Name	Naman Choudhary
SRN	PES2UG20CS209
Section	D

Sniffing and Spoofing Lab

Assignment 2

Task 2.1 : Sniffing - Writing Packet Sniffing Program

Task 2.1 A : Understanding how a Sniffer Works

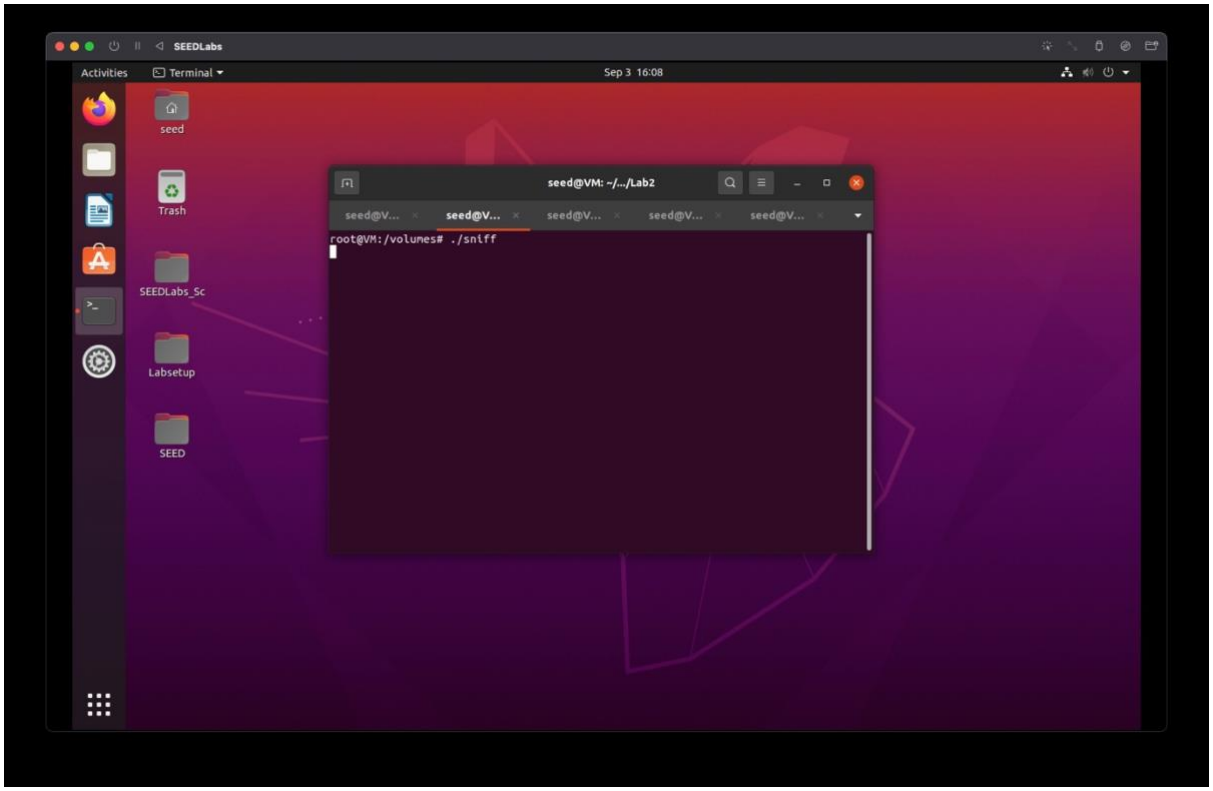
On the host VM :
 # gcc -o sniff Task2.1A.c -lpcap
 # docker cp sniff [Attacker machine docker container ID]:/volumes

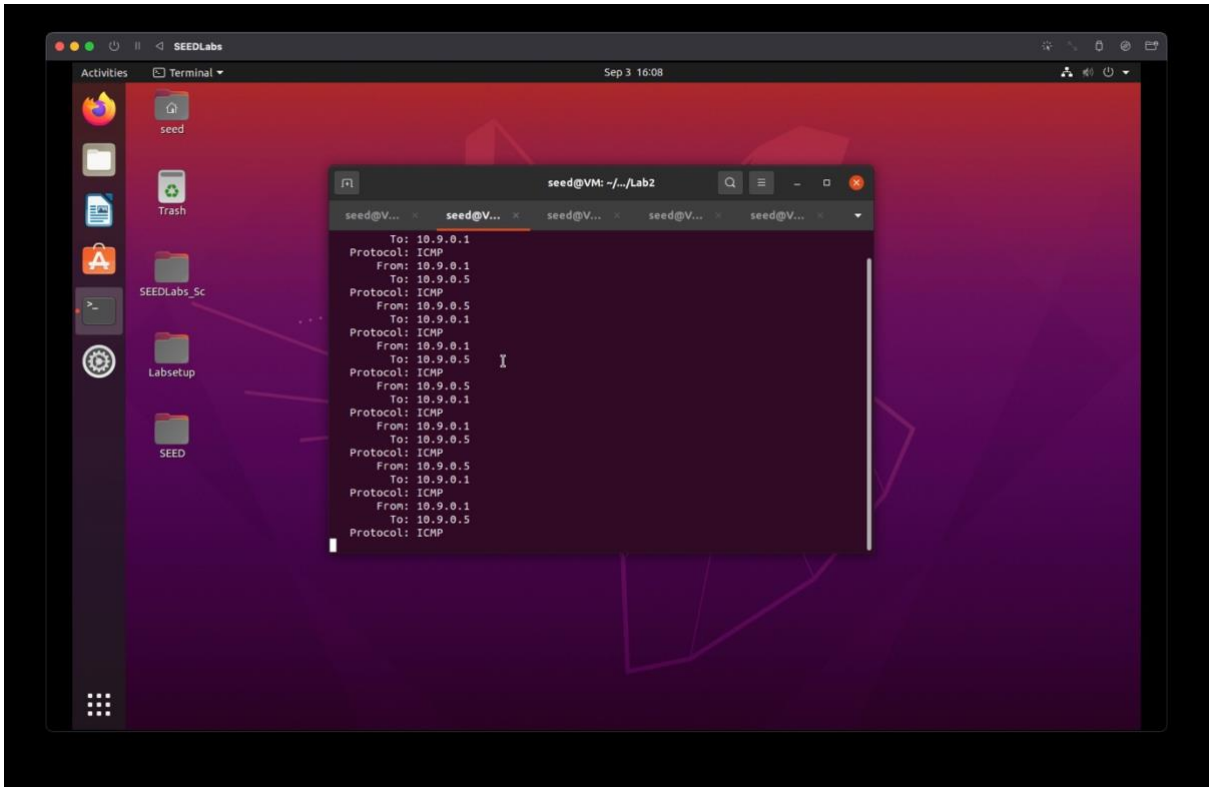
On the Attacker container run the command: # ./sniff

On Host A terminal : # ping 10.9.0.1

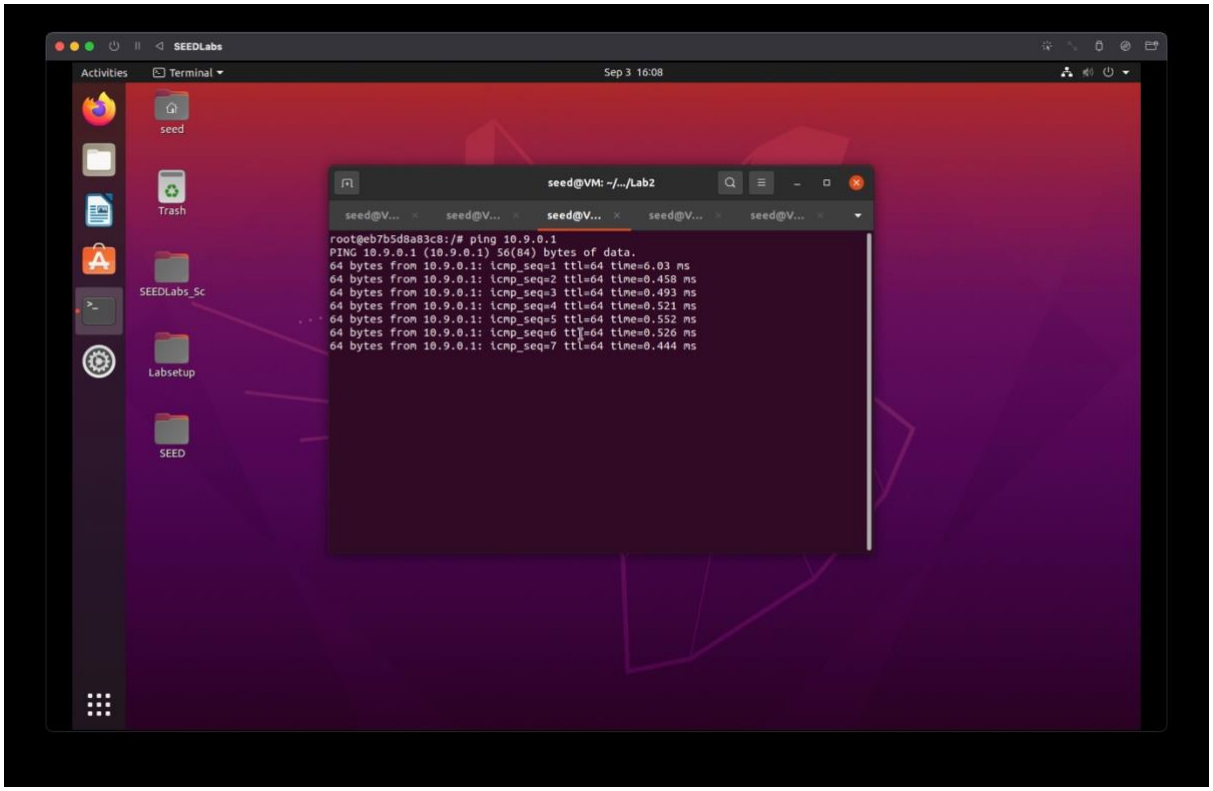
Provide screenshots of your observations.

