**COMPUTER NETWORK SECURITY**

**LAB REPORT**

**Sniffing and Spoofing using PCAP Library**

**NAME**: **VISHWAS M**

**SRN : PES2UG20CS390**

**SEC : F**

**DATE : 03/09/2022**

**LAB NO : 2**

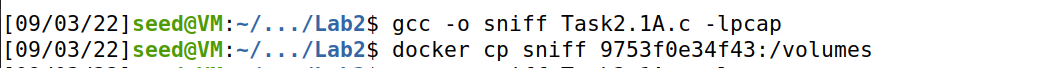
Task 2.1 : Sniffing - Writing Packet Sniffing Program

The objective of this lab is to understand the sniffing program which uses the pcap library. With pcap, the task of sniffers becomes invoking a simple sequence of procedures in the

pcap library. You should provide screenshots to show that your program runs successfully and produces expected results.

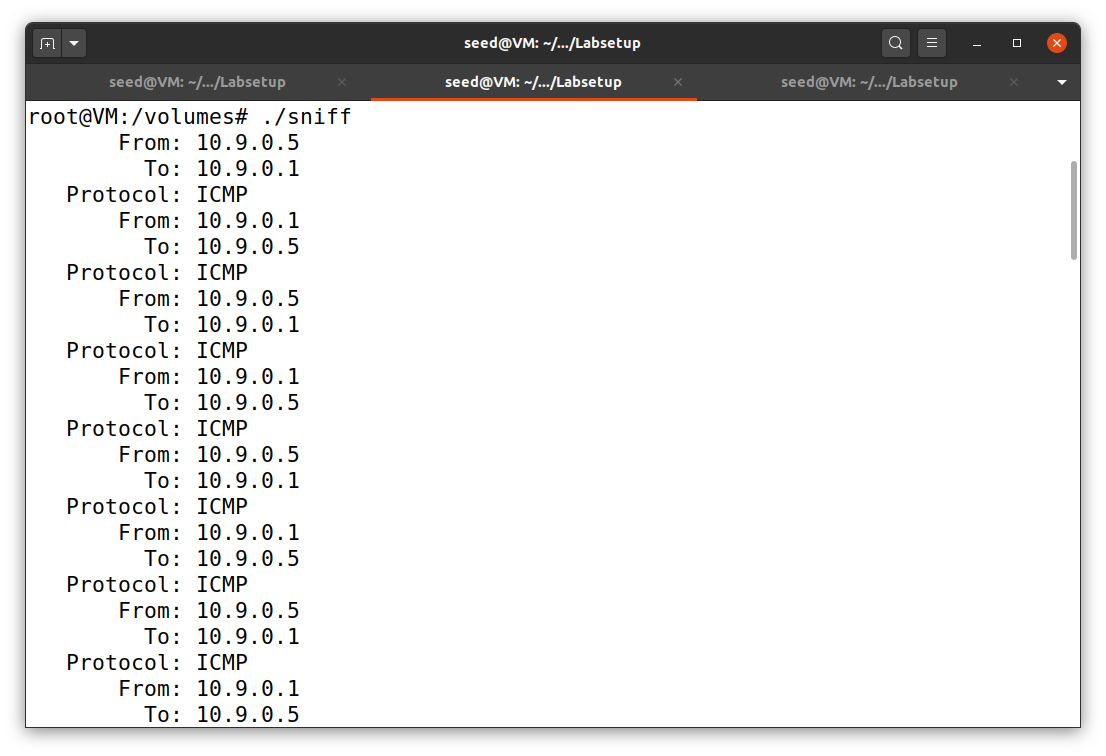
Task 2.1 A : Understanding how a Sniffer Works

Host VM:



We are compiling the code in the host machine and then we are copying the compiled code inside the container.

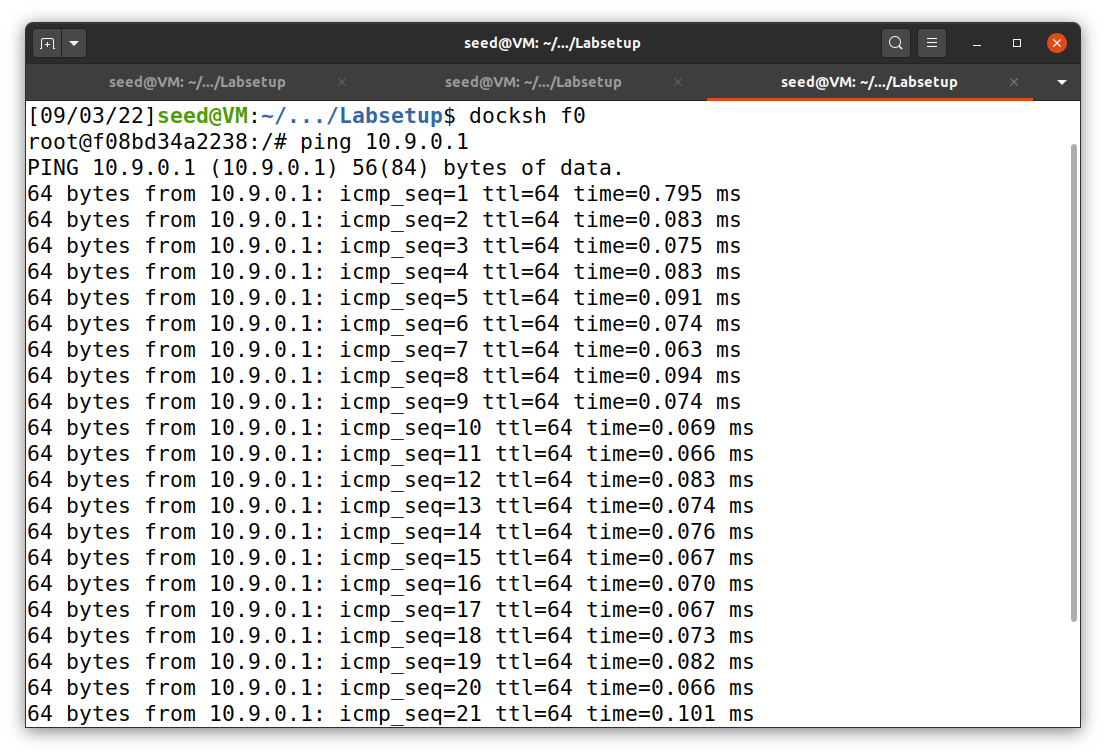
Attacker’s Terminal:



We are printing out the source and destination

IP addresses of each captured packet in the attacker’s terminal.

