|  |  |
| --- | --- |
| 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

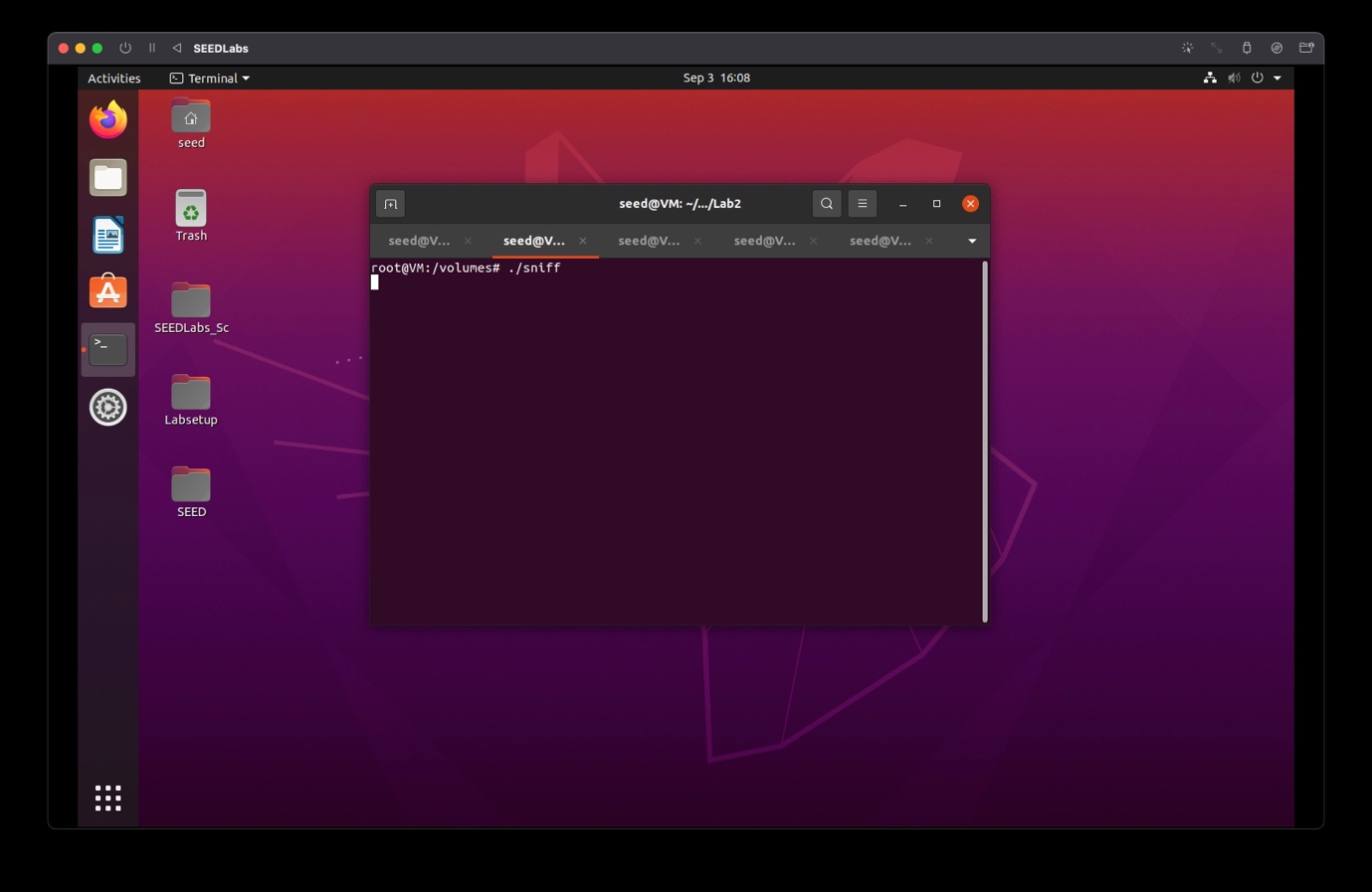
**On the host VM** :  
**# gcc -o sniff Task2.1A.c -lpcap  
# docker cp sniff [Attacker machine docker container ID]:/volumes**

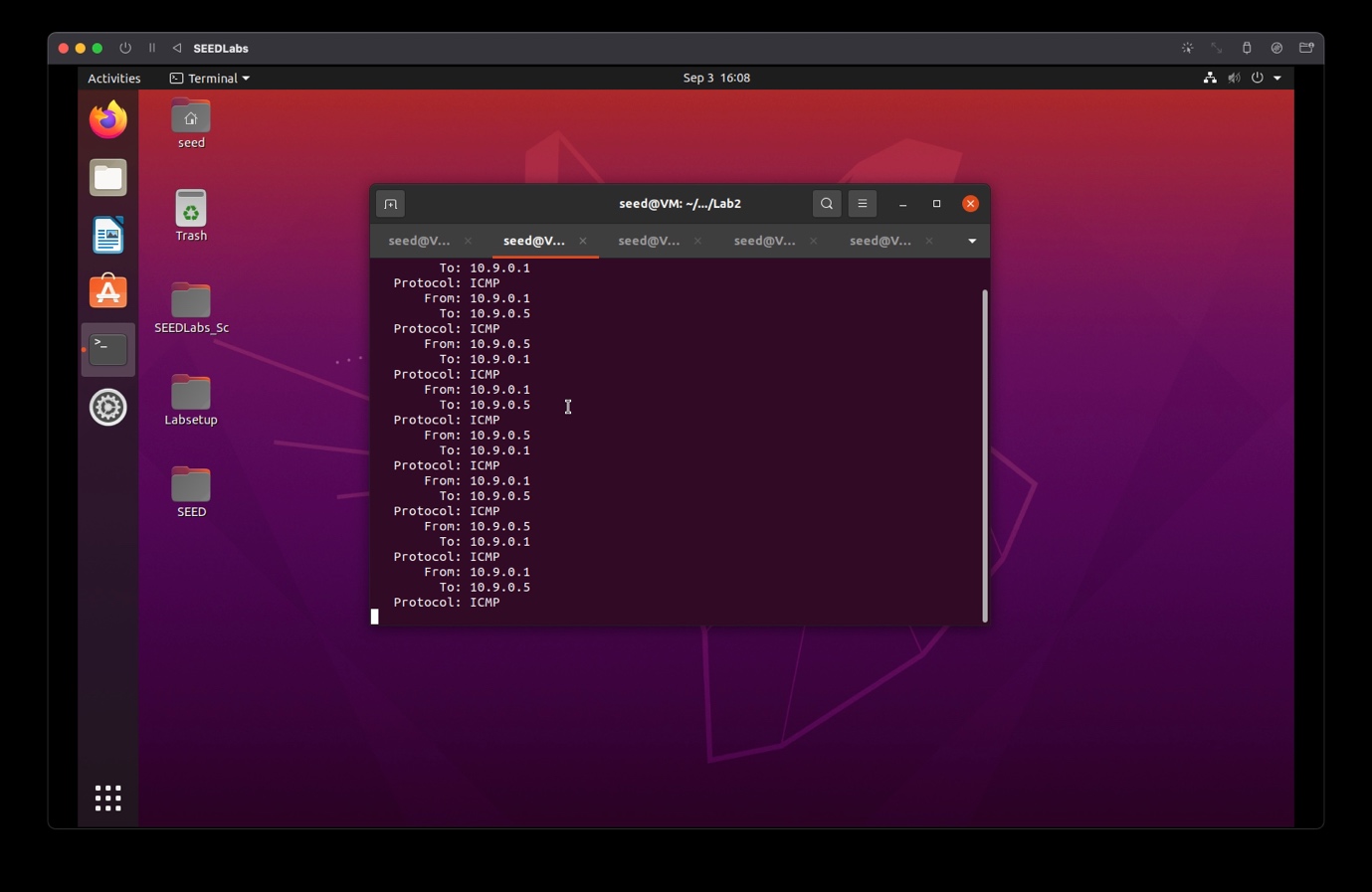
**On the Attacker container run the command: # ./sniff**