Attacker:


SEEDLabs
Activities
Terminal
Sep 3 16:08
seed
Trash
SEEDLabs_Sc
Labsetup
SEED
seed@VM: ~/.../Lab2
seed@V...
root@VM:/volumes# ./sniff


SEEDLabs
Activities
Terminal
Sep 3 16:08
seed
Trash
SEEDLabs_Sc
Labsetup
SEED
seed@VM: ~/.../Lab2
seed@V...
To: 10.9.0.1
Protocol: ICMP
From: 10.9.0.1
To: 10.9.0.5
Protocol: ICMP
From: 10.9.0.5
To: 10.9.0.1
Protocol: ICMP
From: 10.9.0.1
To: 10.9.0.5
Protocol: ICMP
From: 10.9.0.5
To: 10.9.0.1
Protocol: ICMP
From: 10.9.0.1
To: 10.9.0.5
Protocol: ICMP
From: 10.9.0.5
To: 10.9.0.1
Protocol: ICMP
From: 10.9.0.1
To: 10.9.0.5
Protocol: ICMP

Host A:



Observation: Attacker was able to sniff ICMP packets from source:10.9.0.1 to destination:10.9.0.5 and vice versa

In addition, please answer the following questions:

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.

➔ The order of the library calls are

i. pcap_openlive
ii. pcap_openlive
iii. pcap_setfilter
iv. pcap_setfilter
v. pcap_setfiler
vi. pcap_close

Question 2: Why do you need the root privilege to run sniffex? Where does the program fail if executed without the root privilege?

➔ Root privilege is necessary so that, we can sniff the packets
➔ The program fails showing the error message ‘Segmentation fault’ when executed without root privilege

On the Attacker container run the command :

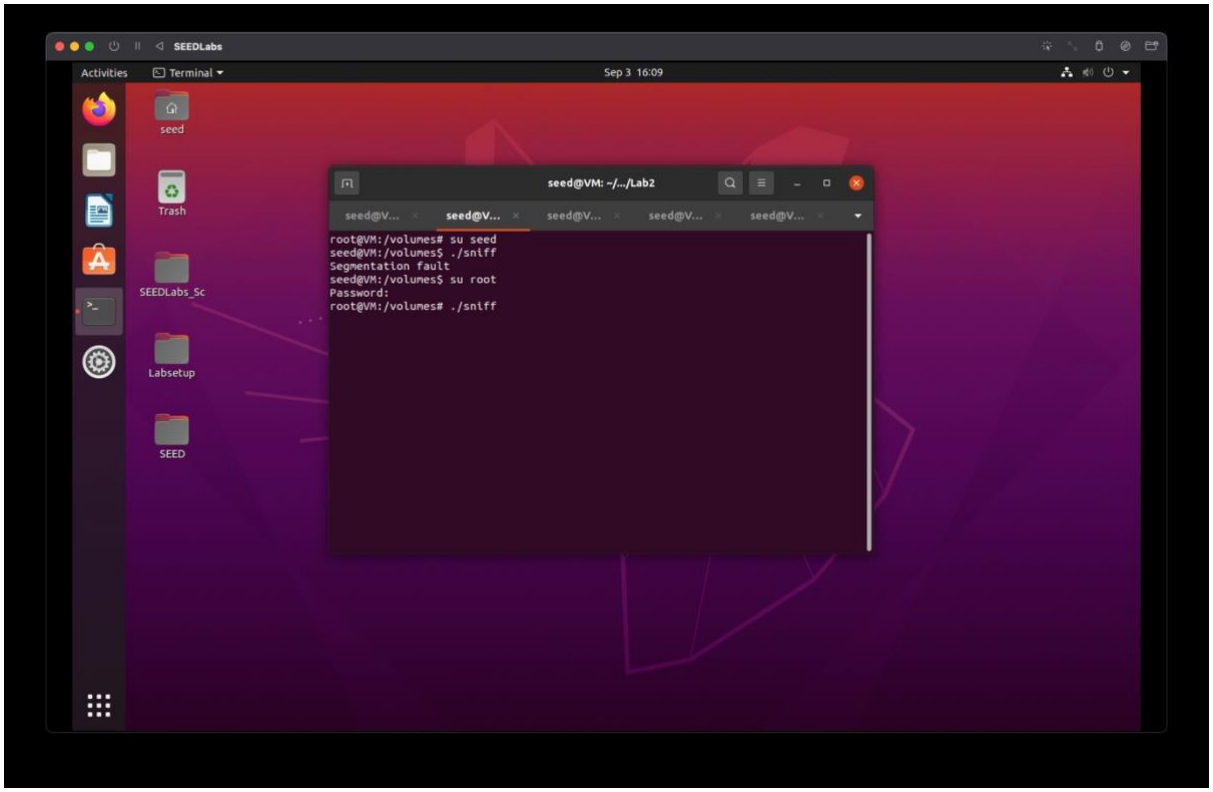
 # su seed

 # ./sniff

After running the sniff program run the command to return to root user on the attacker container:

 # su root

Provide a screenshot of your observations.



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?

➔ When the value in the third parameter of pcap_open_live is changed to 0, that is, the promiscuous mode is turned off, then we notice that the packets cannot be sniffed, and hence promiscuous mode should always be turned on to sniff packets

Change the code given in line 69 of Task2.1A.c file to the following :

handle = pcap_open_live("br-**", BUFSIZ, 0, 1000, errbuf);**

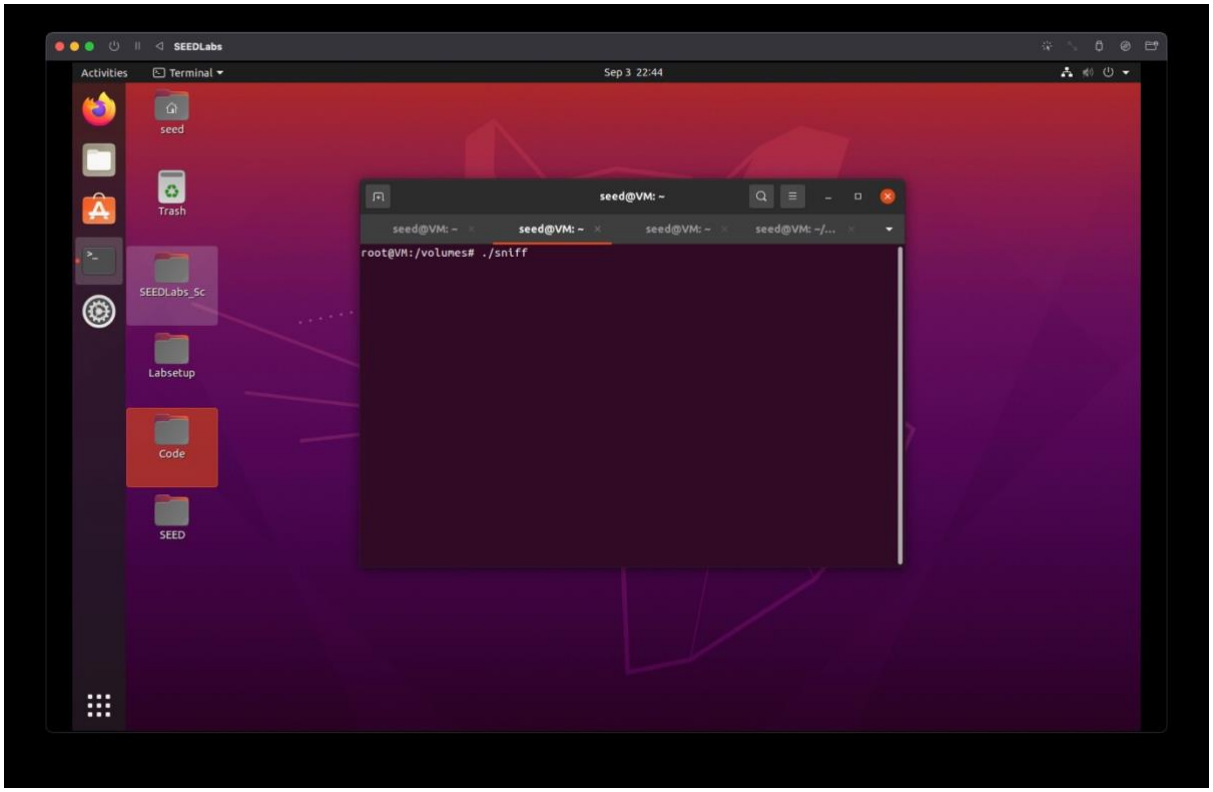
On the host VM :
 # gcc -o sniff Task2.1A.c -lpcap
 # docker cp sniff [Attacker machine docker container ID]:/volumes

