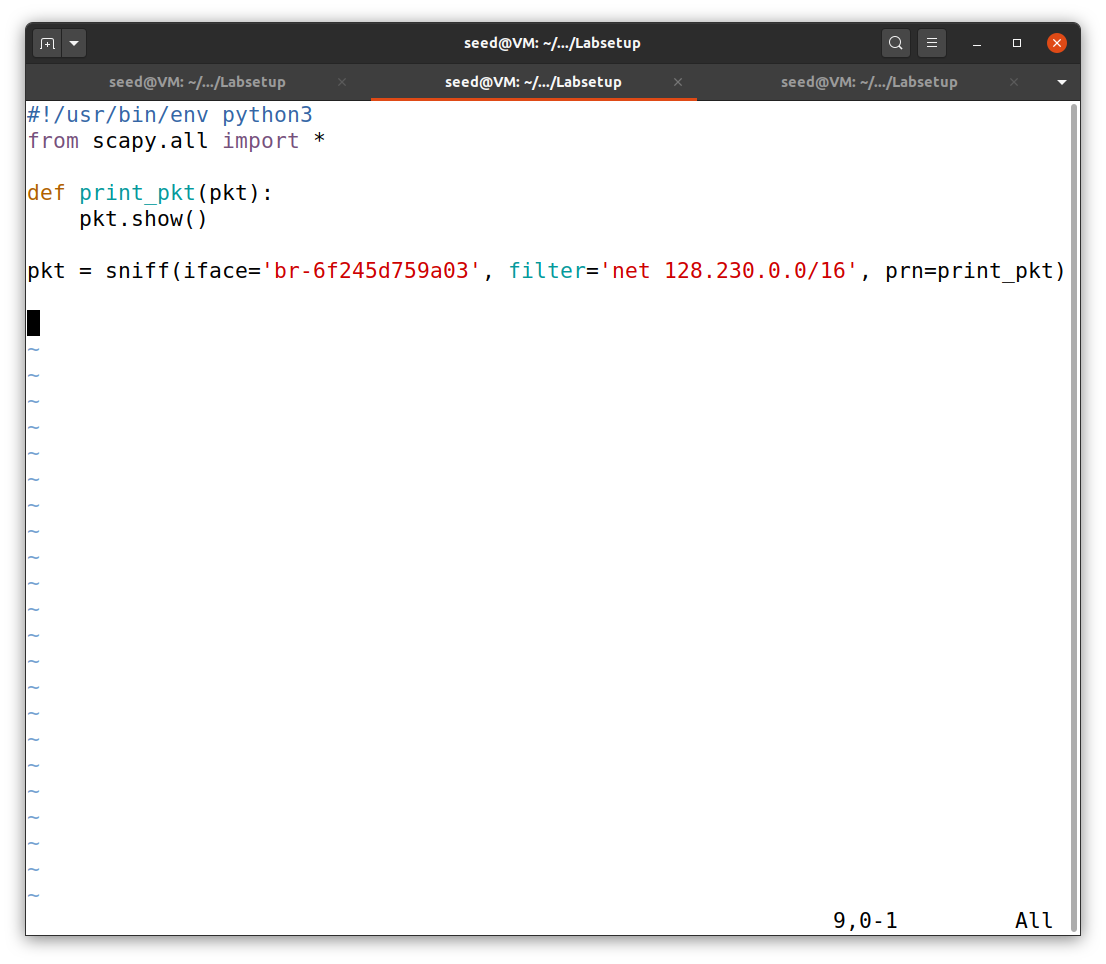
hostA telnet hostB

Terminal that runs sniffer\_tcp\_dst.py

Finally, we will capture packets comes from or to go to a particular subnet. The code and results are shown in Figure 14 and 15, respectively.