**On Host A terminal : # ping 10.9.0.1**

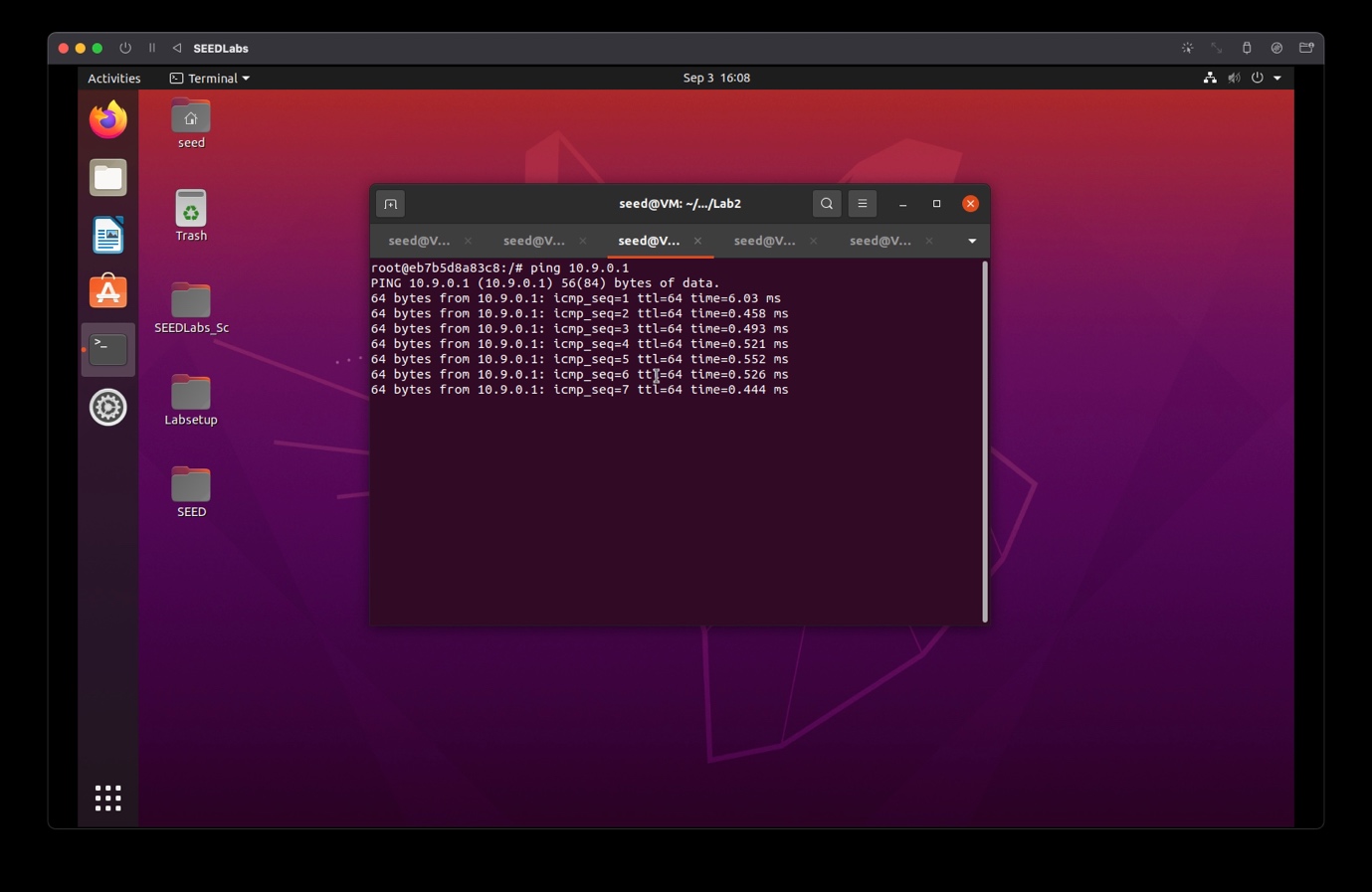
Provide screenshots of your observations.

**Attacker:**

****

****

**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

1. pcap\_openlive
2. pcap\_openlive
3. pcap\_setfilter
4. pcap\_setfilter
5. pcap\_setfiler
6. 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 :**