On the Attacker terminal run the command: # ./sniff

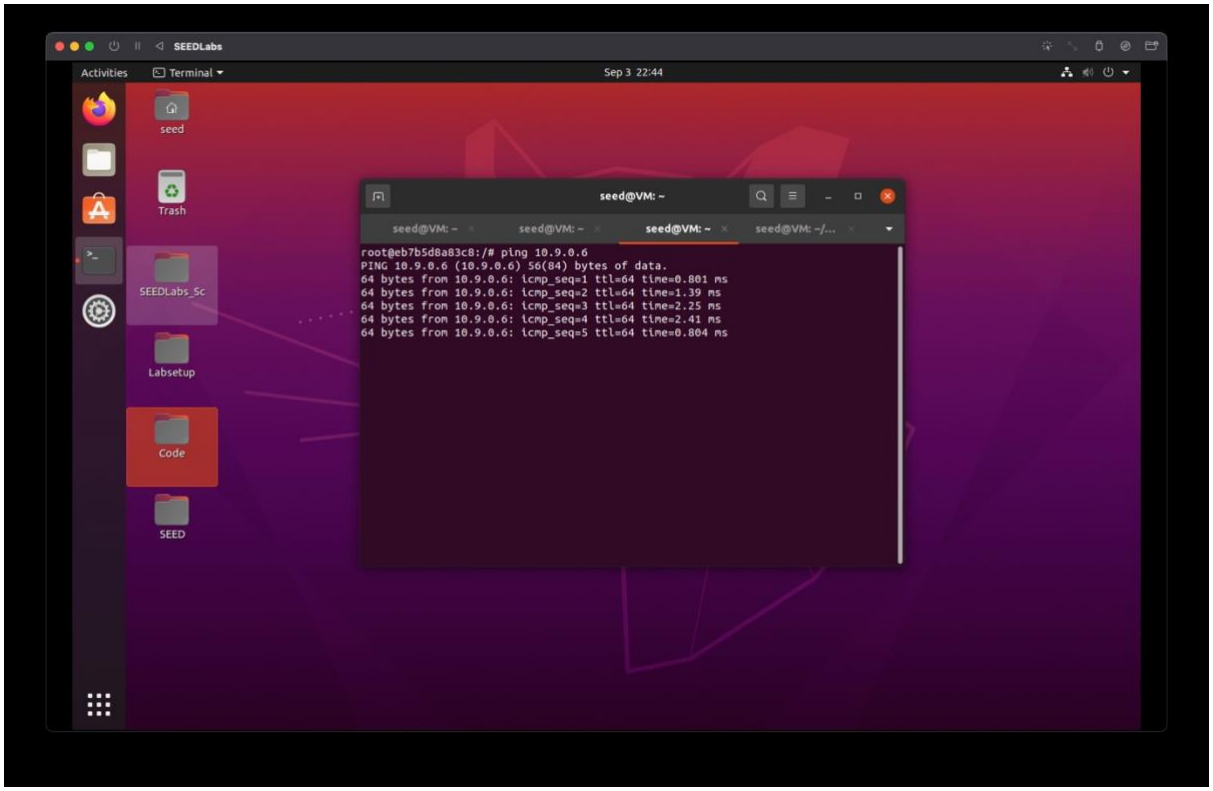
On Host A terminal : # ping 10.9.0.6

Provide screenshots of your observations.

Attacker:



Host A:



Observation: Attacker was able to sniff ICMP packets from source:10.9.0.6 to destination:10.9.0.5 and vice versa

Task 2.1 B : Writing Filters

Capture the ICMP packets between two specific hosts

On the host VM :
 # gcc -o sniff Task2.1B-ICMP.c -lpcap
 # docker cp sniff [Attacker machine docker container ID]:/volumes

On the Attacker terminal run the command:

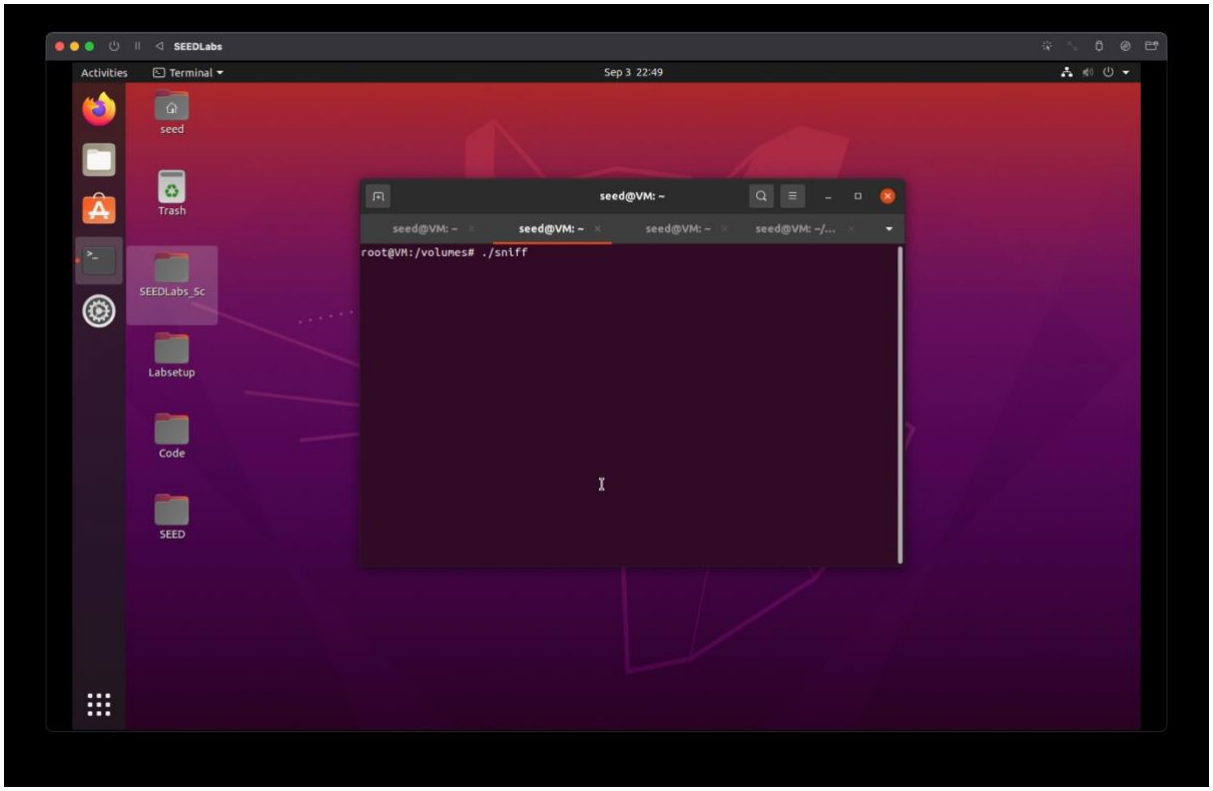
 # ./sniff

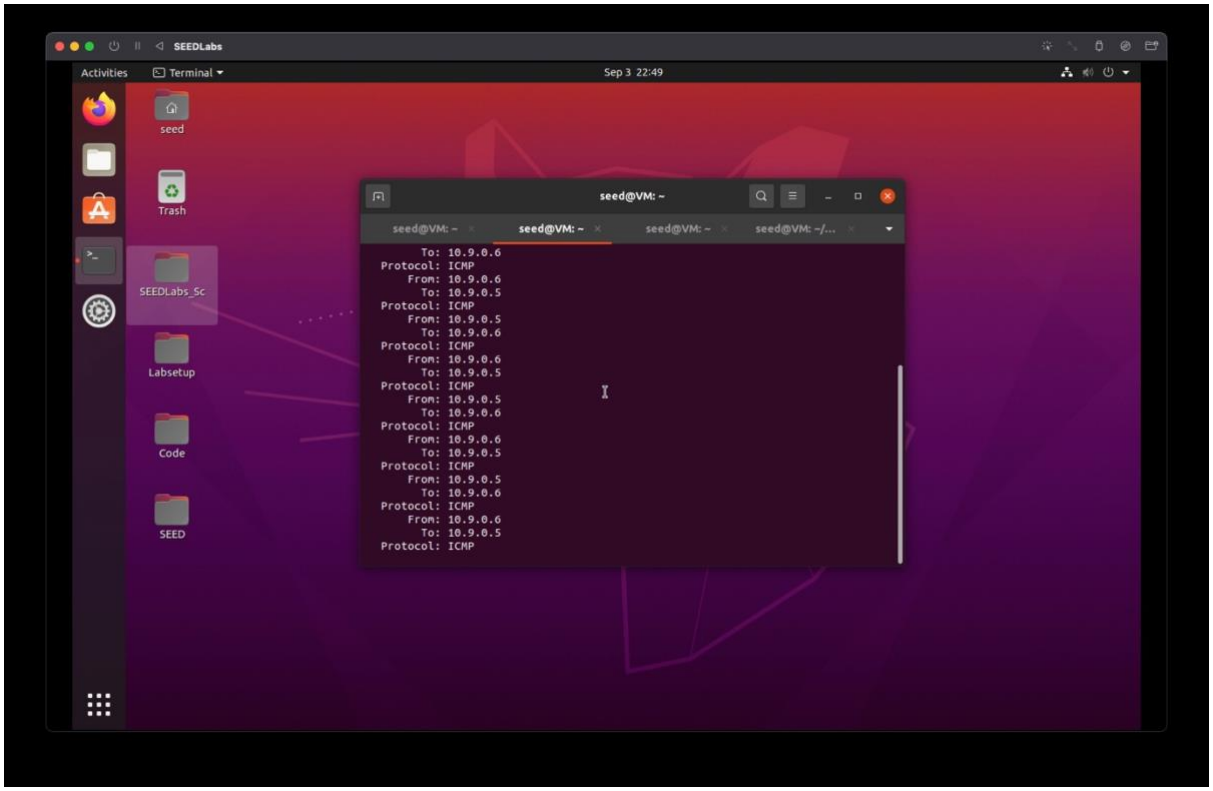
In the host A machine ping any ip address