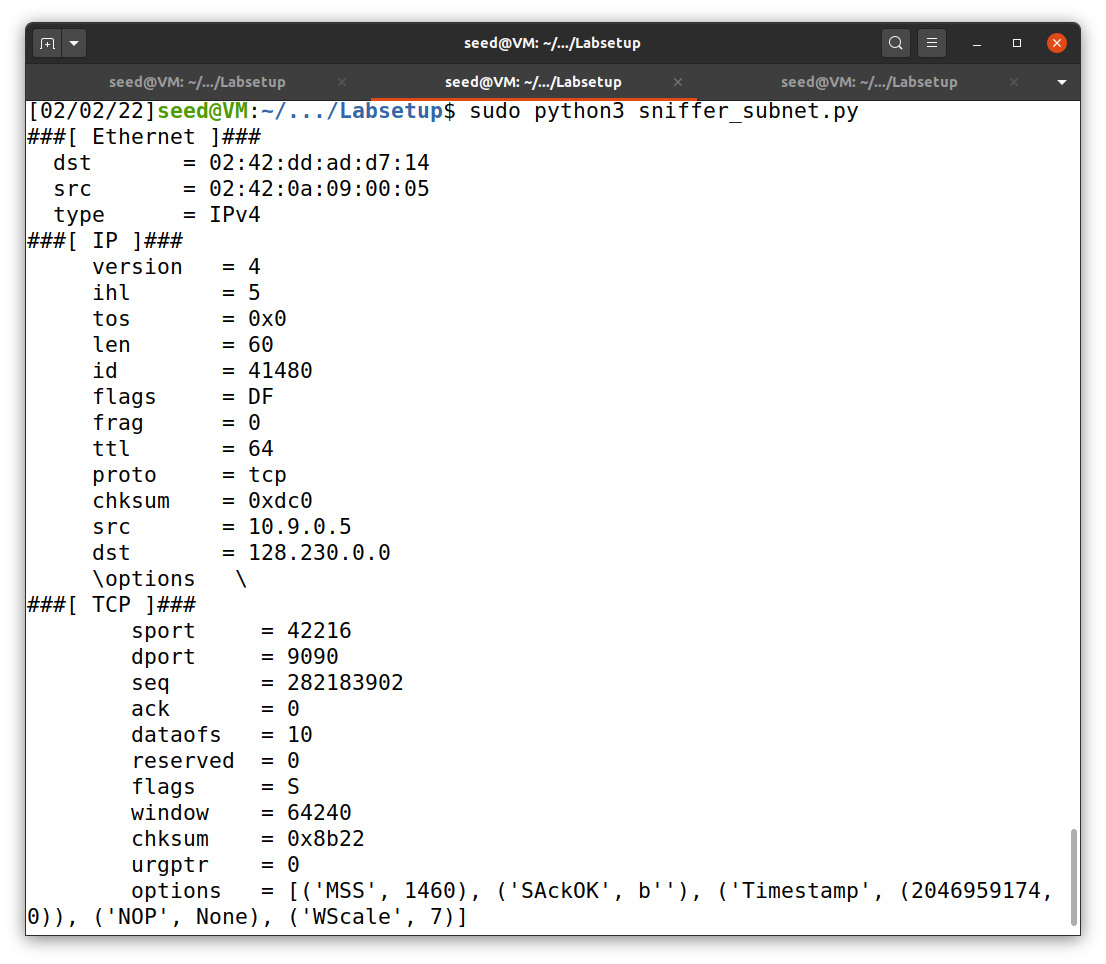
**Figure 14**

*sniffer\_subnet.py*



**Figure 15**

*The result of capturing packets from particular subnet*



hostA

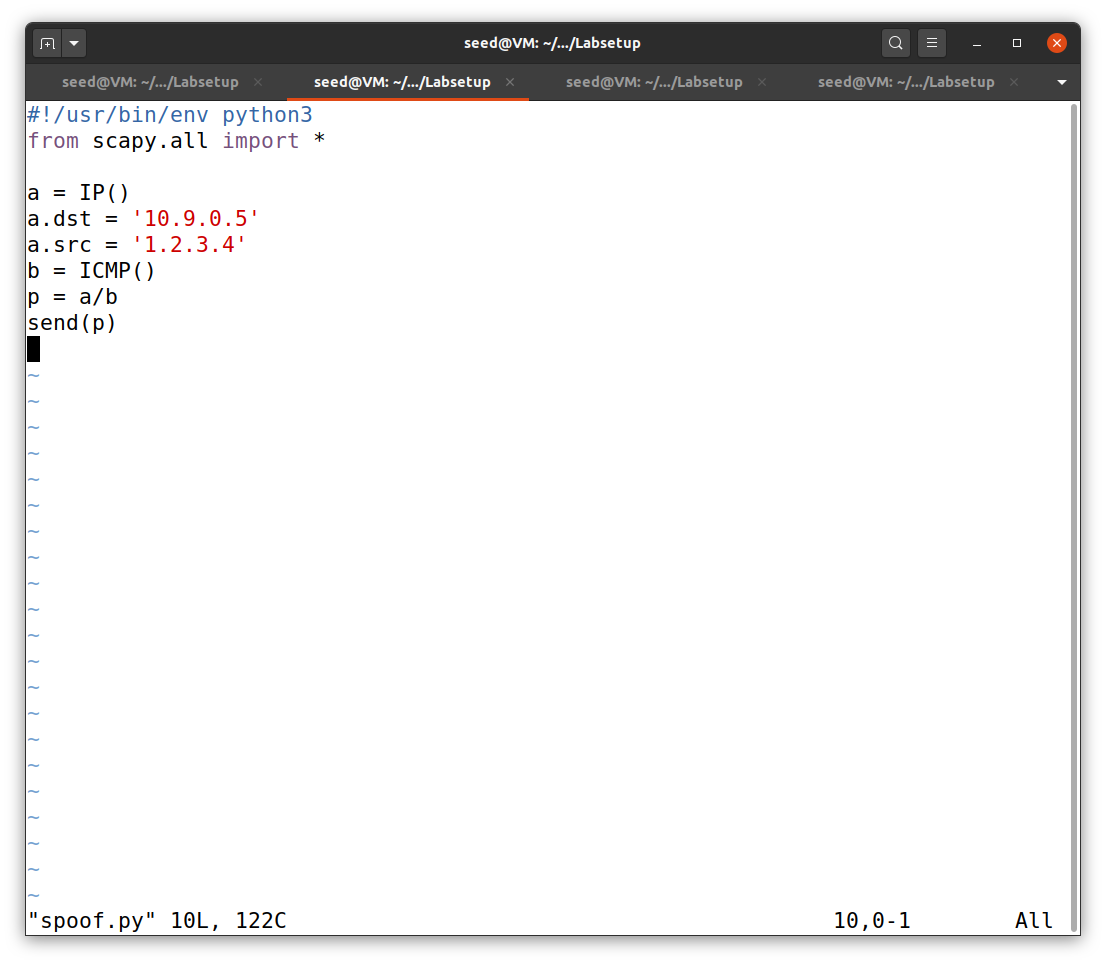
Terminal that runs sniffer\_subnet.py

**Task 1.2: Spoofing ICMP Packets**

To spoof ICMP packets, I first built spoof.py, as shown in Figure 16.

**Figure 16**

*spoof.py*



After that, I run the command below on one terminal. The other terminal was used to run spoof.py file. These are shown in Figure 17. This spoof packet can also be observe via wireshark, as shown in Figure 18.