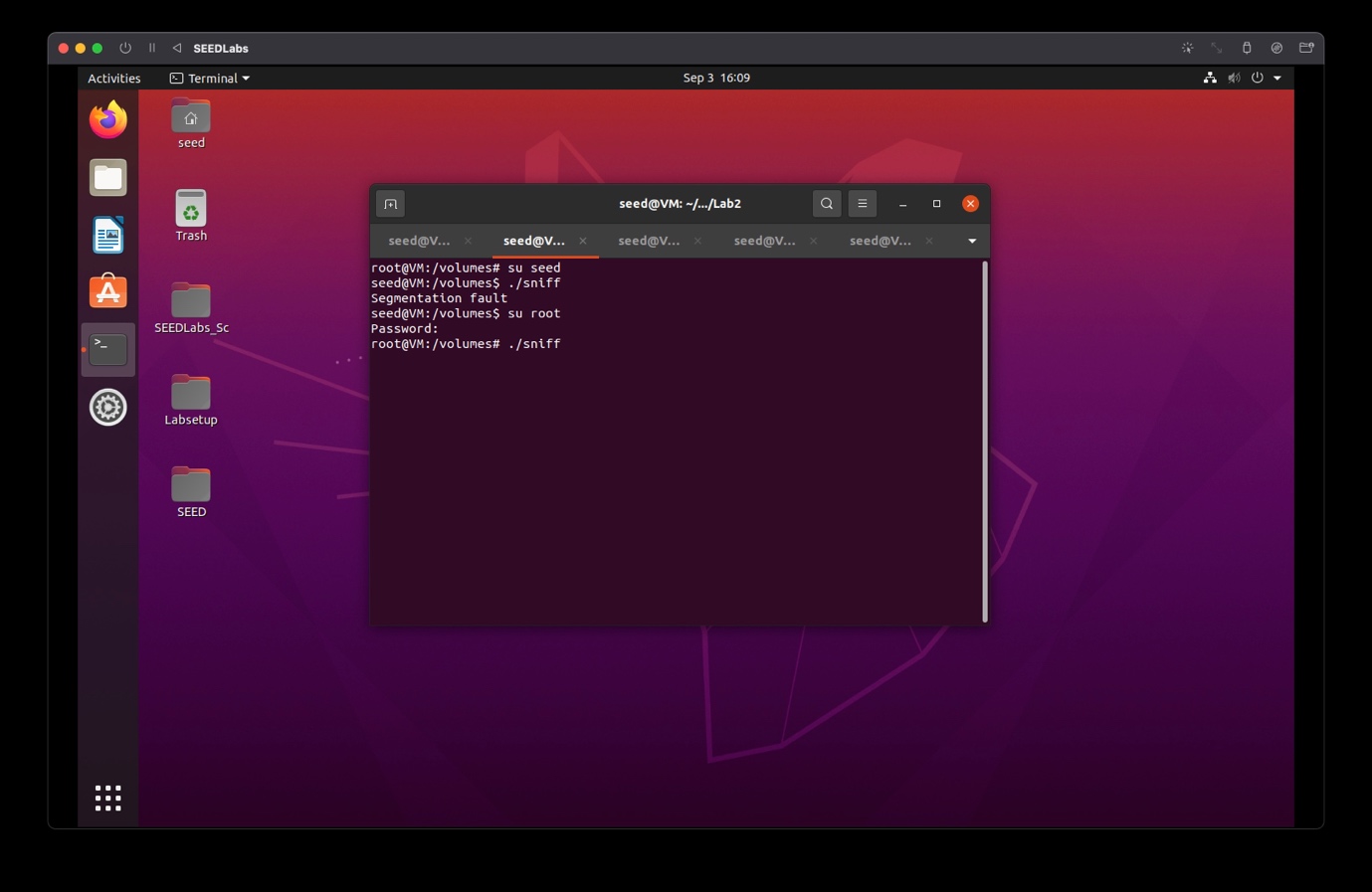
**# su seed**

**# ./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);**

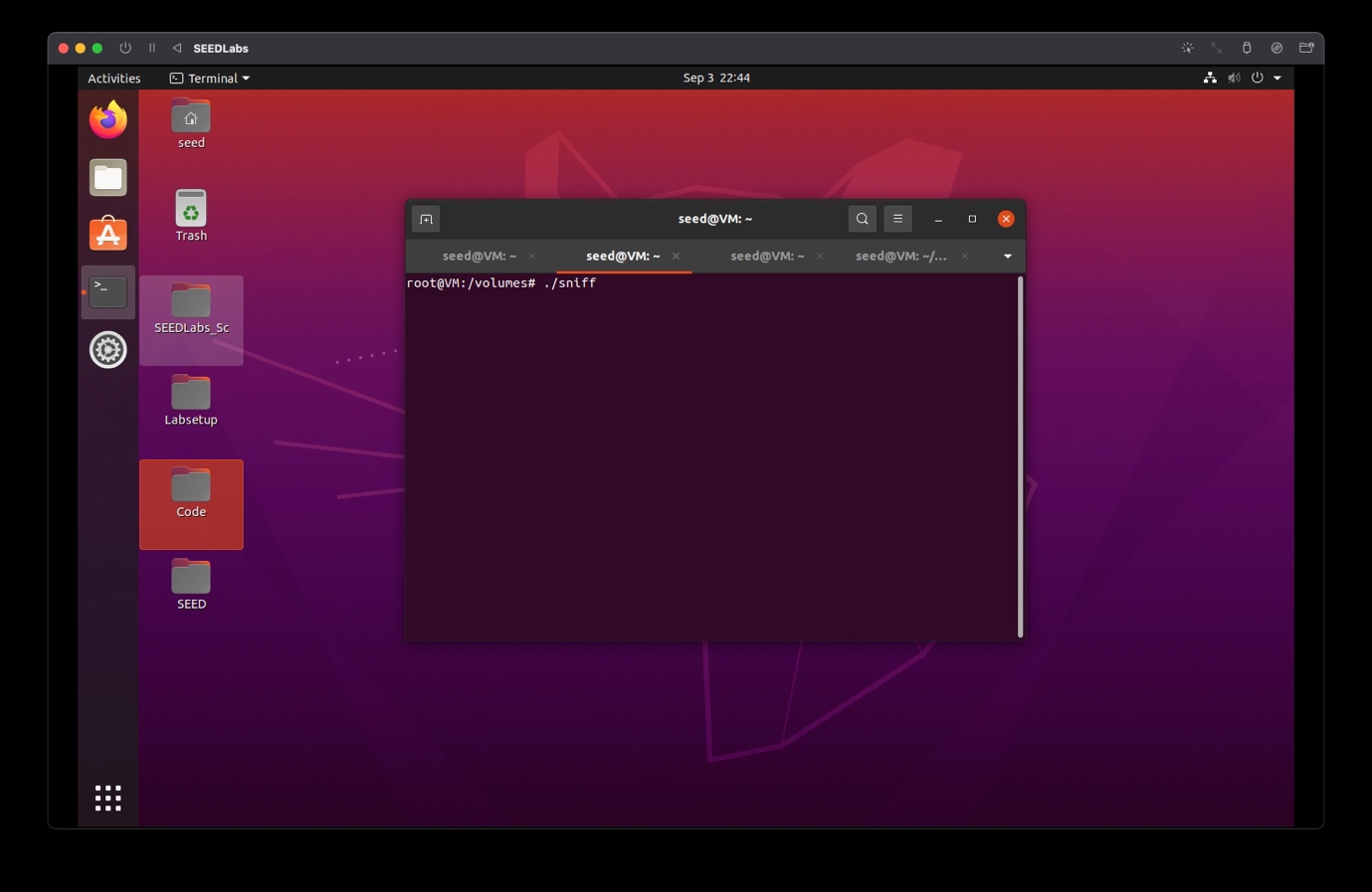
**On the host VM :  
# gcc -o sniff Task2.1A.c -lpcap  
# docker cp sniff [Attacker machine docker container ID]:/volumes**