Host A terminal:



We are pinging 10.9.0.1 from the host A terminal.

**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 detailed explanation like

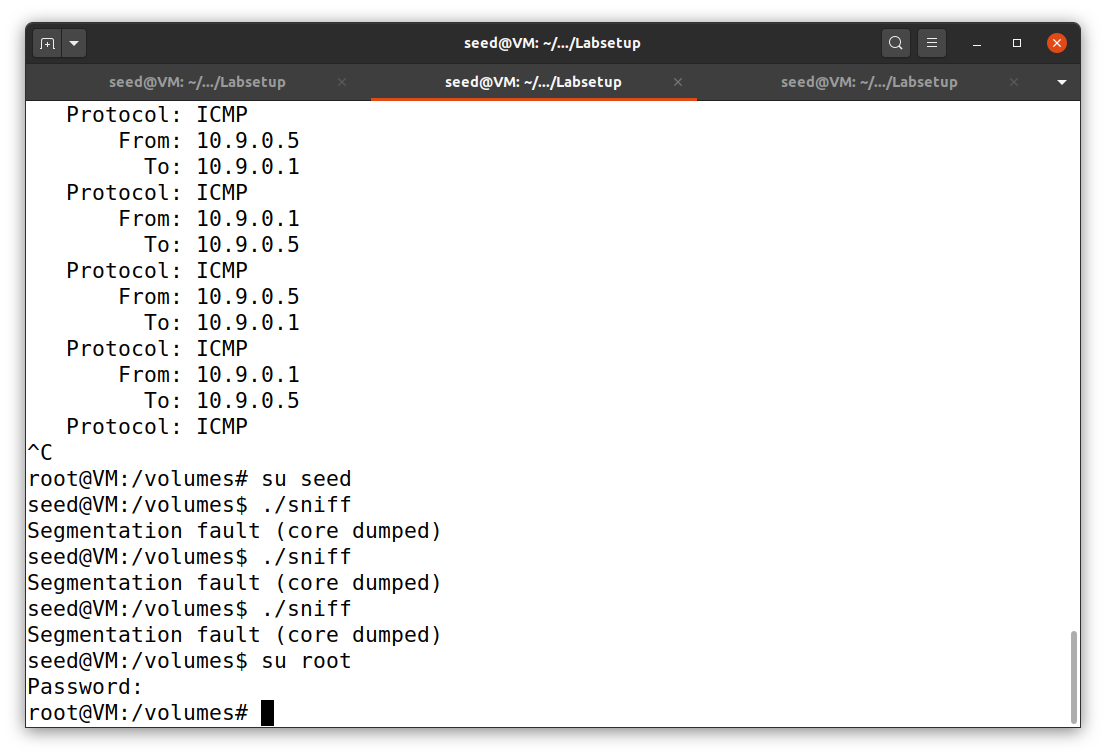
the one in the tutorial.

Ans: We are using PCAP library to sniff and spoof the packet across different hosts. It is also used to create packets and modify the packets.

**Question 2**: Why do you need the root privilege to run sniffer? Where does the program fail

if executed without the root privilege?

Ans: Packets will not be sniffed and Segmentation fault arises.

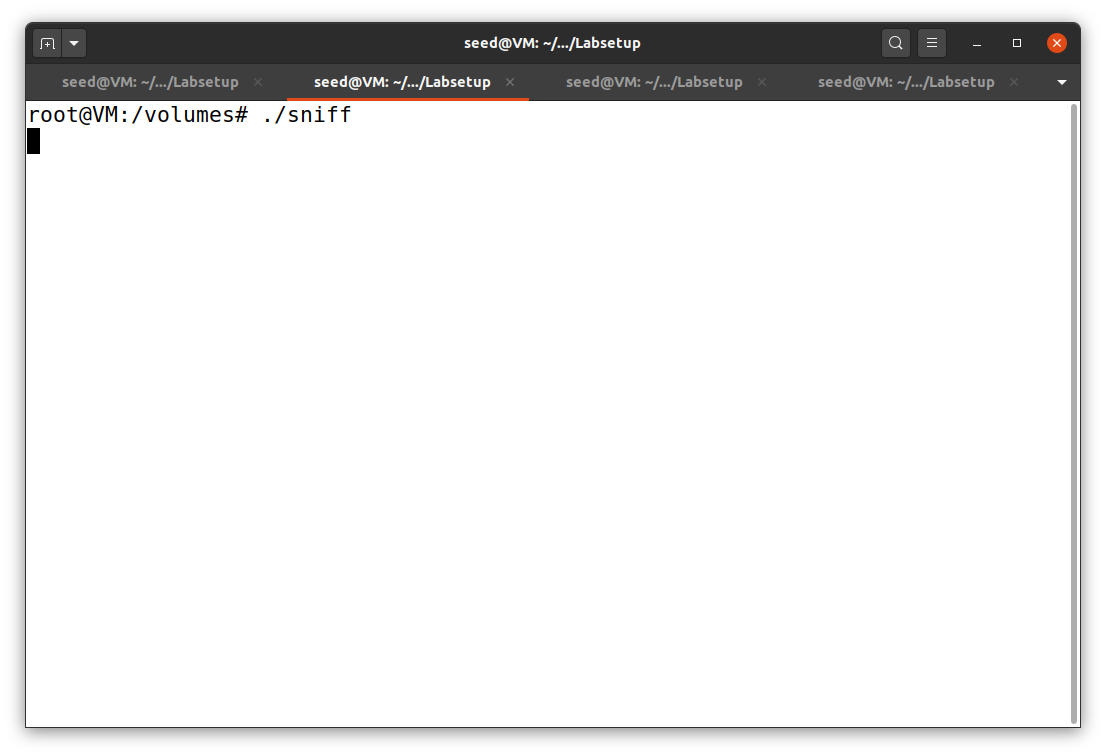


**Question 3**: Please turn on and turn off the promiscuous mode in your sniffer program. The value 1 of the third parameter in the **pcap\_open\_live() function** turns on the promiscuous mode (use 0 to turn it off). Can you demonstrate the difference when this mode is on and