sudo tcpdump -w /tmp/packets -v icmp

**Figure 17**

*spoof.py*



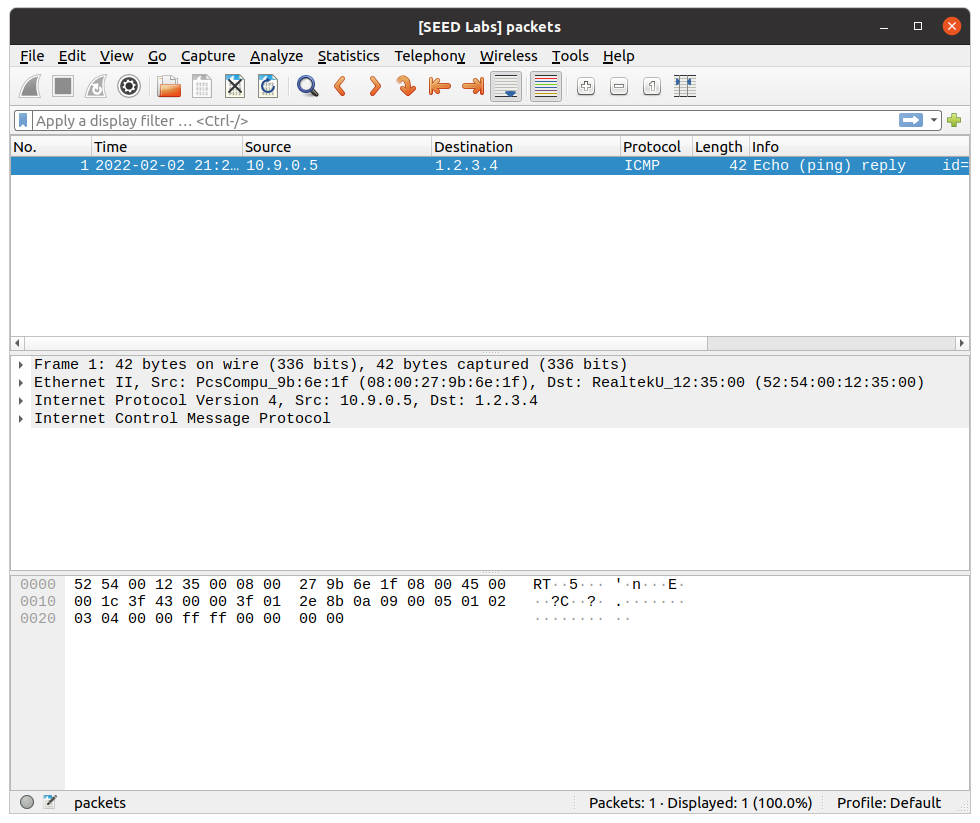
Terminal 2

Terminal 1

Initially shows 0, but will change to 1 after the file is being sent

**Figure 18**

*Wireshark on spoof.py*



This can be achieved using the following command.