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

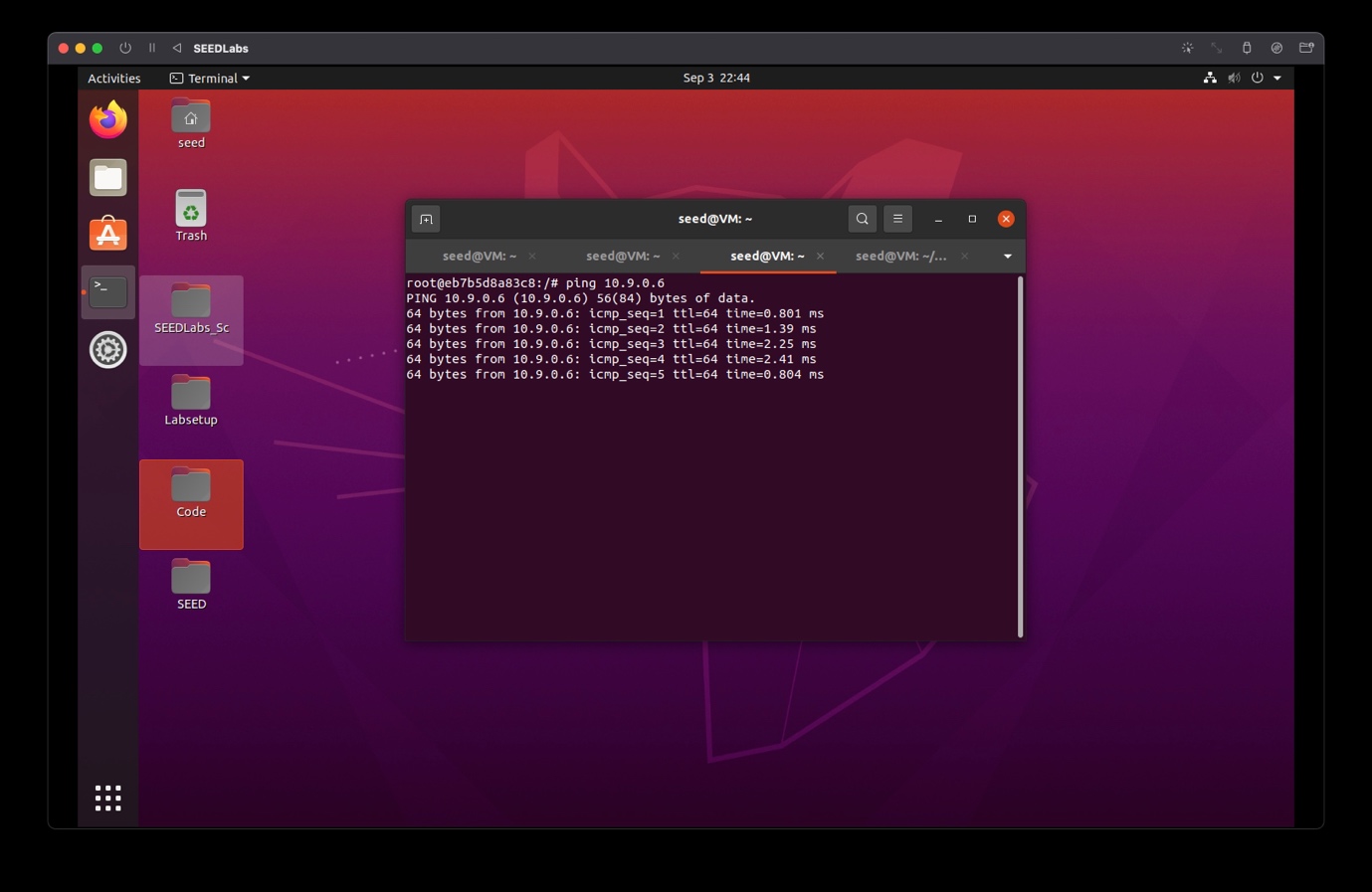
**On Host A terminal : # ping 10.9.0.6**