 # ping 10.9.0.6

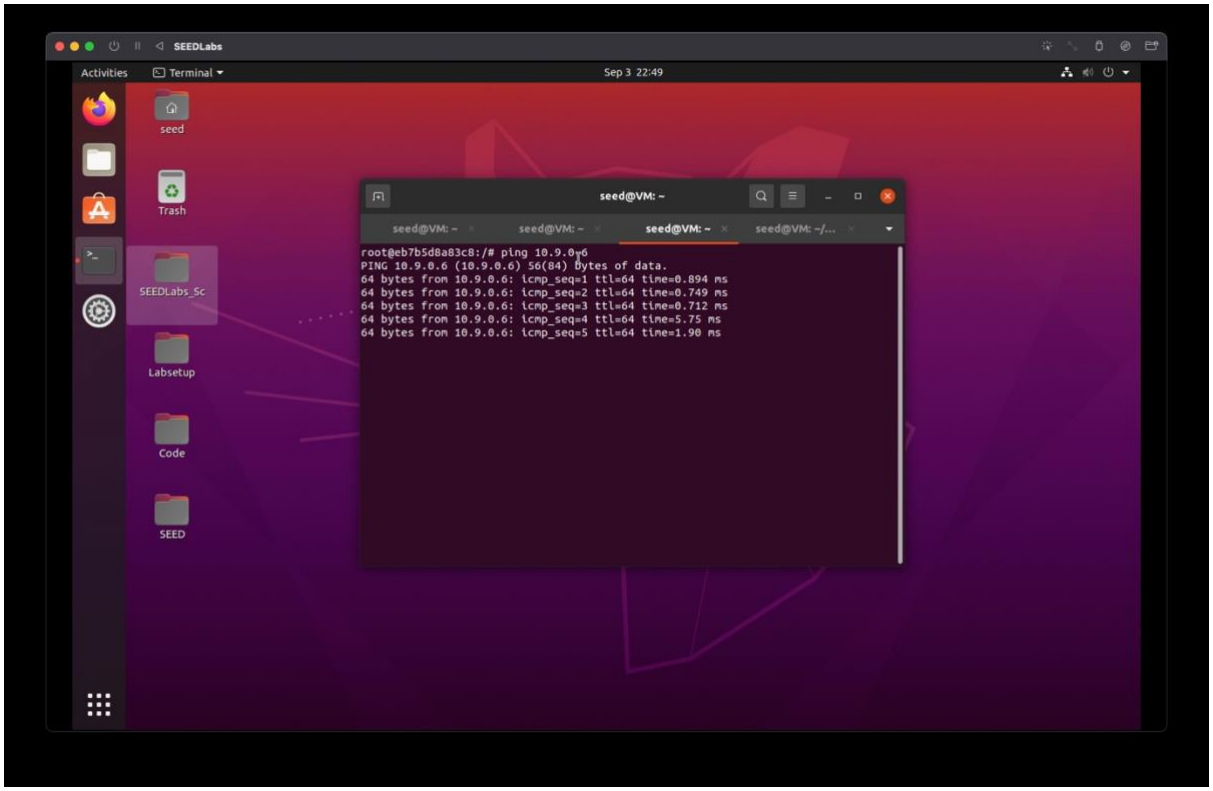
Provide screenshots of your observations.

Attacker:


SEEDLabs
Activities
Terminal
Sep 3 22:49
seed
Trash
SEEDLabs_Sc
Labsetup
Code
SEED
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed@VM: ~/...
root@VM:/volumes# ./sniff


SEEDLabs
Activities
Terminal
Sep 3 22:49
seed
Trash
SEEDLabs_Sc
Labsetup
Code
SEED
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed@VM: ~/...
To: 10.9.0.6
Protocol: ICMP
From: 10.9.0.6
To: 10.9.0.5
Protocol: ICMP
From: 10.9.0.5
To: 10.9.0.6
Protocol: ICMP
From: 10.9.0.6
To: 10.9.0.5
Protocol: ICMP
From: 10.9.0.5
To: 10.9.0.6
Protocol: ICMP
From: 10.9.0.6
To: 10.9.0.5
Protocol: ICMP
From: 10.9.0.5
To: 10.9.0.6
Protocol: ICMP
From: 10.9.0.6
To: 10.9.0.5
Protocol: ICMP

Host A:



Observation: Attacker was able to sniff ICMP packets from source:10.9.0.6 to destination:10.9.0.5 and vice versa

Capture the TCP packets that have a destination port range from to sort 10 - 100.

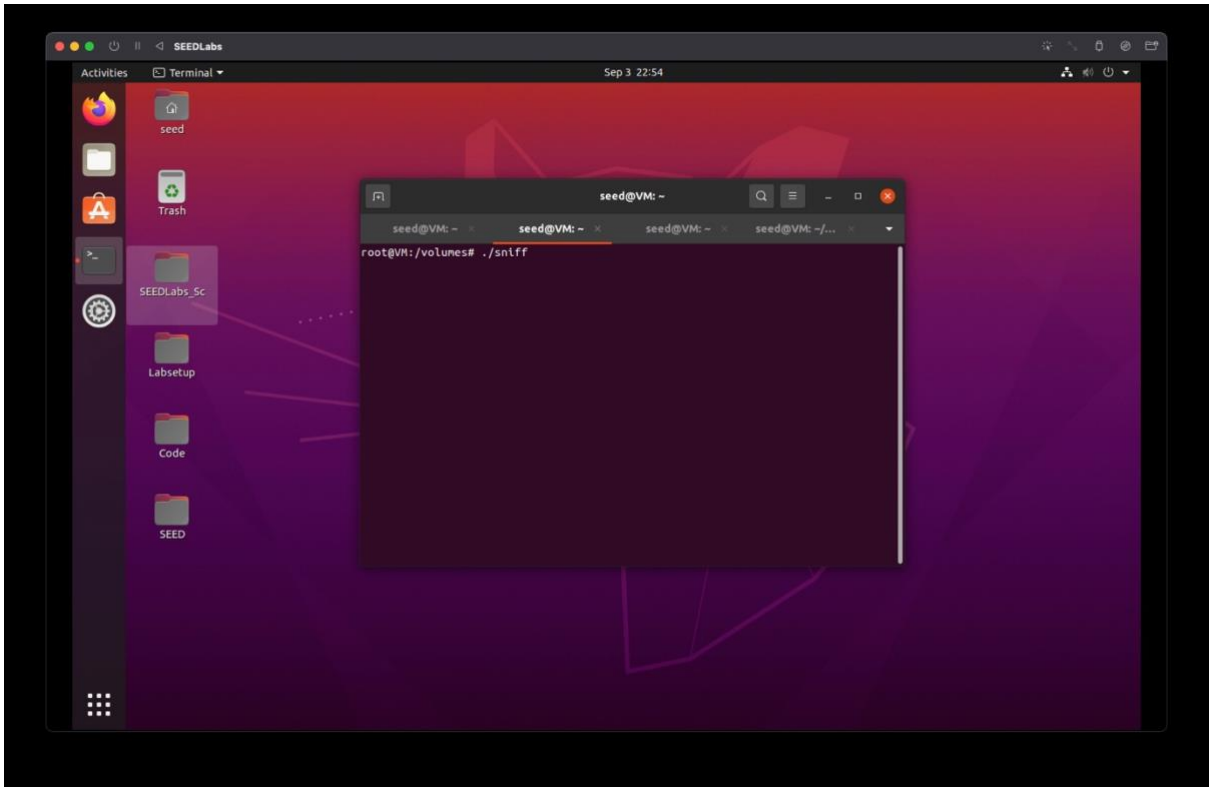
On the host VM :
 # gcc -o sniff Task2.1B-TCP.c -lpcap
 # docker cp sniff [Attacker machine docker container ID]:/volumes