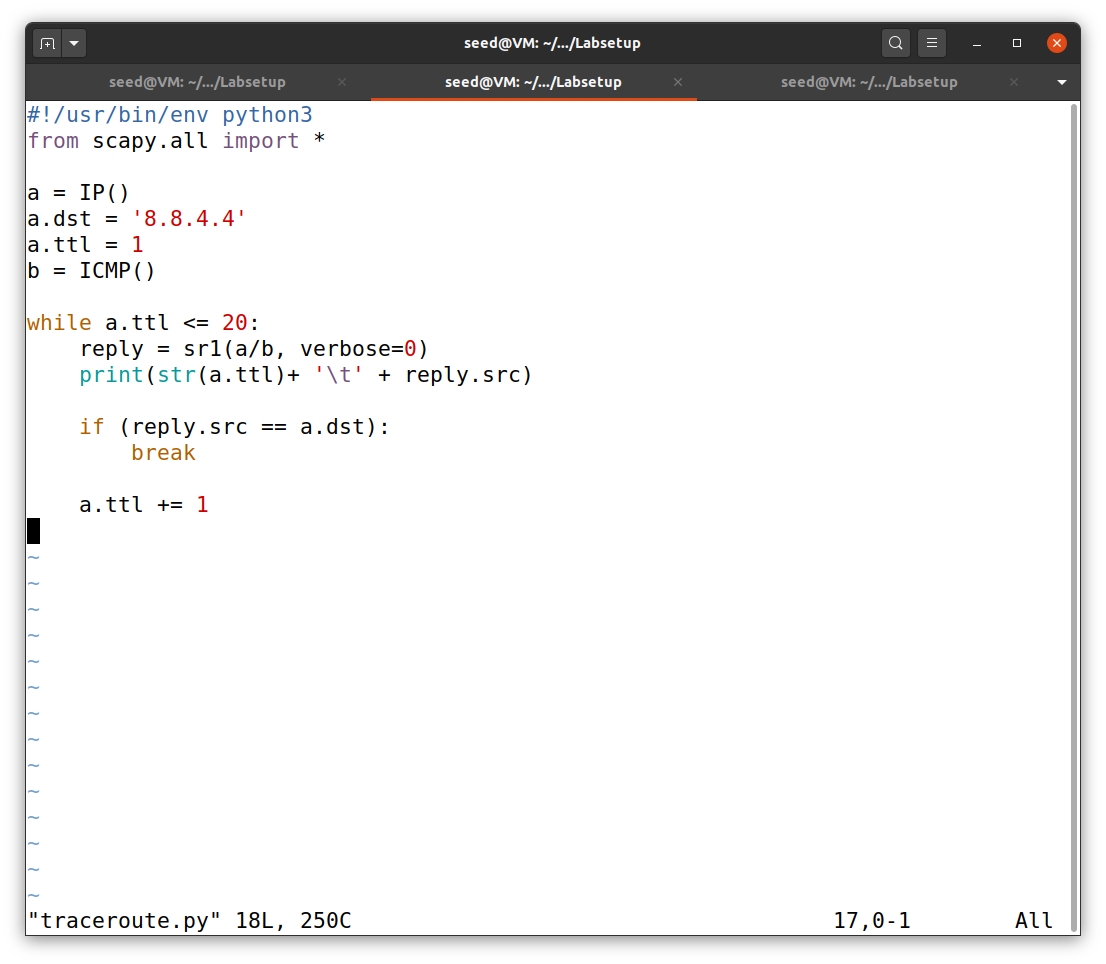
wireshark /tmp/packets

**Task 1.3: Traceroute**

The purposes of this task are to find routers’ IP and how many hops it take to reach the destination IP. I choose to traceroute to google.com (8.8.4.4). The code of traceroute.py is shown in Figure 19 and the result is shown in Figure 20.