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:**

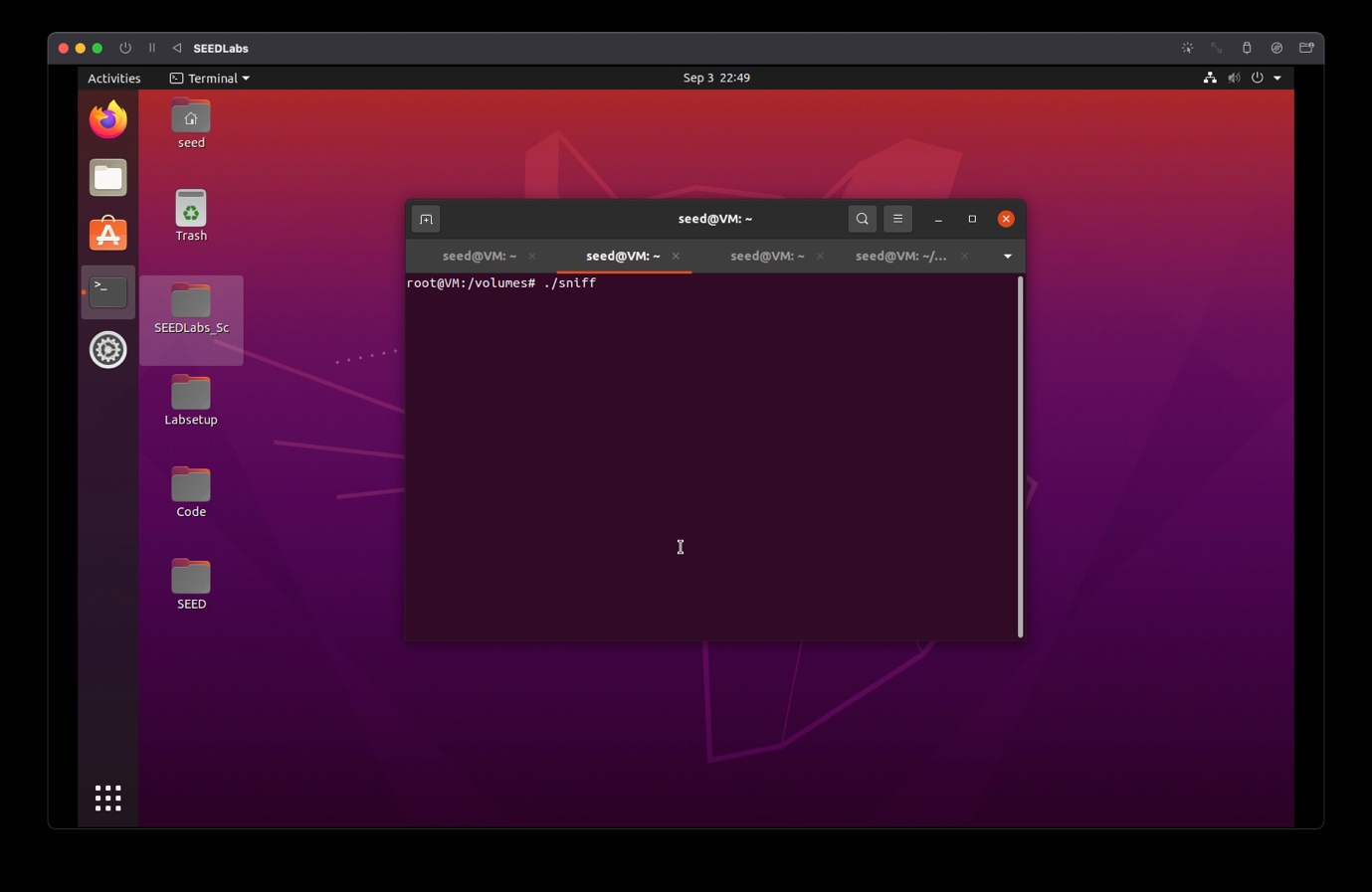
**# ./sniff**

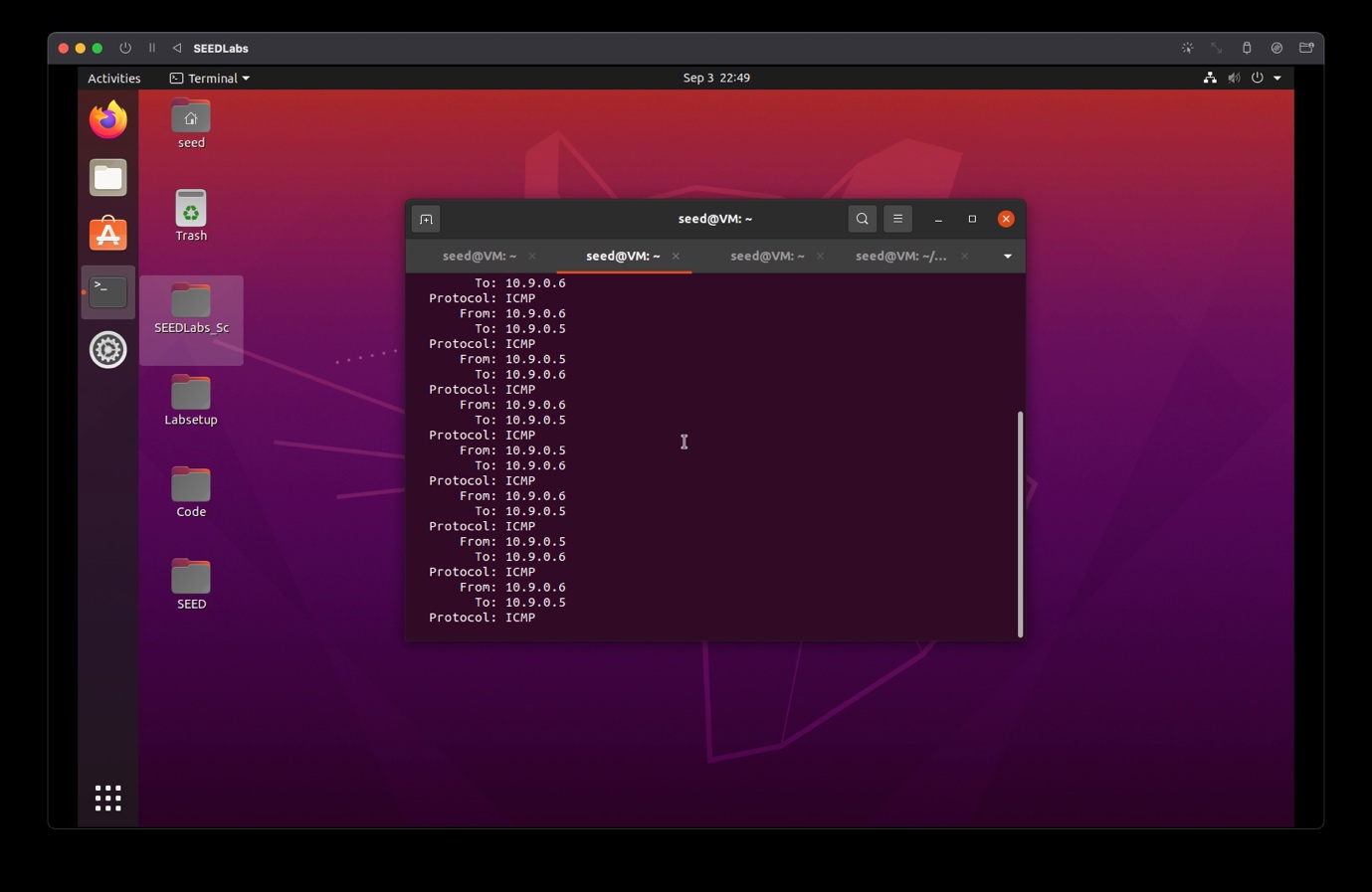
**In the host A machine ping any ip address**

**# ping 10.9.0.6**

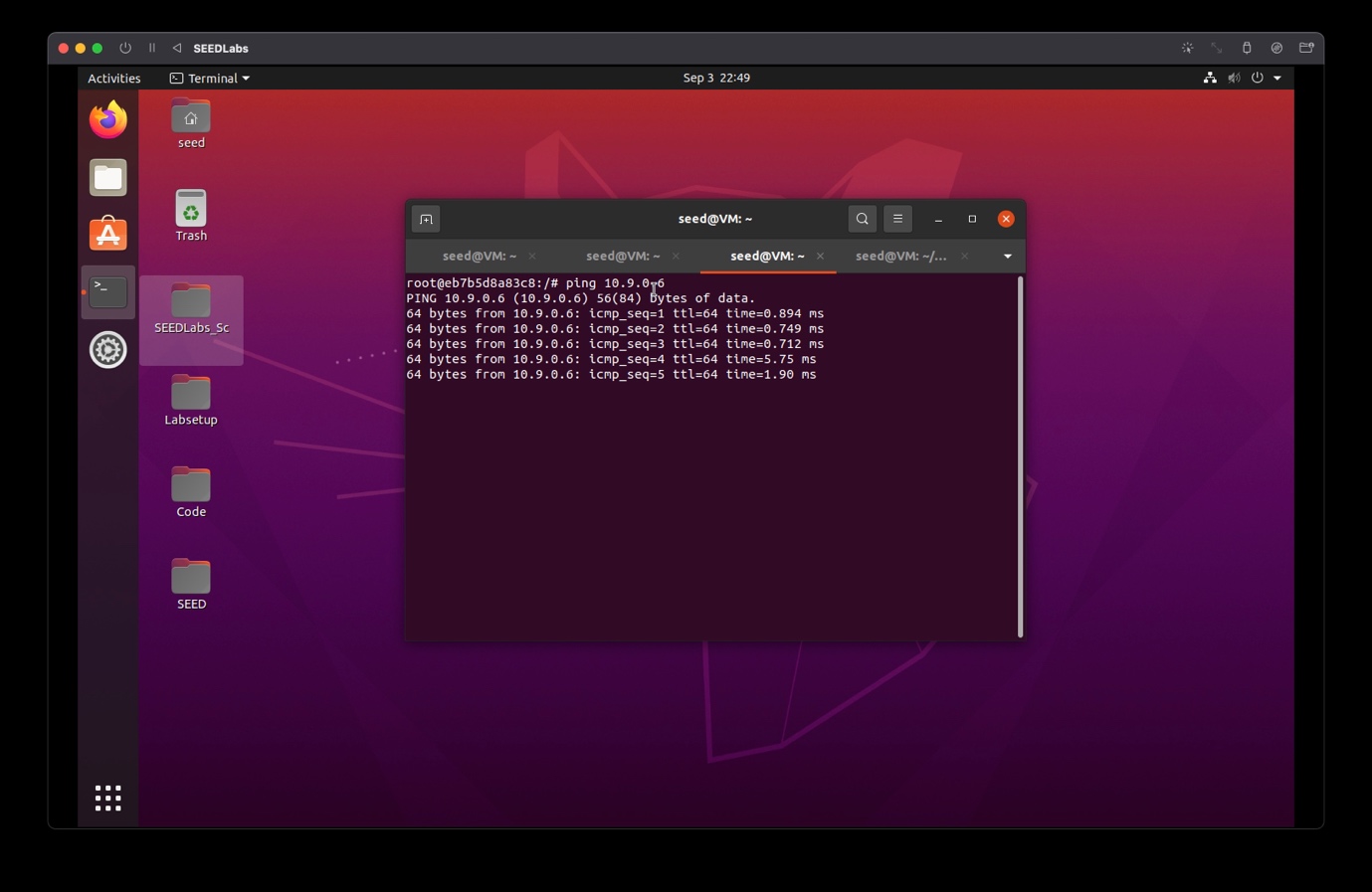
Provide screenshots of your observations.