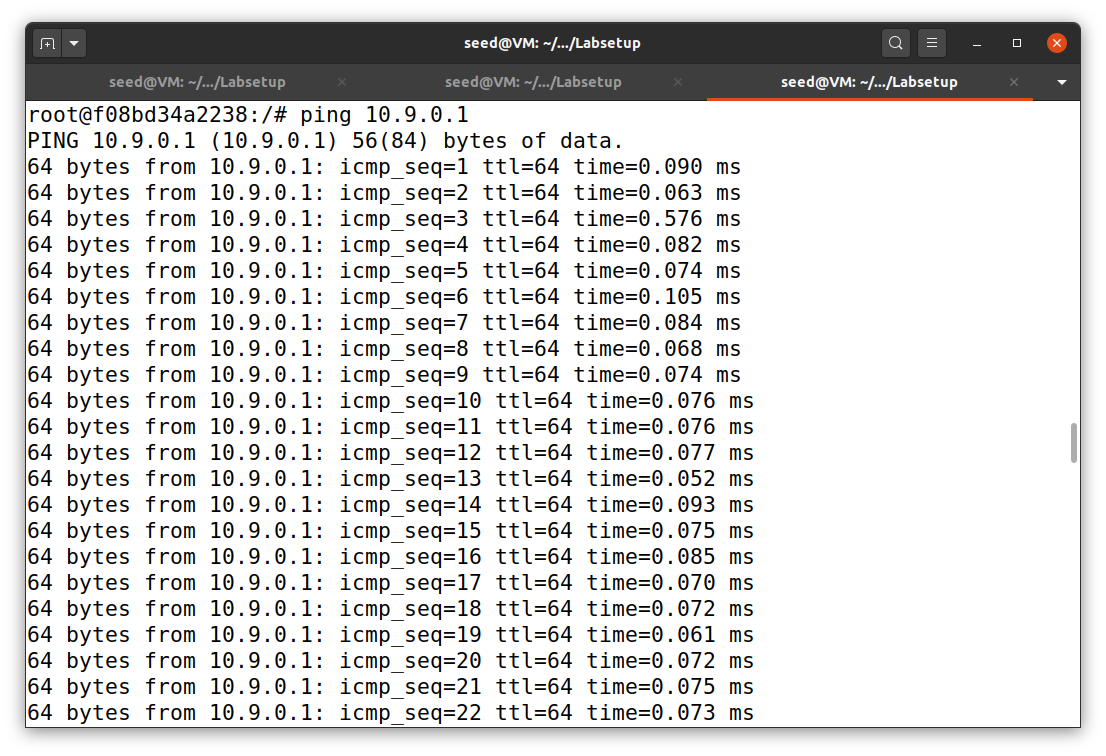
off?

Ans: If we turn off the Promiscuous mode then we packets will not be sniffed.

Attacker’s Terminal:



Host A terminal:



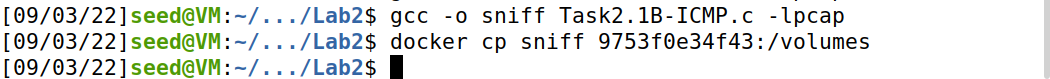
**Task 2.1 B : Writing Filters**

In this task we capture all ICMP packets between two hosts. In this task, we need to modify the pcap filter of the sniffer code. The filter will allow us to capture ICMP packets between

two hosts . Complete the filter expression in the code and show that when we send ICMP packets to IP address 1 from IP address 2 using the ping command, the sniffer program

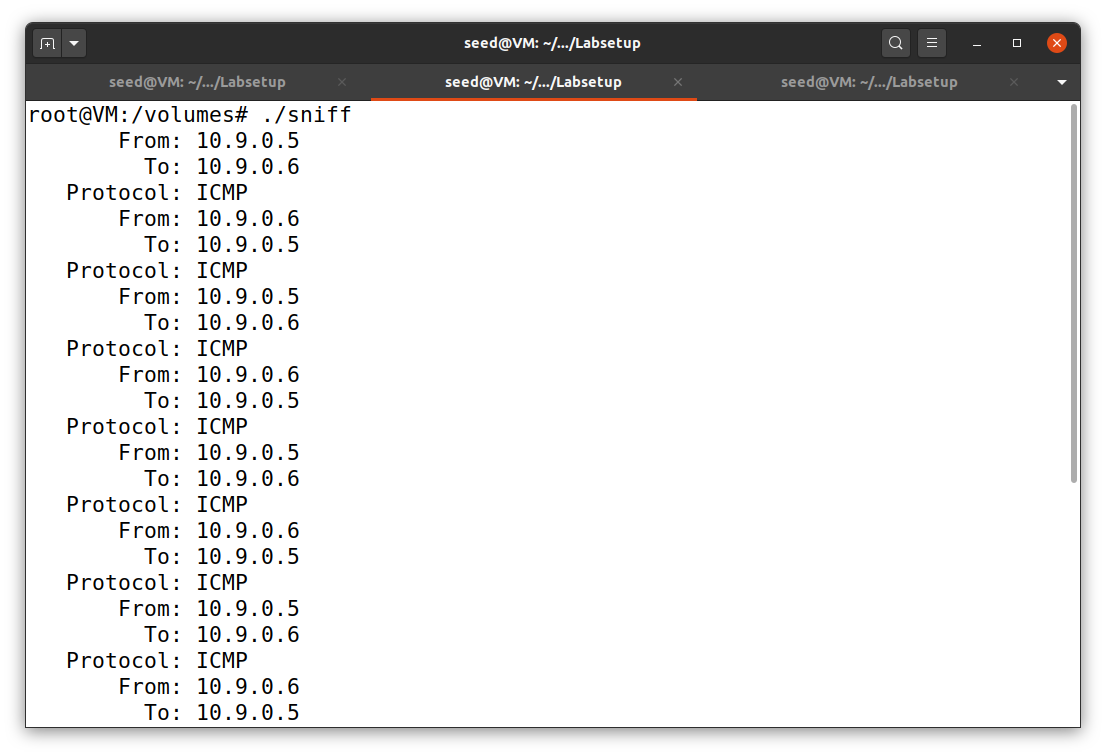
captures the packets based on the filter. Observe the packets being sent using Wireshark.