**Figure 19**

*traceroute.py*



**Figure 20**

*The result from traceroute.py*

**Text

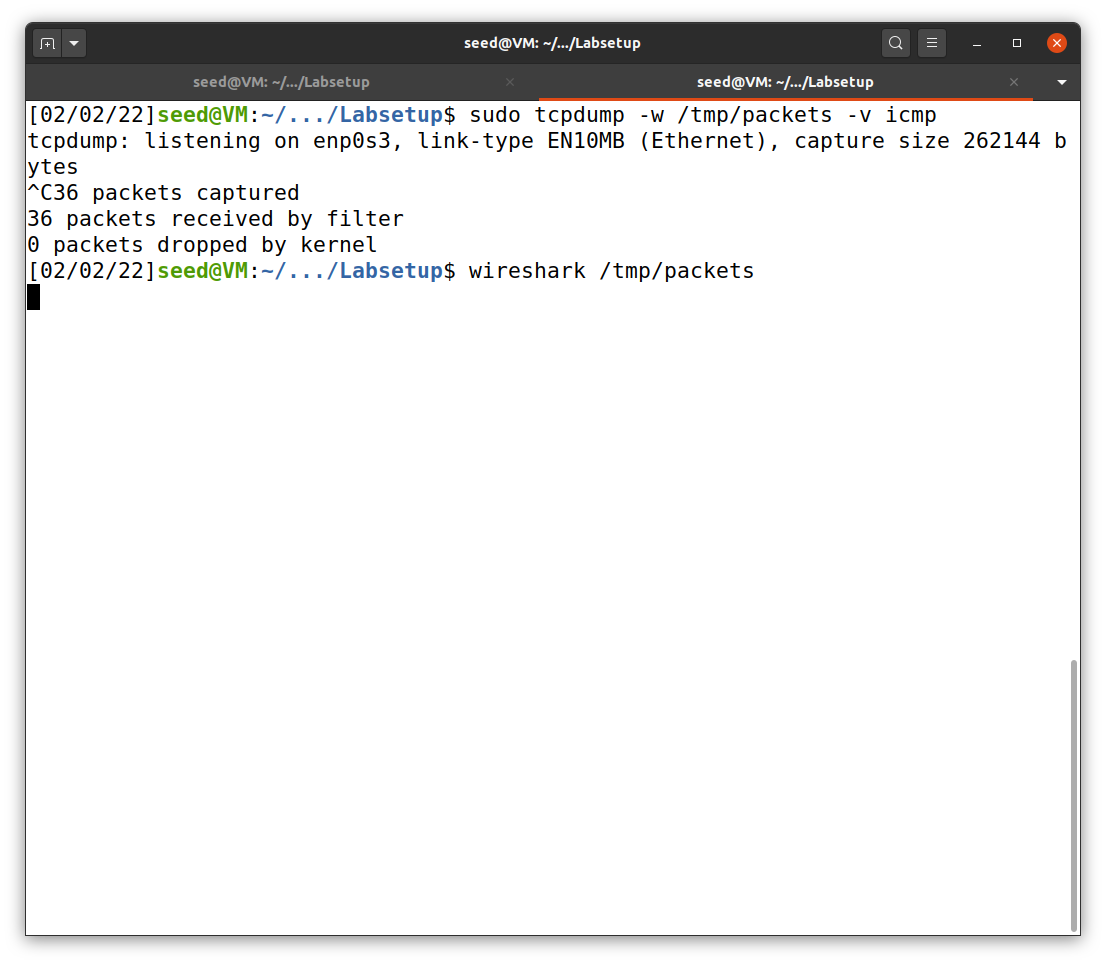
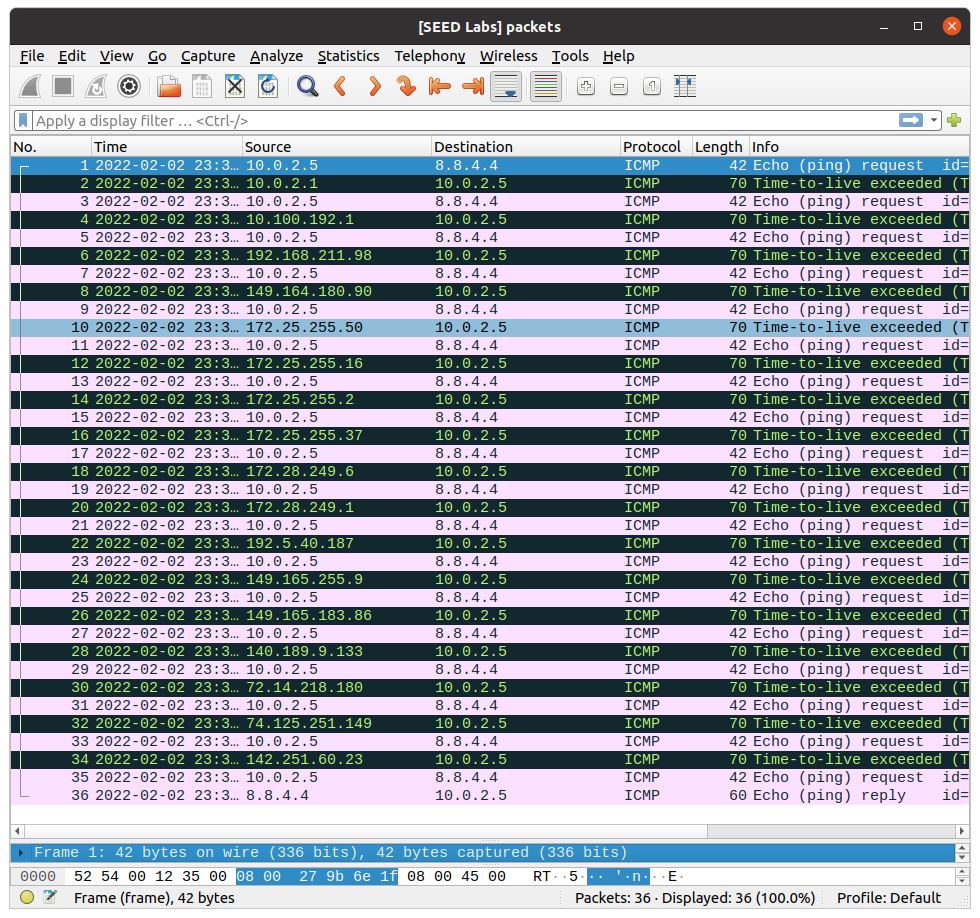
Description automatically generated**

We can also use Wireshark to track these packets by using the command from task 1.2 (Figure 21). I got 36 captured packets.