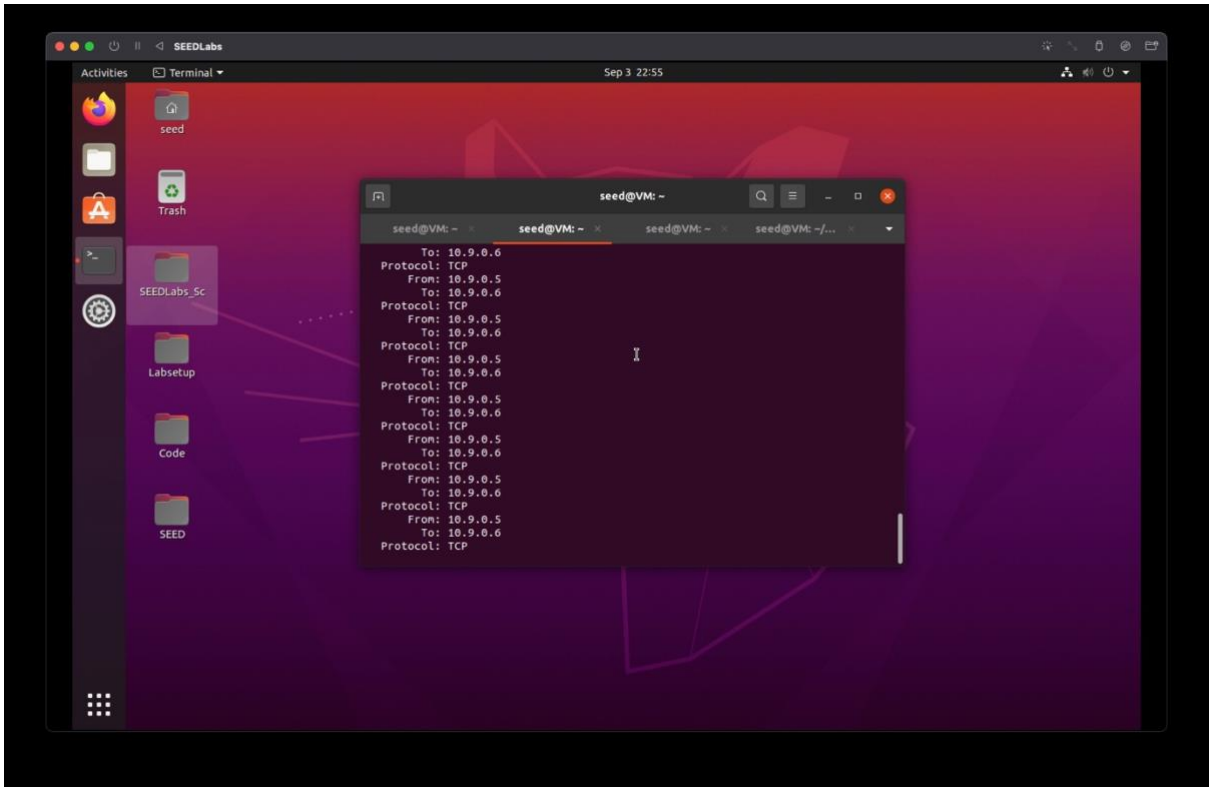
On Attacker Machine terminal : # ./sniff

On Host A terminal :
 # telnet 10.9.0.6

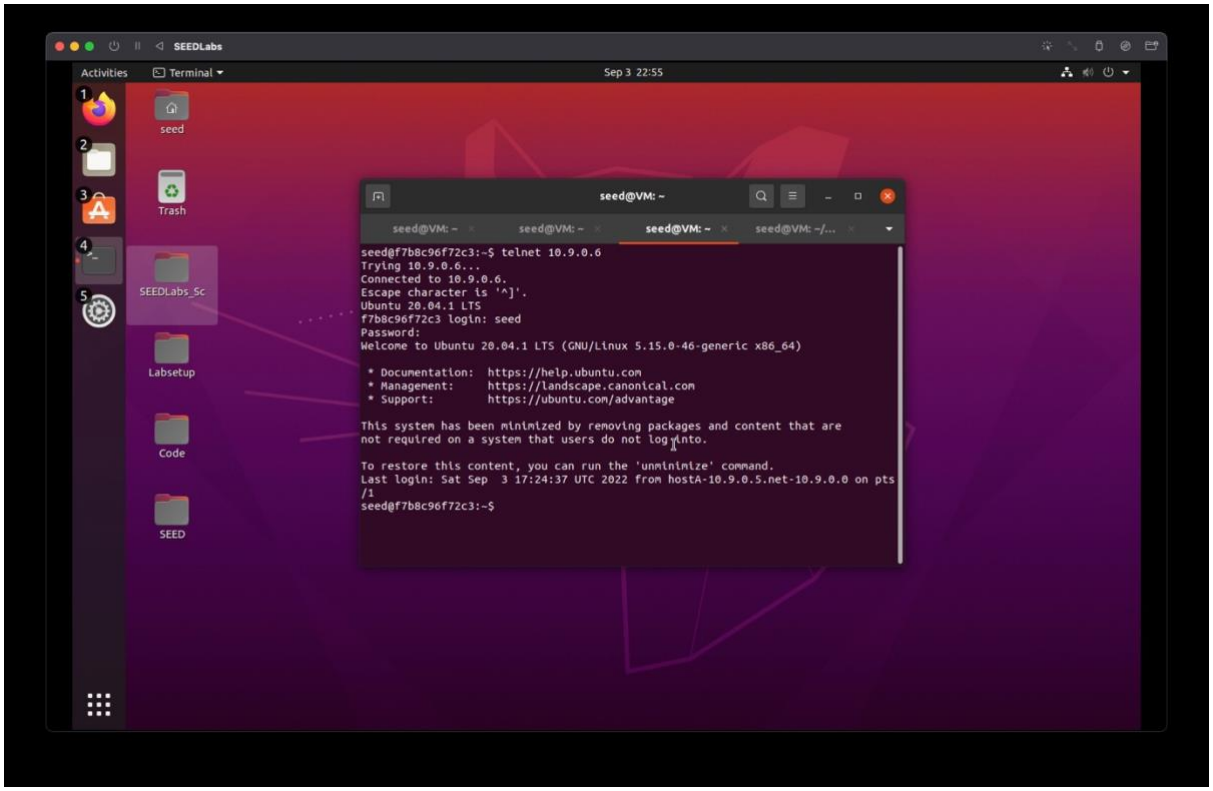
Provide screenshots of your observations.

Attacker:


SEEDLabs
Activities
Terminal
Sep 3 22:54
seed
Trash
SEEDLabs_Sc
Labsetup
Code
SEED
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed@VM: ~/...
root@VM:/volumes# ./sniff


SEEDLabs
Activities
Terminal
Sep 3 22:55
seed
Trash
SEEDLabs_Sc
Labsetup
Code
SEED
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed@VM: ~/...
To: 10.9.0.6
Protocol: TCP
From: 10.9.0.5
To: 10.9.0.6
Protocol: TCP
From: 10.9.0.5
To: 10.9.0.6
Protocol: TCP
From: 10.9.0.5
To: 10.9.0.6
Protocol: TCP
From: 10.9.0.5
To: 10.9.0.6
Protocol: TCP
From: 10.9.0.5
To: 10.9.0.6
Protocol: TCP
From: 10.9.0.5
To: 10.9.0.6
Protocol: TCP

Host A:



Attacker was able to sniff TCP packets from source:10.9.0.6 to destination:10.9.0.5 and vice versa