Host VM:



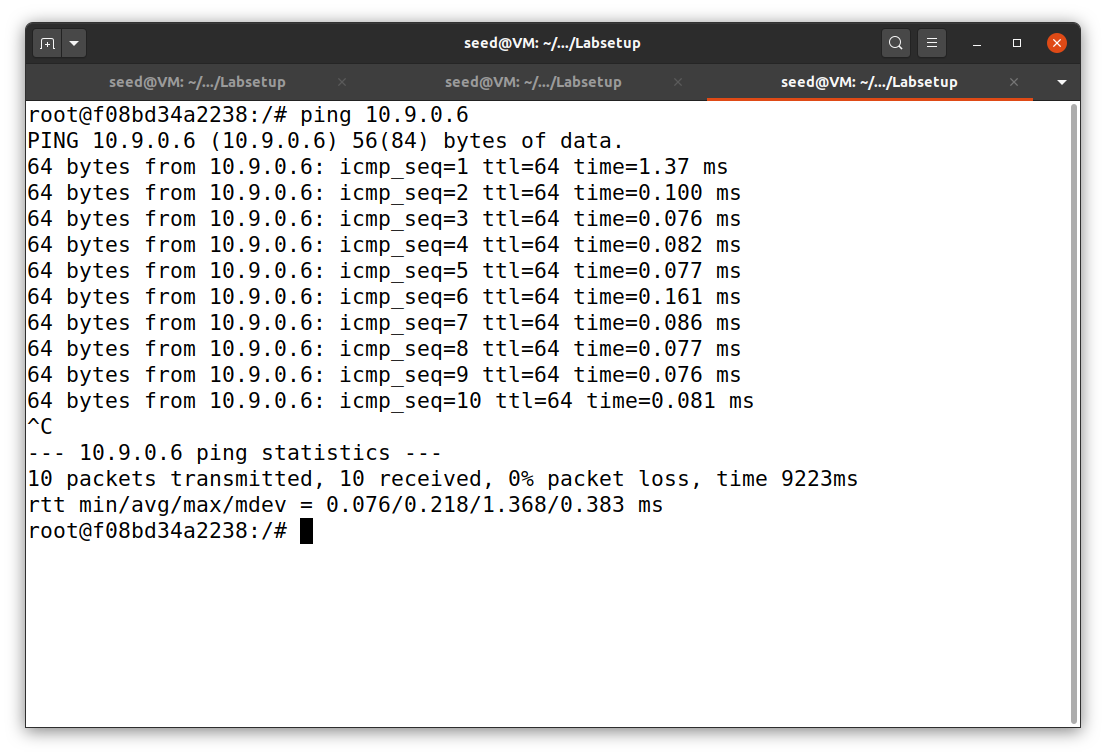
We are compiling the code in the host machine and then we are copying the compiled code inside the container.

Attacker’s Terminal:



We are capturing all the ICMP packets moving across the two hosts and displaying the source and destination address

Host A terminal:



We are pinging the IP address 10.9.0.6 from the host A terminal.

**Capture the TCP packets that have a destination port range from to sort 10 –100**

In this task we capture all TCP packets with a destination port range 10-100. Below we have

the filter expression required to filter for TCP packets in a given port range.

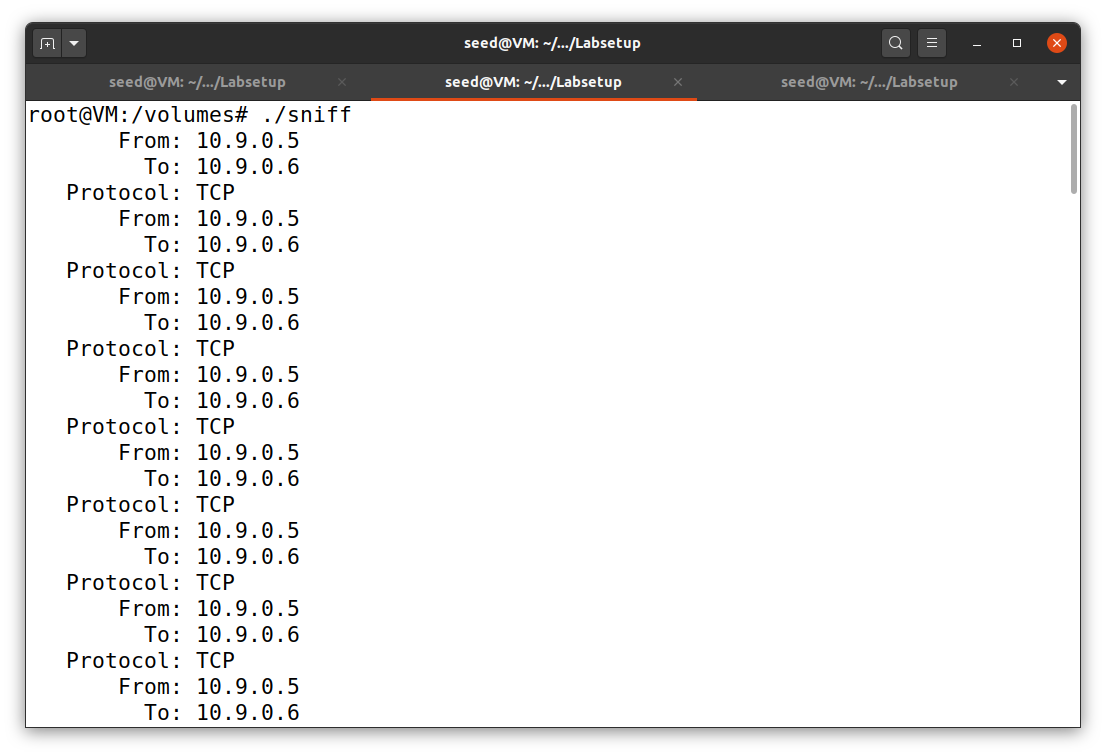
We send FTP (runs over TCP) packets to the destination machine. As telnet runs over port

21, we should be able to capture all the packets sent with destination port 21.

Host VM:

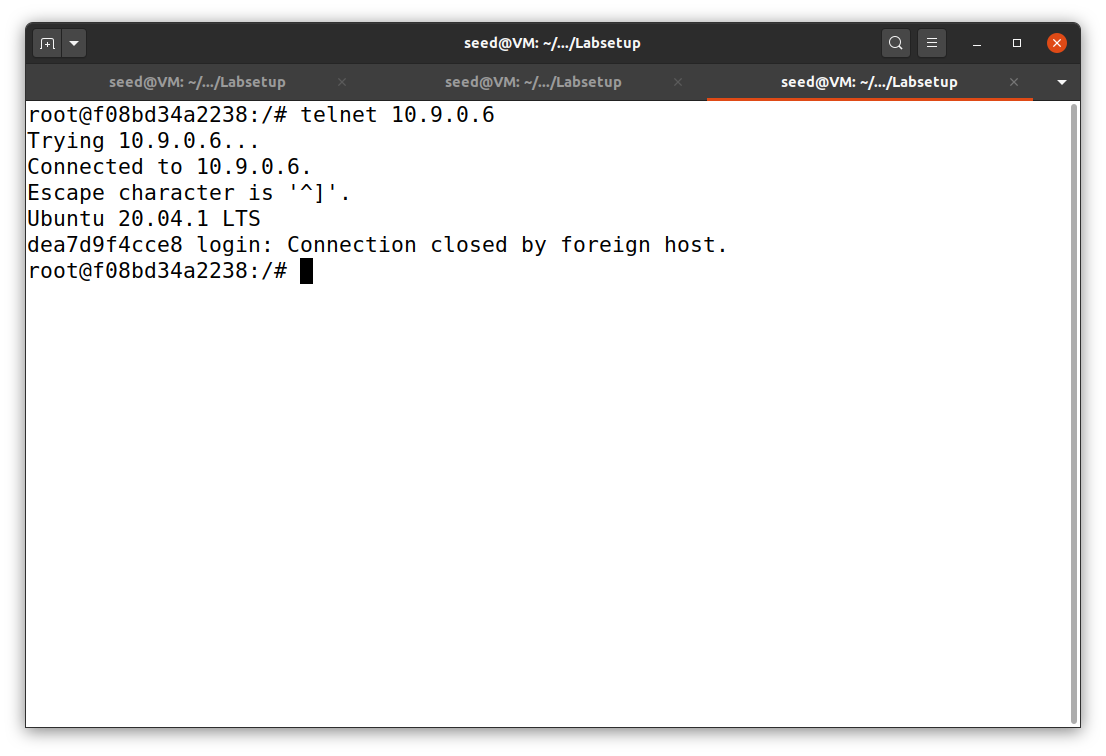


Attacker’s VM terminal:



We are capturing all the TCP packets moving across the two hosts whose destination port ranges from 10-100 and displaying the source and destination address.

Host A Terminal:



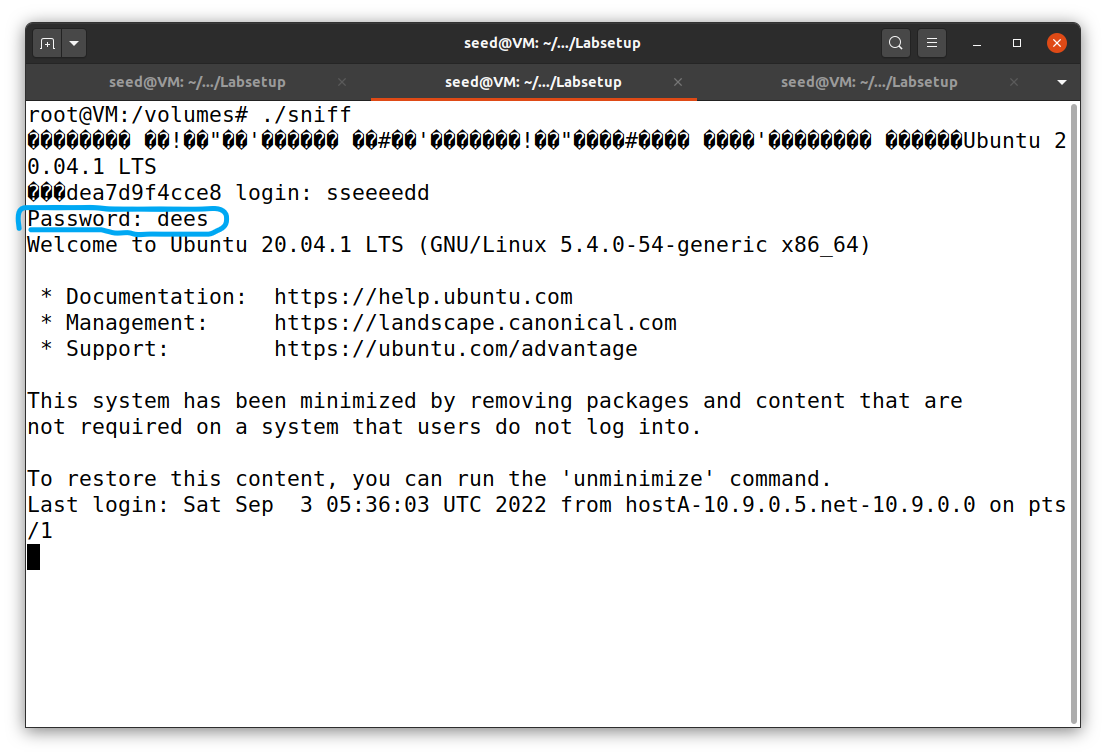
We send FTP (runs over TCP) packets to the destination machine. As telnet runs over port 21, we should be able to capture all the packets sent with destination port 21.

**Task 2.1 C : Sniffing Passwords**

Host VM:

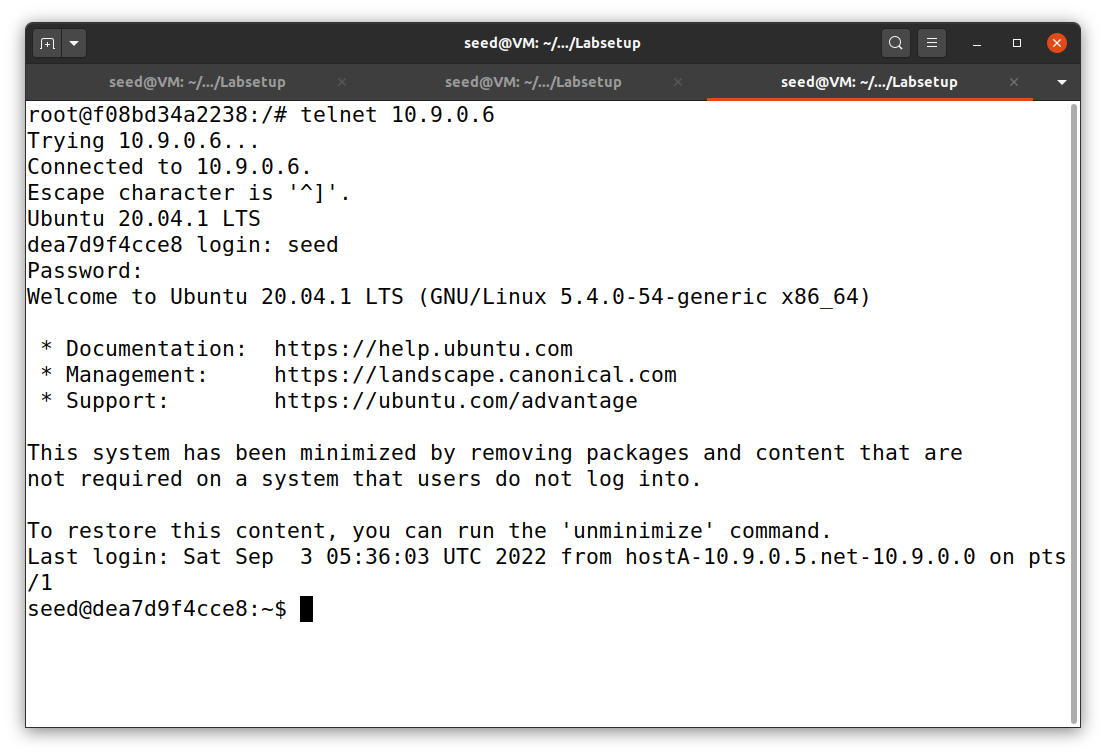


Attacker’s Host terminal:



We are capturing the password when somebody is using telnet on the network that we are monitoring. We are printing out the entire data part, and then manually mark where the password is(marked in blue colour).

Host A Terminal:



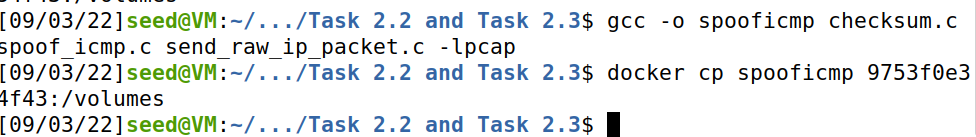
In host we are using telnet in the network that the attacker is keeping an eye on. Apparently all these data will be sniffed by the attacker.

**Task 2.2 Spoofing**

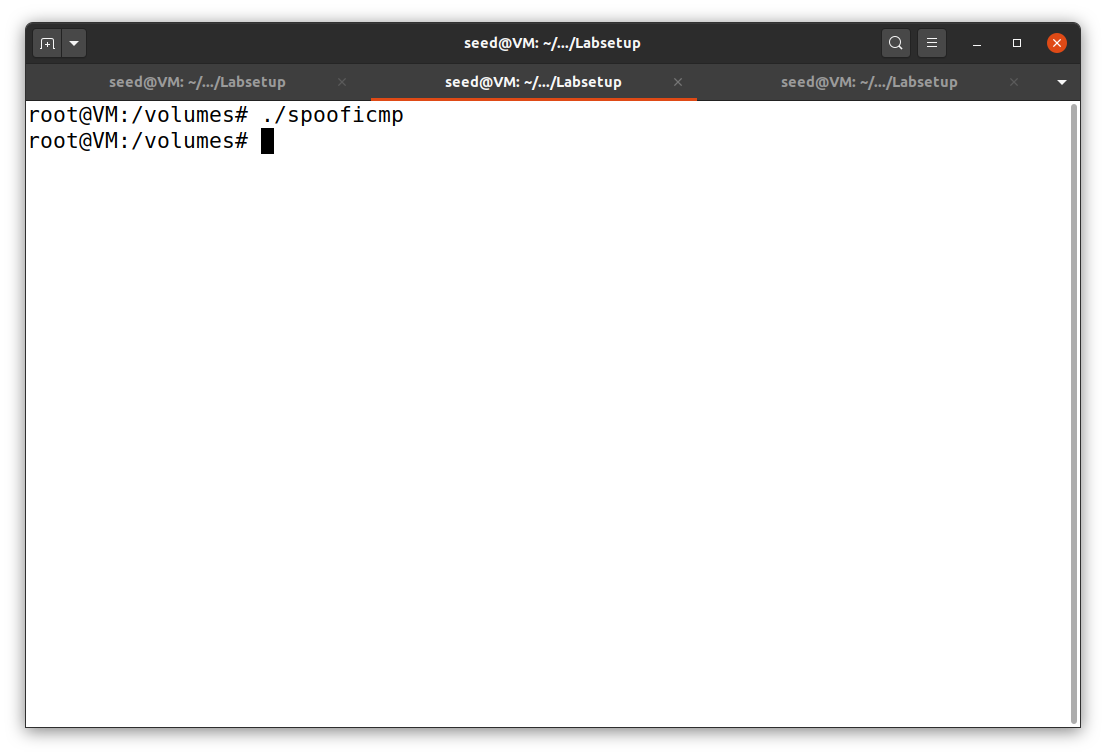
The objective of this task is to create raw sockets and send spoof packets to the user/victim machine raw sockets give programmers the absolute control over the packet construction.