Note: The IP in Figure 21 may be different from Figure 20 because they are not the same run.

**Figure 21**

*The result from traceroute.py in Wireshark*

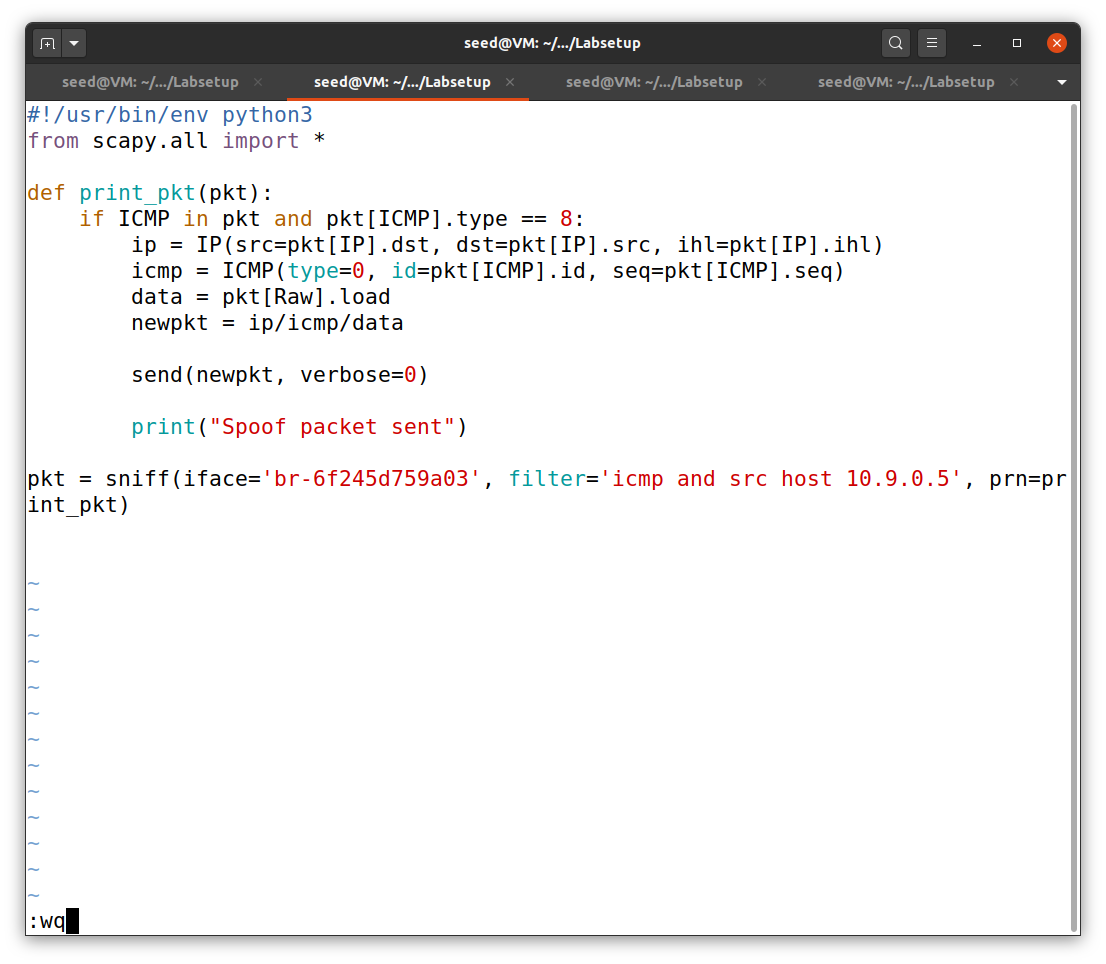
****

**Task 1.4: Sniffing and-then Spoofing**

In this final task, I used the code to sniff and spoof, as shown in Figure 22.