Task 2.1 C : Sniffing Passwords

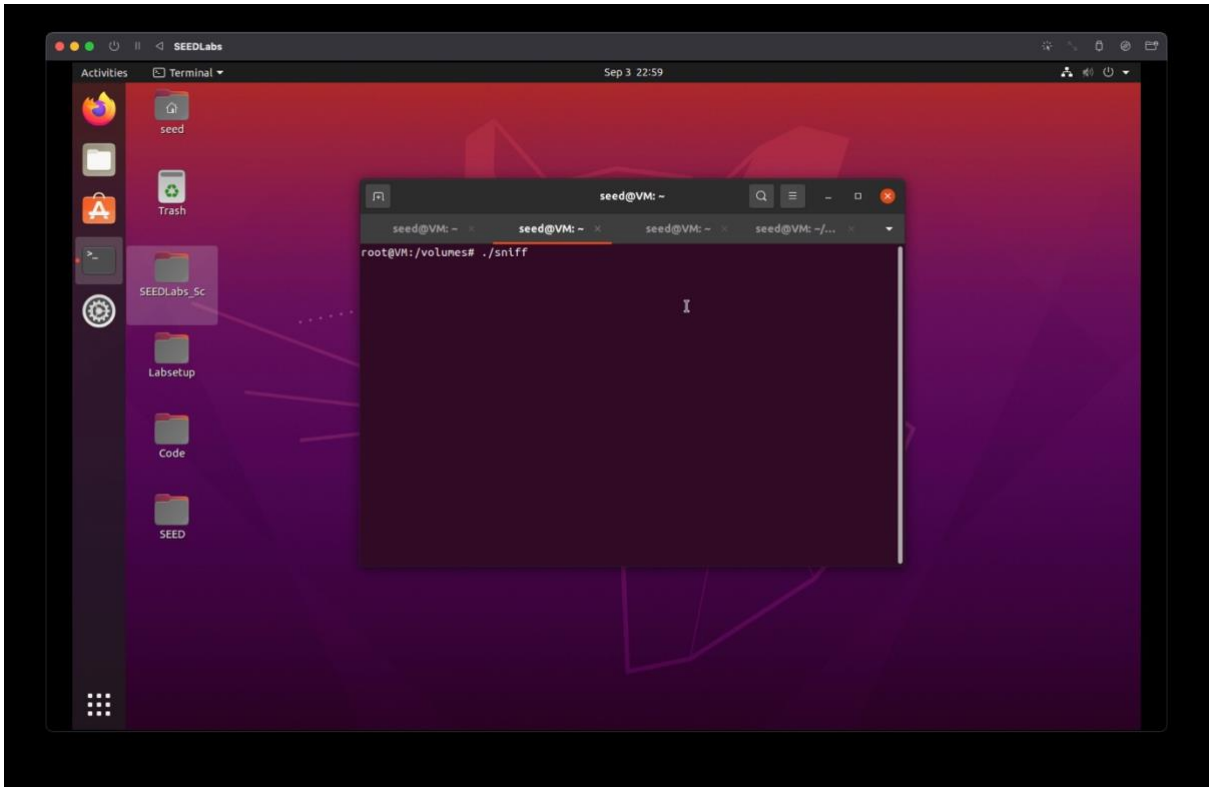
On the host VM :
 # gcc -o sniff Task2.1C.c -lpcap
 # docker cp sniff [Attacker machine docker container ID]:/volumes

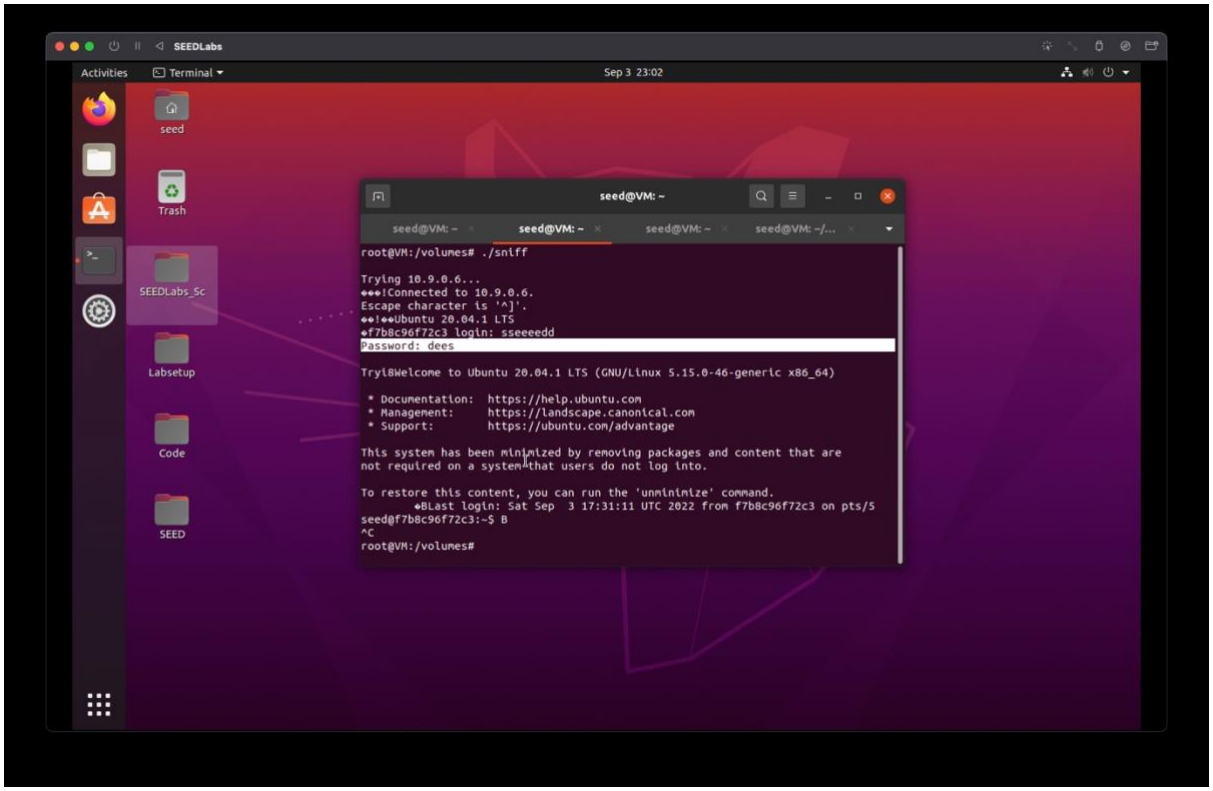
On the Attacker terminal run the command: # ./sniff

On Host A terminal :
 # telnet 10.9.0.6

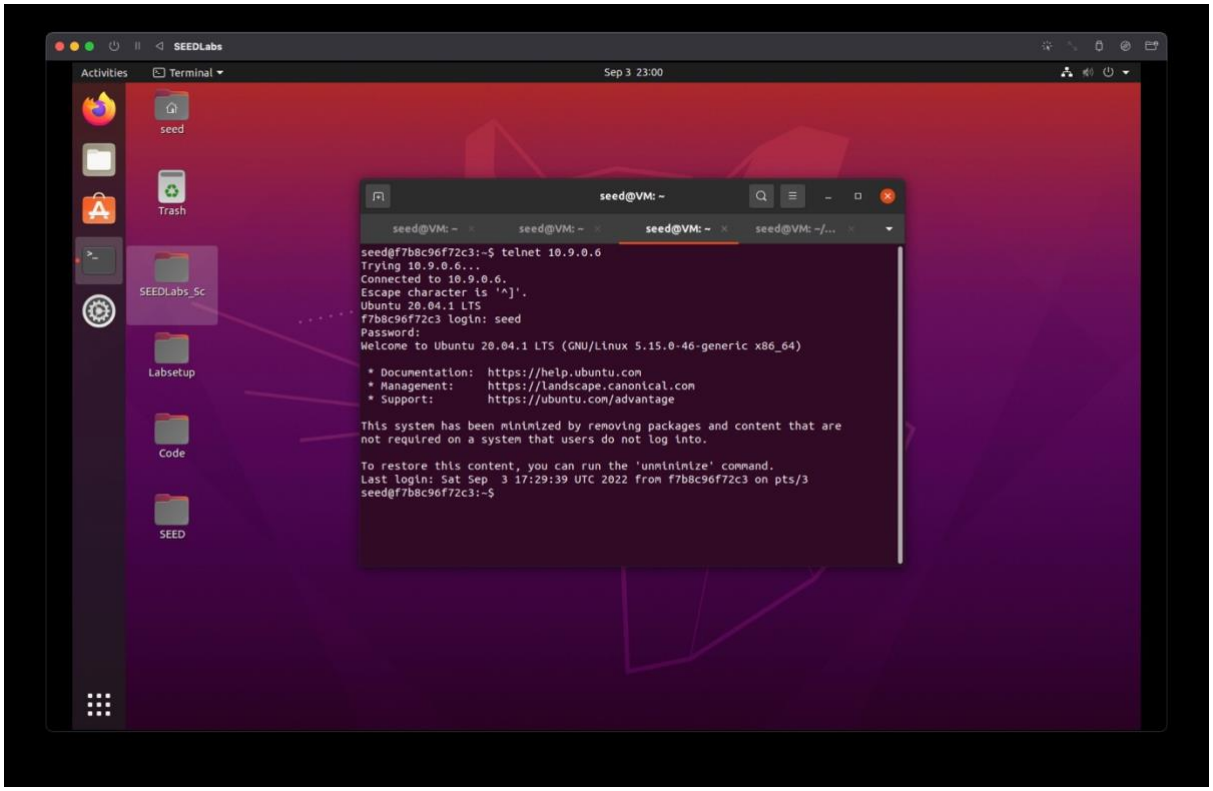
Provide screenshots of your observations.