**Task 2.2 B: Spoof an ICMP Echo Request**

**Host VM:**



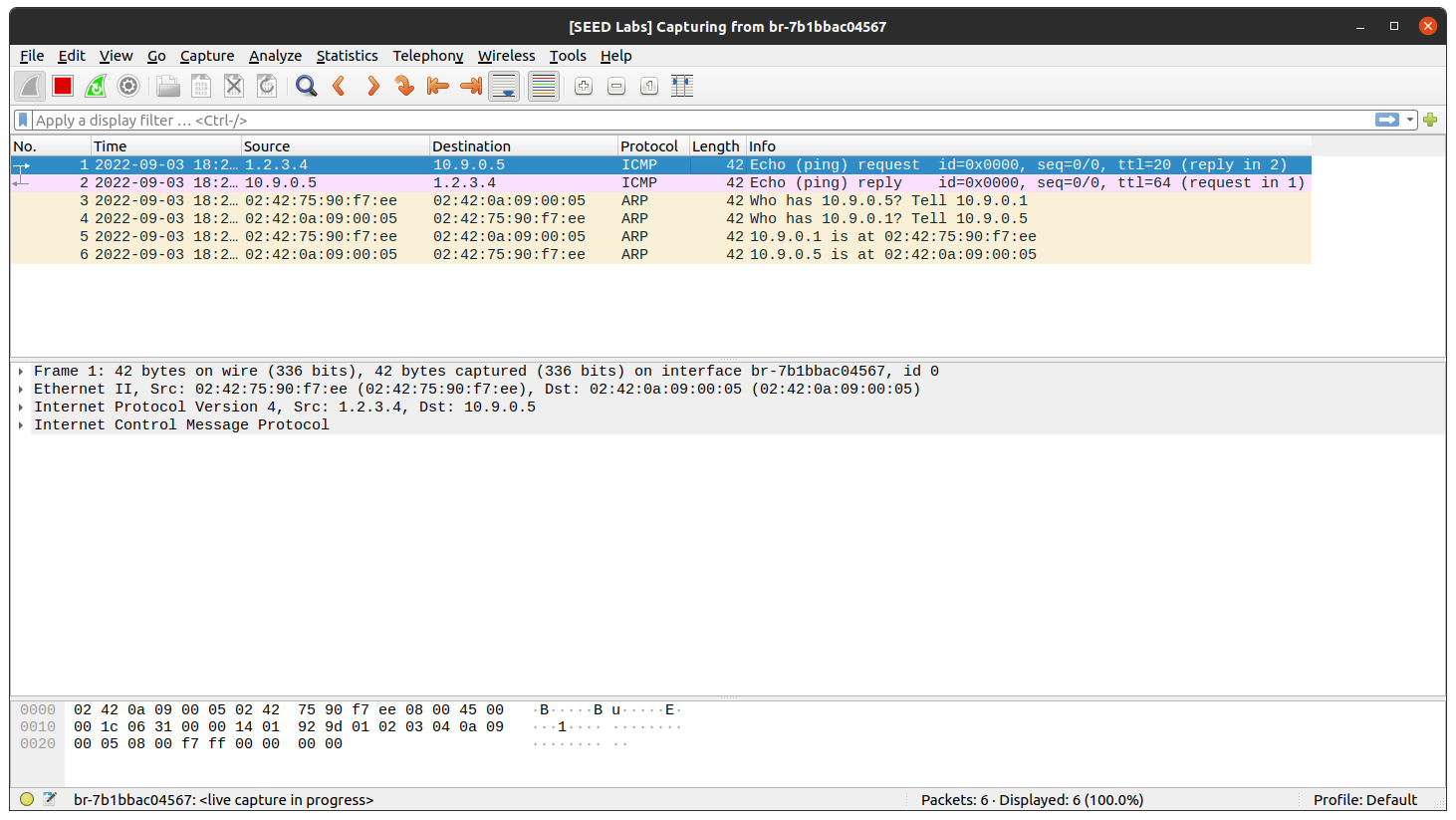
**Attacker’s Terminal:**



We are Spoofing an ICMP echo request packet on behalf of another machine (i.e., using another

machine’s IP address as its source IP address). This packet should be sent to a remote machine on the Internet (the machine must be alive).

**Wireshark Screenshot:**



**Question 4**: Using the raw socket programming, do you have to calculate the checksum for the IP header?

Ans: Yes we are calculating the checksum while using the raw socket programming.

**Question 5**: Why do you need the root privilege to run the programs that use raw sockets? Where does the program fail if executed without the root privilege?

Ans: If we did not run using root privileges, then we will not be able to sniff any packet because segmentation error occurs when we run the command in the attacker’s terminal.

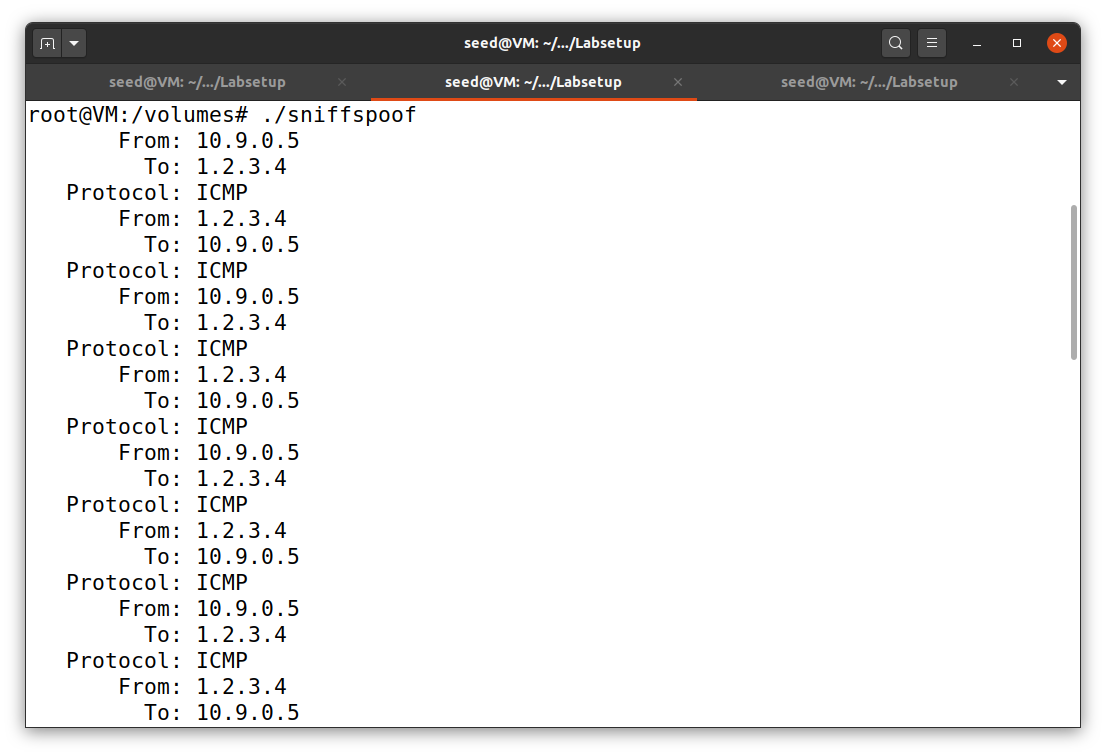
**Task 2.3 Sniff and then Spoof**

In this task, the victim machine pings a non-existing IP address “1.2.3.4”. As the attacker machine is in the same network, it sniffs the request packet, creates a new echo reply packet with IP and ICMP header and sends it to the victim machine. Hence the user will always receive an echo reply from a non-existing IP address indicating that the machine is alive.

We create a buffer of maximum length and fill it with an IP request header.