Attacker:





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.**

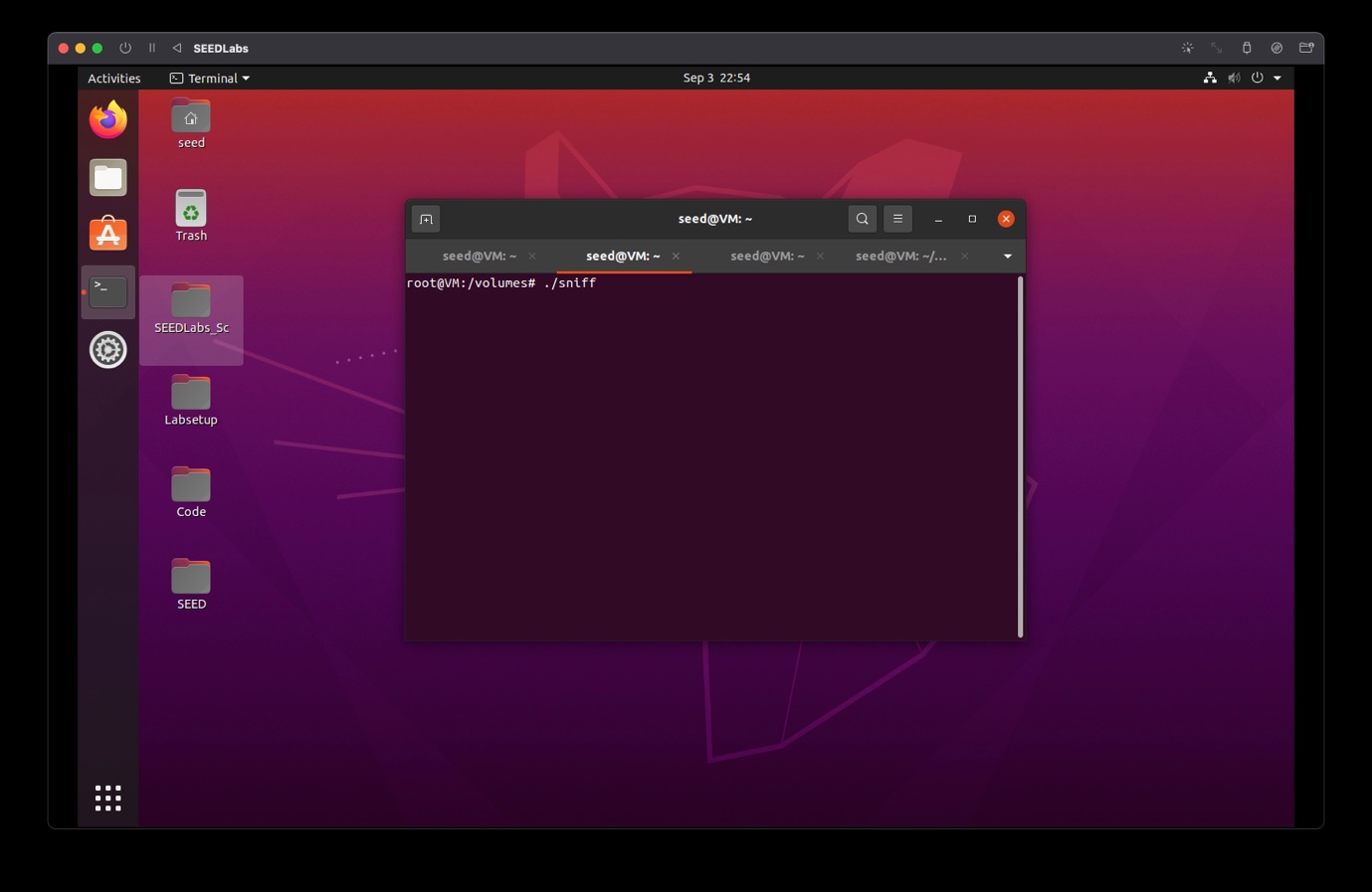
**On the host VM** :  
**# gcc -o sniff Task2.1B-TCP.c -lpcap  
# docker cp sniff [Attacker machine docker container ID]:/volumes**