Attacker:


SEEDLabs
Activities
Terminal
Sep 3 22:59
seed
Trash
SEEDLabs_Sc
Labsetup
Code
SEED
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed@VM: ~/...
root@VM:/volumes# ./sniff


SEEDLabs
Activities
Terminal
Sep 3 23:02
seed
Trash
SEEDLabs_Sc
Labsetup
Code
SEED
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed@VM: ~
seed@VM: ~/...
root@VM:/volumes# ./sniff
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
f7b8c96f72c3 login: sseeeedd
Password: dees
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-46-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.
To restore this content, you can run the 'unminimize' command.
Last login: Sat Sep 3 17:31:11 UTC 2022 from f7b8c96f72c3 on pts/5
seed@f7b8c96f72c3:~$ B
^C
root@VM:/volumes#

Host A:



Observation: The attacker was successful in capturing the password to be ‘dees’

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

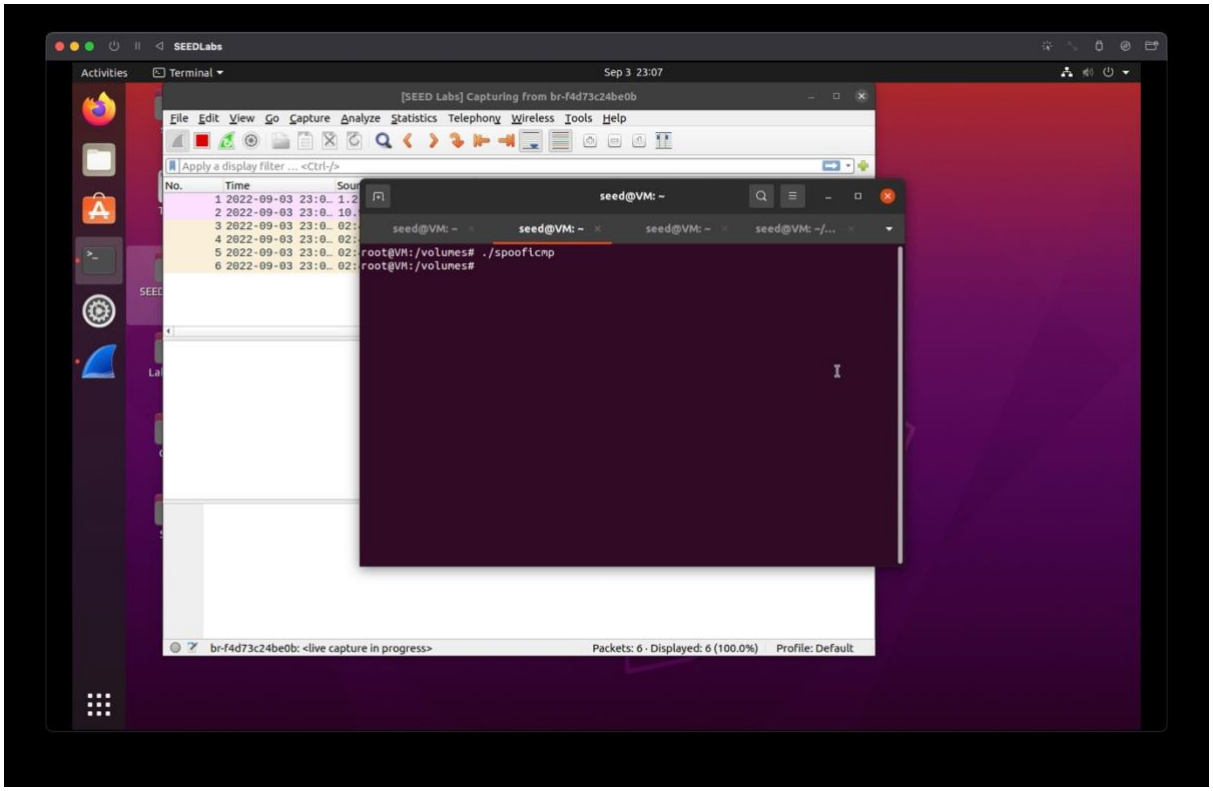
Open wireshark before executing the program and select the same interface in wireshark, as used in the code for each task ie. the attacker machines interface.c

On the host VM :
 # gcc -o spooficmp Task2.2.c -lpcap
 # docker cp spooficmp [Attacker machine docker container ID]:/volumes

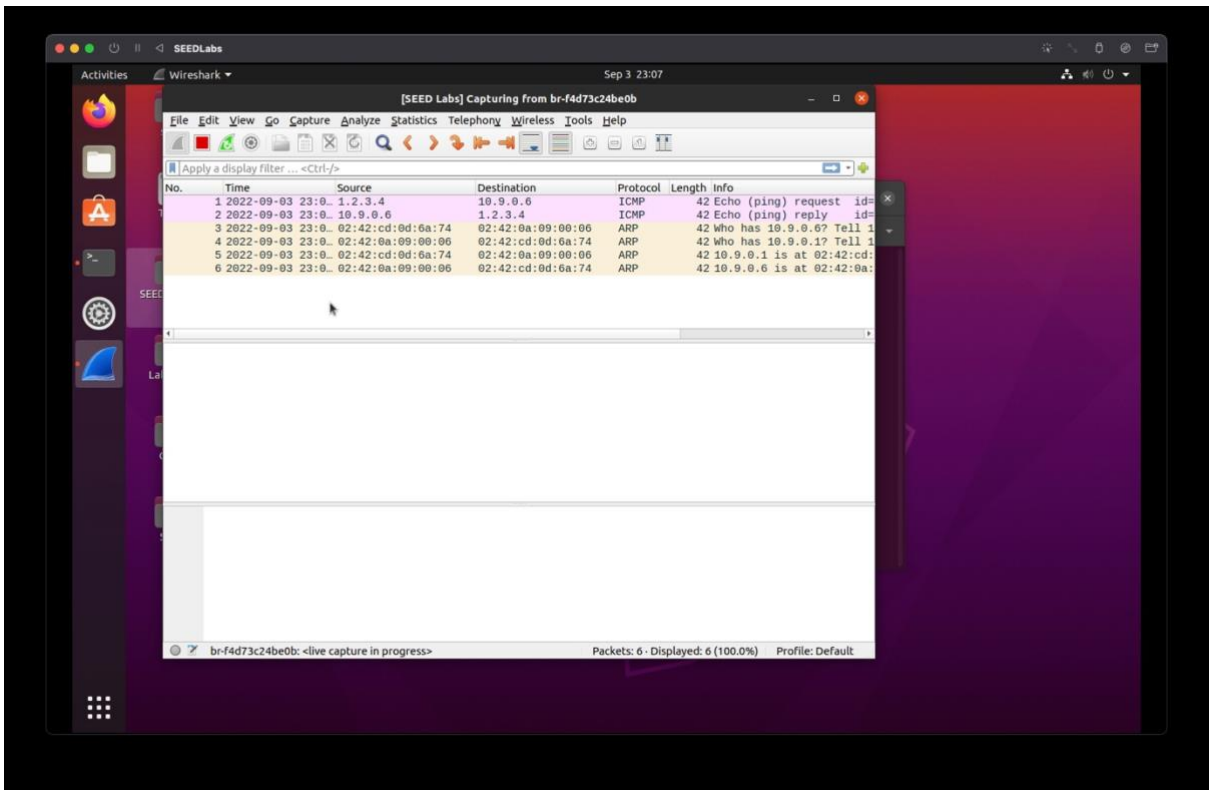
On Attacker Machine terminal : **# ./spooficmp**

Provide screenshots of your observations.

Attacker:



Wireshark:

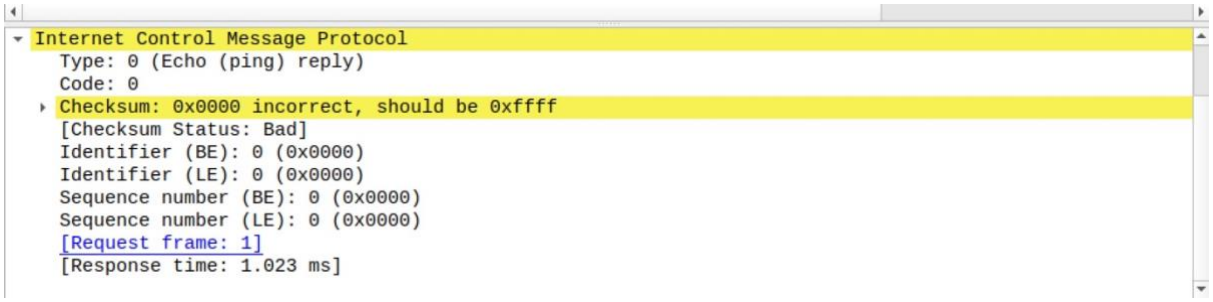


Observation: It appears that attacker spoofed the packet to be from source 1.2.3.4, but the checksum of the packet is incorrect

Please answer the following questions.

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

➔ Yes we need to calculate the checksum for the IP header, because the address of source has changed, and this will result in incorrect checksum, and this makes the victim aware of the attack



● **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?

➔ Since creating a raw socket requires the system to access low level the permission of which is not given to a normal user
➔ Root privilege makes sure that no networking rules are broken

Task 2.3 Sniff and then Spoof

Open wireshark before executing the program and select the same interface in wireshark, as used in the code for each task ie. the attacker machines interface.