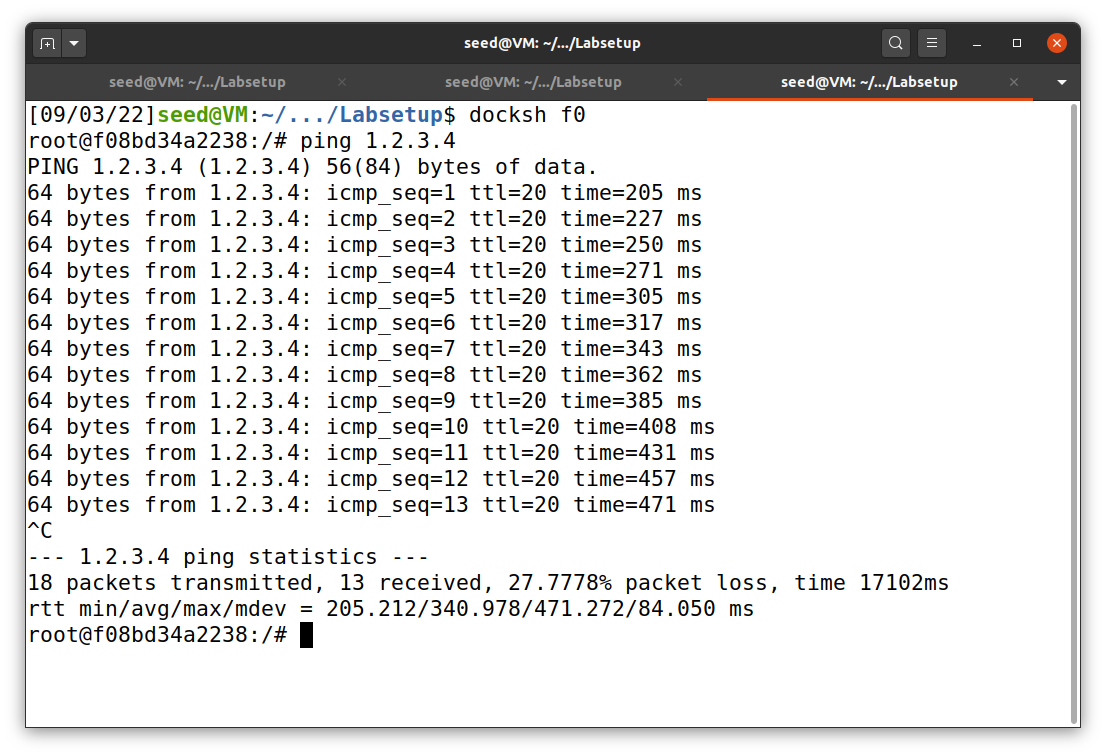
We modify the IP header and ICMP header with our response data. In the new IP header, we interchange the source IP address and destination IP address and send the new IP packet using the raw sockets.

Attacker’s Terminal:



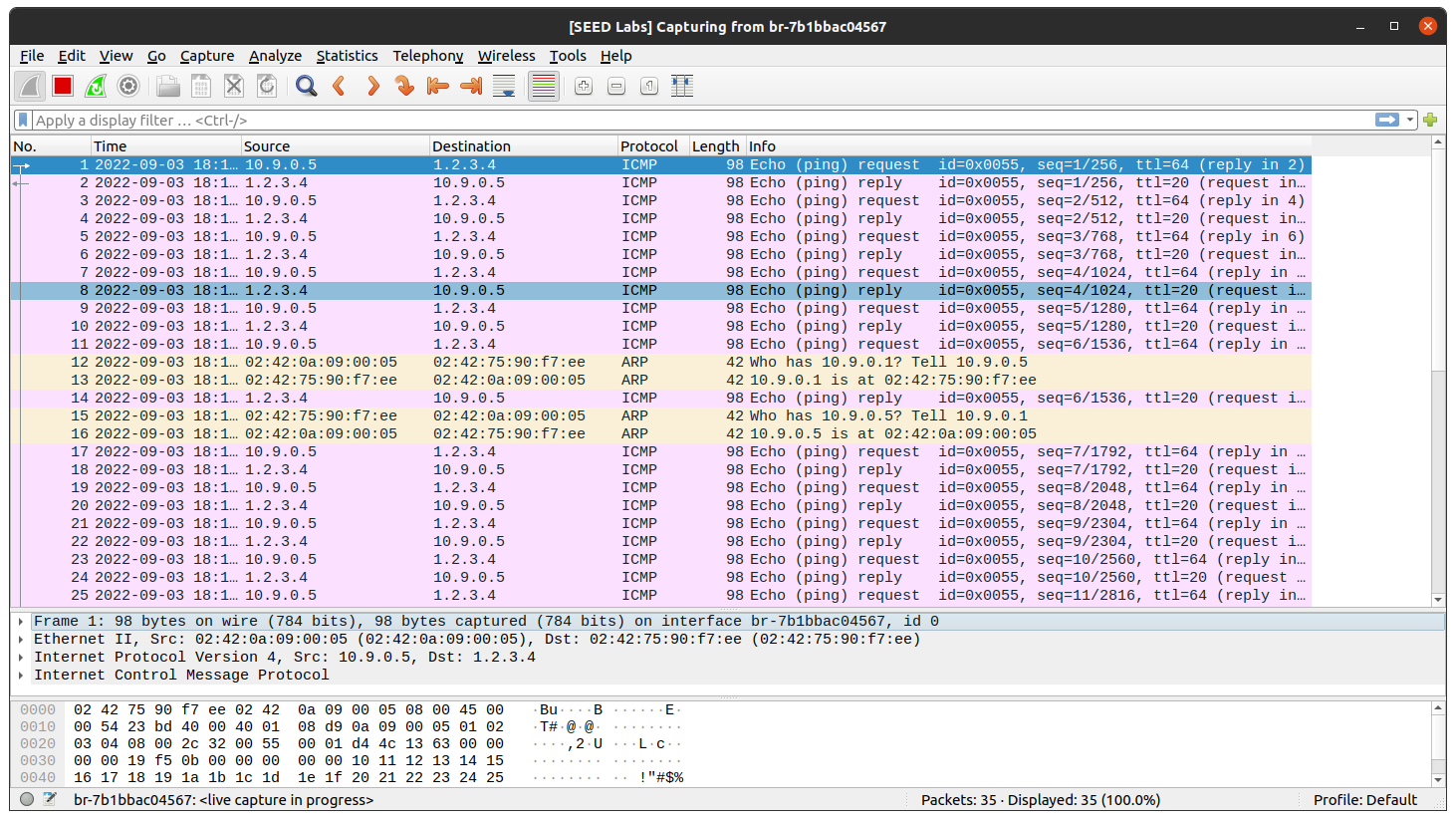
Here we can see the source and destination IP address. We can observe clearly that both request and reply are getting transferred between each host.

Host A Terminal:



Here the host is pinging the IP address 1.2.3.4 which is an unknown IP address.

Wireshark Terminal:



The attacker is spoofing the packet from the host and gets those packets from raw socket. When this happens the host machine thinks that the IP address really exists. The attacker who sniffs the packet will change the data and again sends the echo reply back to the host. This repeats every time the host tries to contact this unknown IP address.