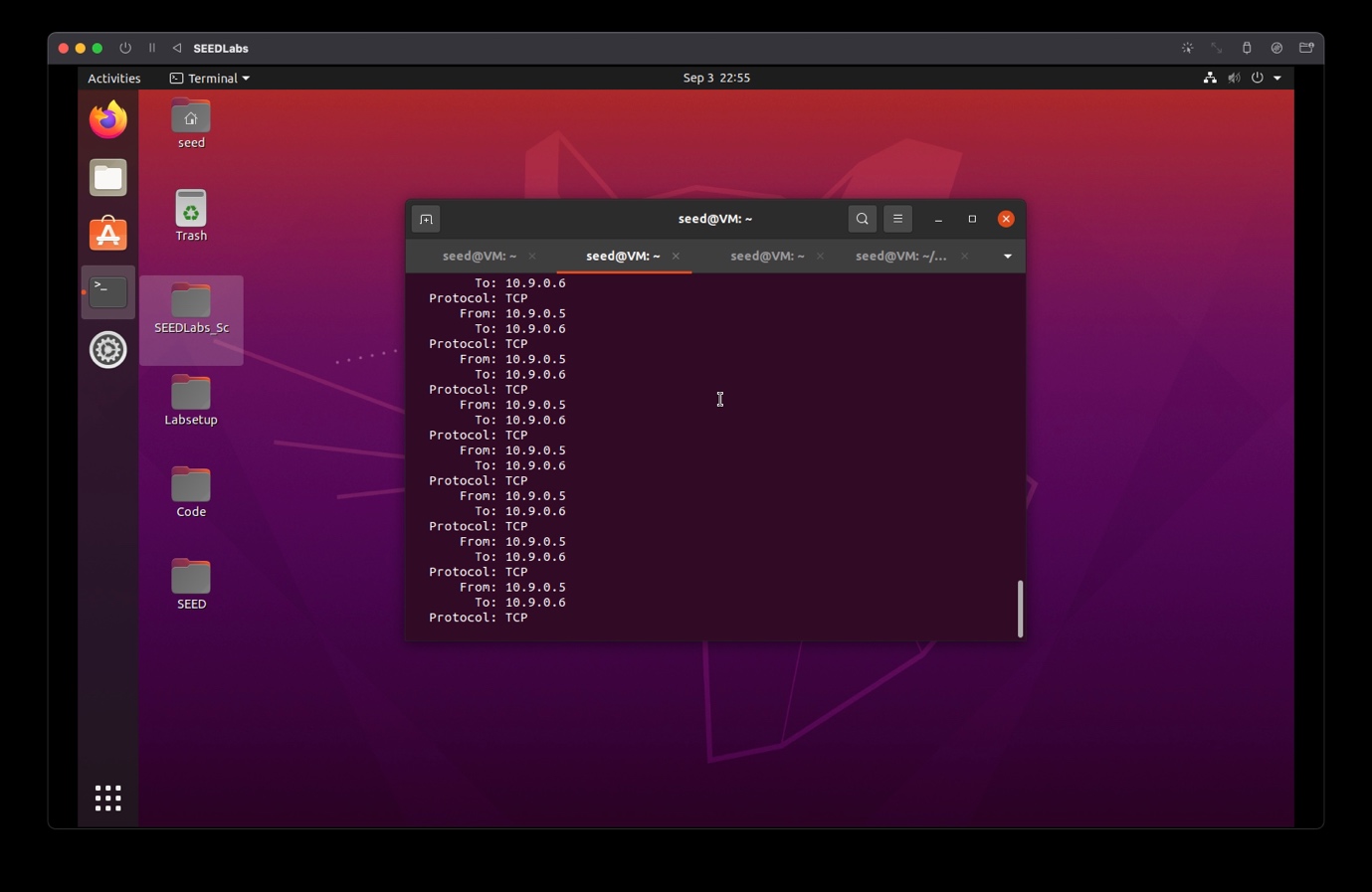
**On Attacker Machine terminal : # ./sniff**

**On Host A terminal :  
# telnet 10.9.0.6**

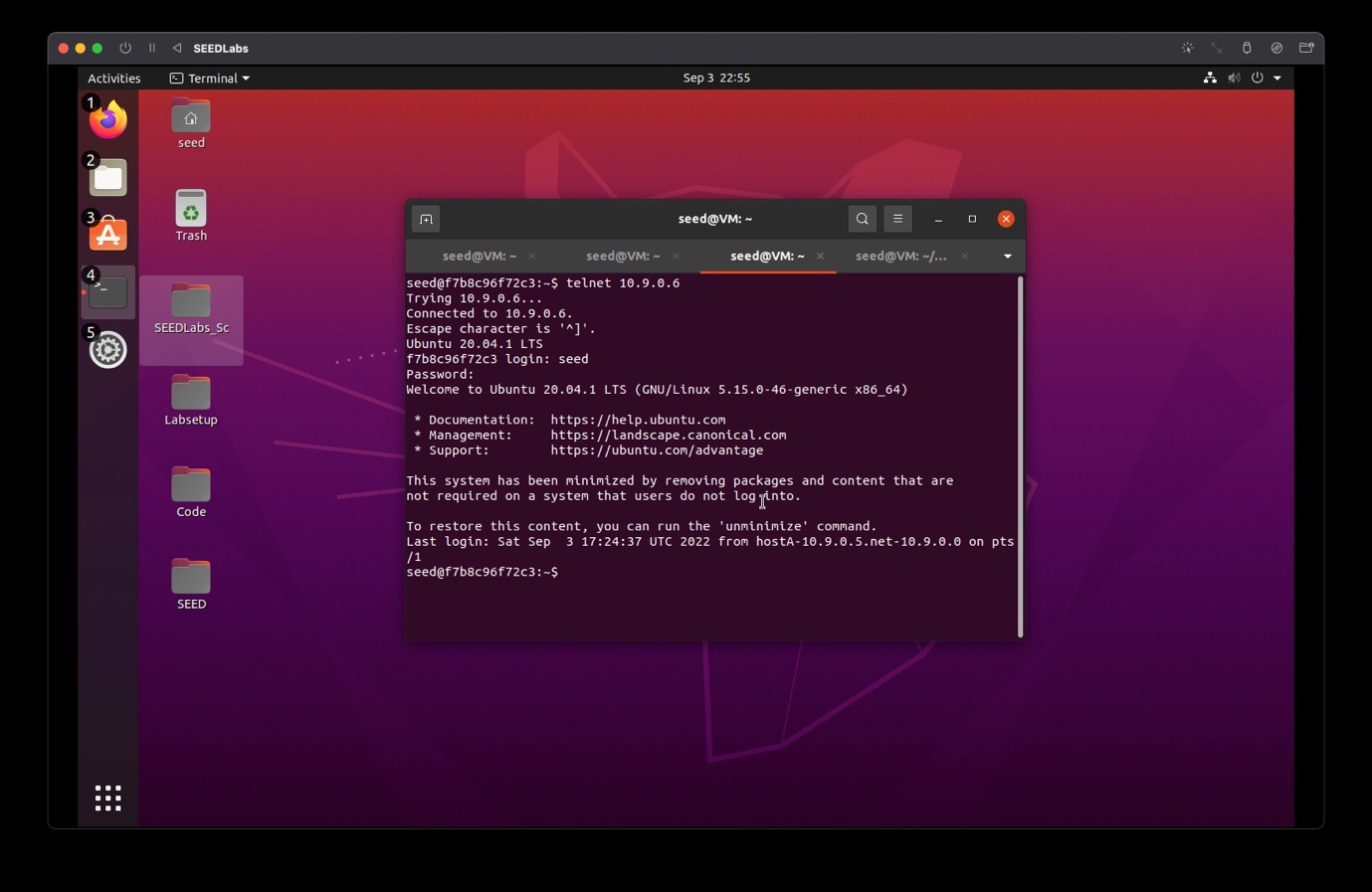
Provide screenshots of your observations.