**Figure 22**

*sniff\_spoof.py*

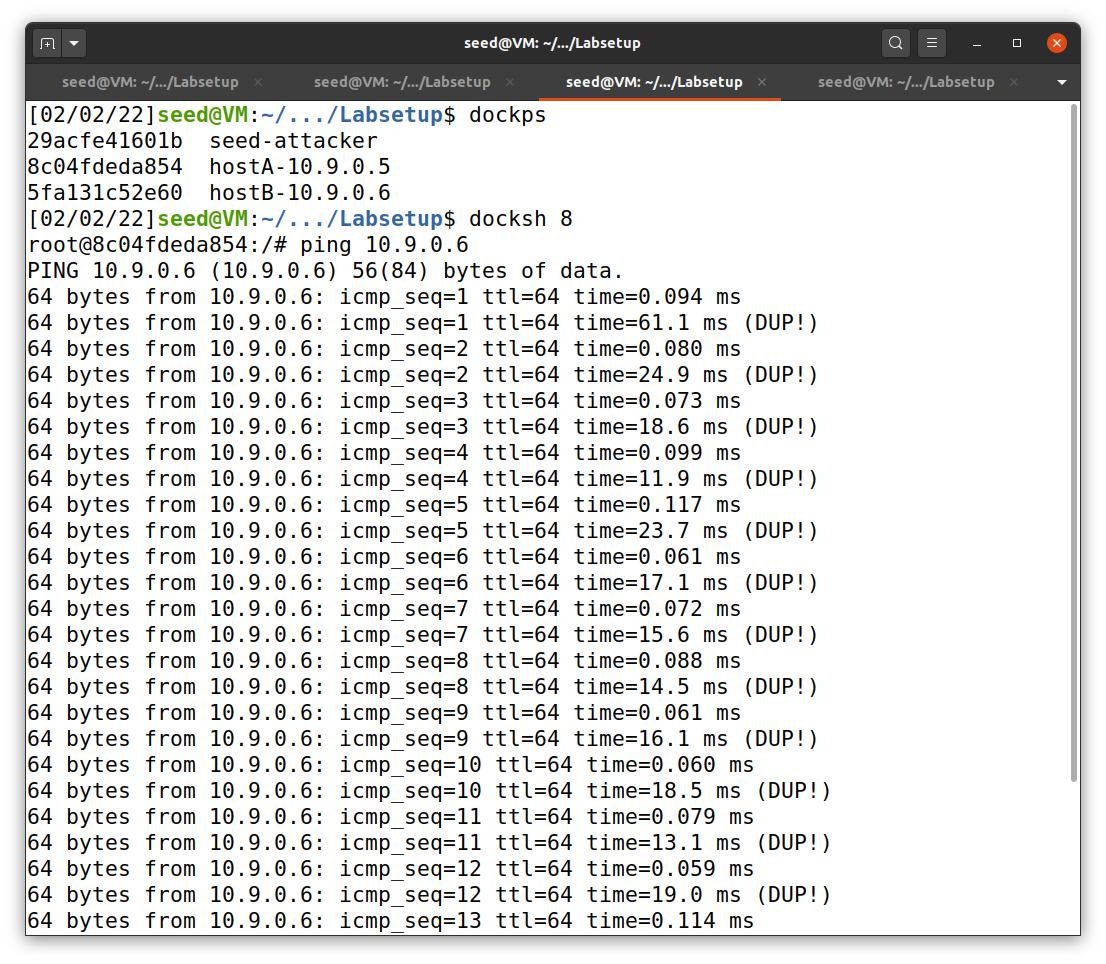


I, then, run the program above. To check the result, I docked to one of the users (host A: 10.9.0.5) and use ping command to another user (host B: 10.9.0.6). This is shown in Figure 23.

**Figure 23**

*The result of sniff and then spoof*

Text

Description automatically generated with medium confidence

hostA

There are some

duplicate replies!

hostA ping hostB

Terminal 1

**References**

Du, W. (2019). Computer & internet security: A hands-on approach (2nd ed.). Independently published.

Du, W. (n.d.). Packet sniffing and spoofing lab. SeedLabs 2.0. https://seedsecuritylabs.org/Labs\_20.04/Networking/Sniffing\_Spoofing/