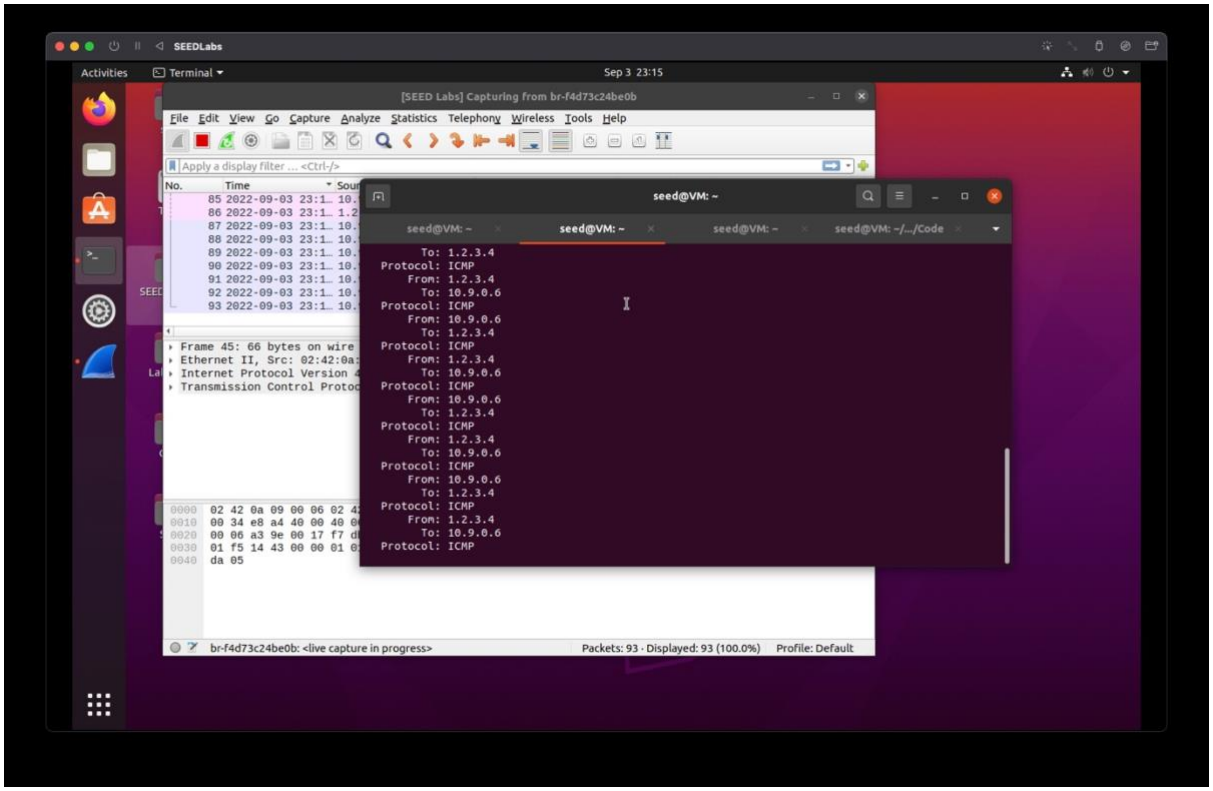
On the host VM :
 # gcc -o sniffspoof Task2.3.c -lpcap
 # docker cp sniffspoof [Attacker machine docker container ID]:/volumes

On Attacker Machine terminal : # ./sniffspoof

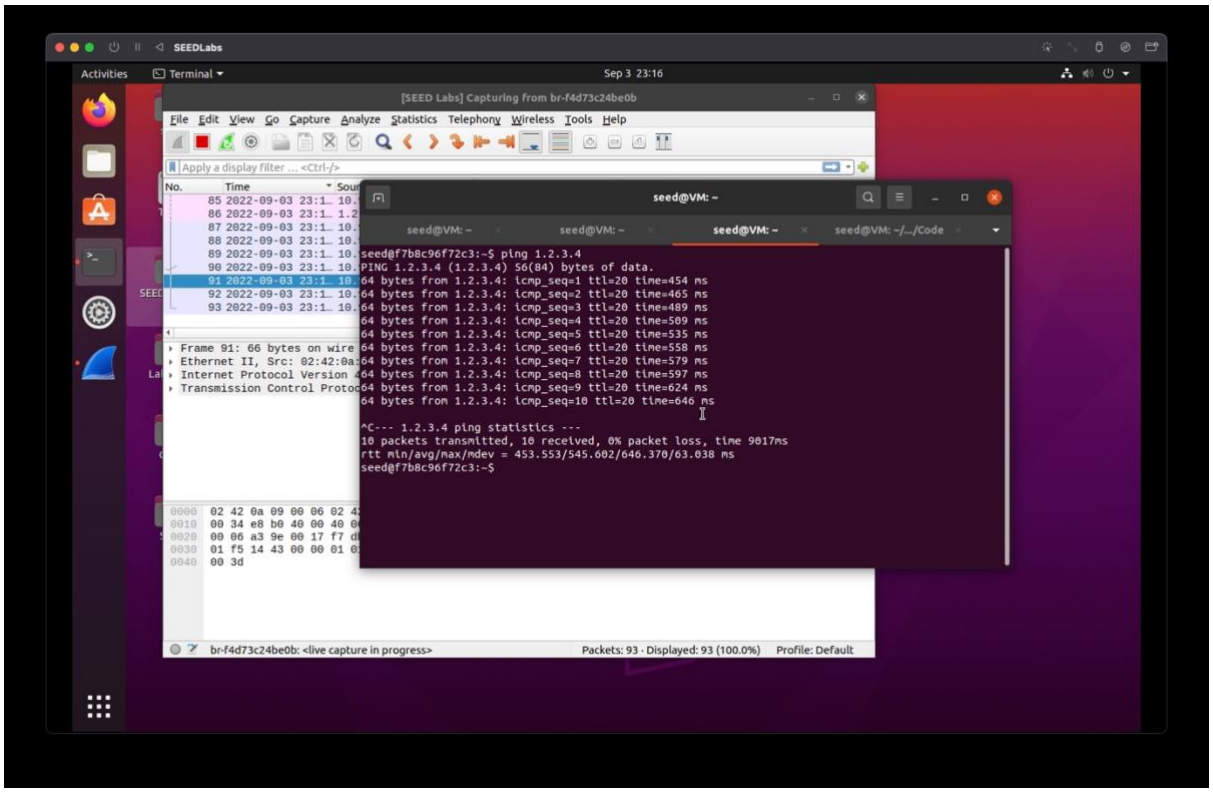
On the Host A terminal ping 1.2.3.4 # ping 1.2.3.4

Provide screenshots of your observations.

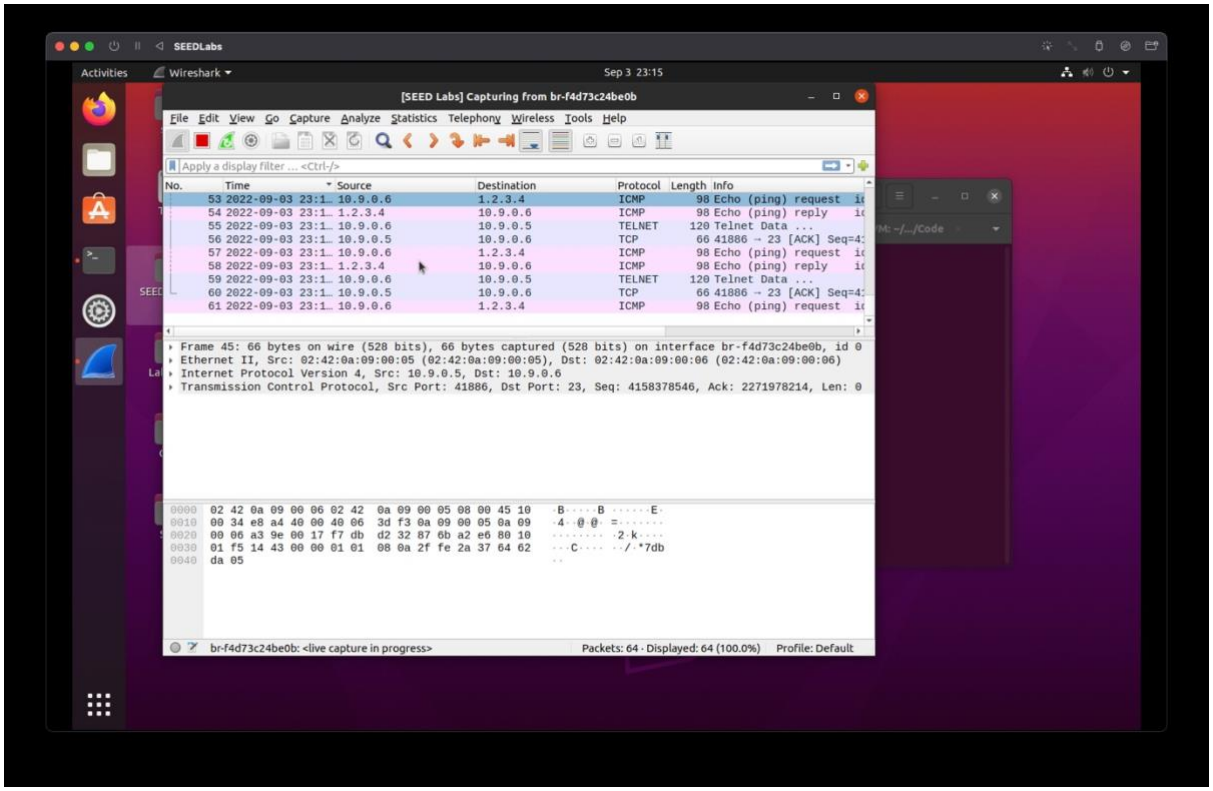
Attacker:



Host A:



Wireshark:



Observation: The attacker is able to create echo response for a non existent IP address (1.2.3.4)