Attacker:





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**

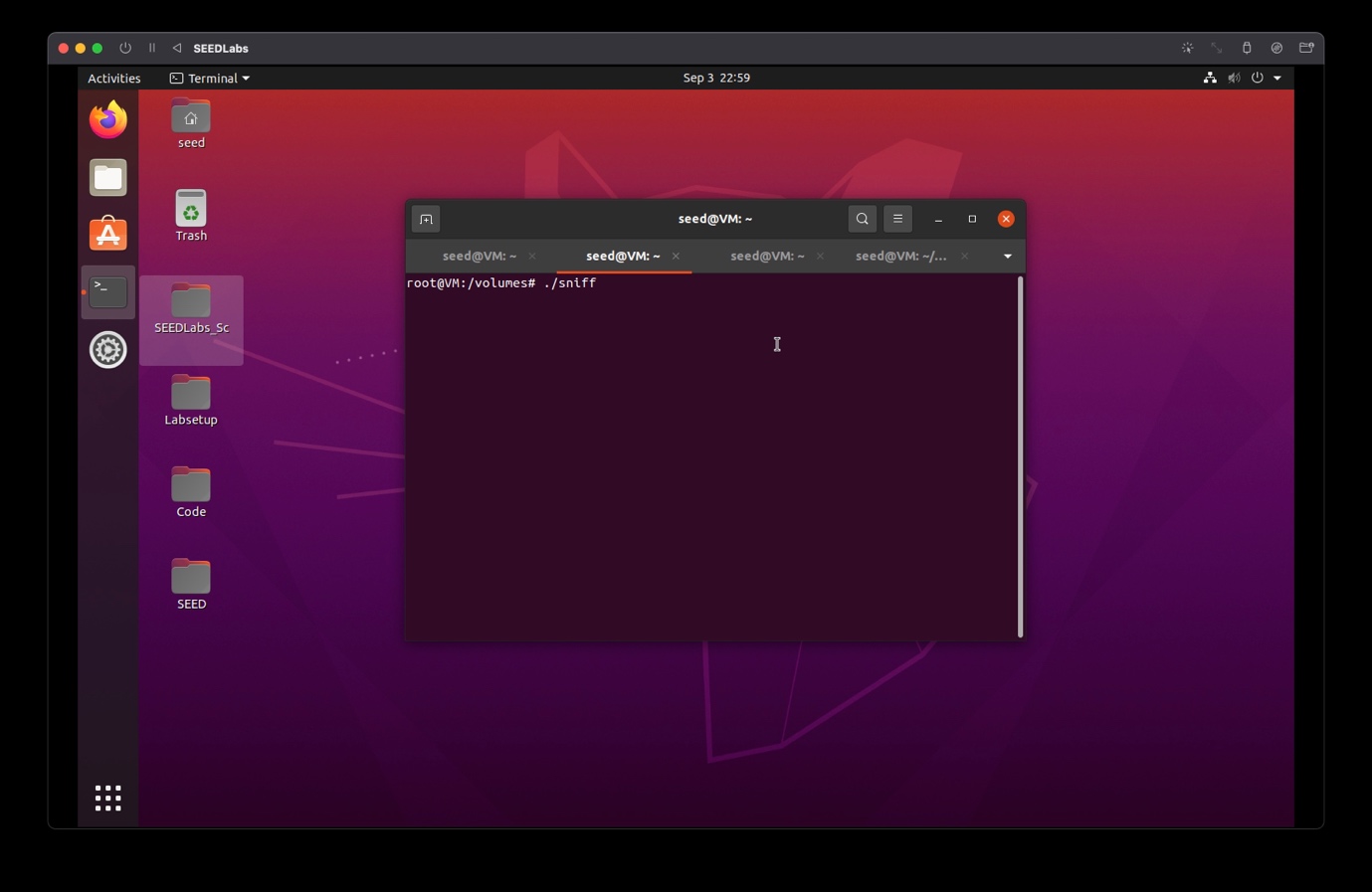
**On the host VM** :  
**# gcc -o sniff Task2.1C.c -lpcap  
# docker cp sniff [Attacker machine docker container ID]:/volumes**

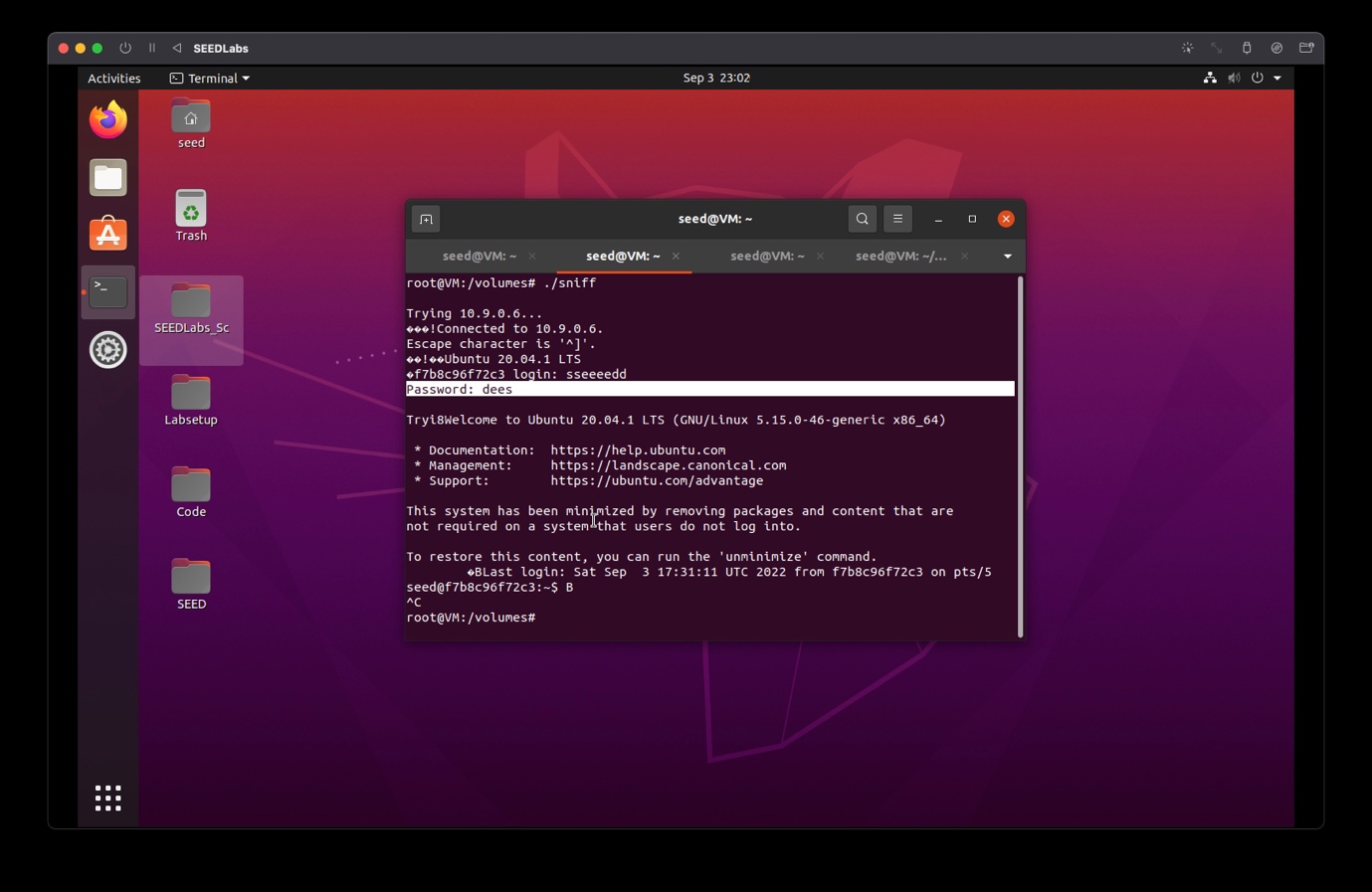
**On the Attacker terminal run the command: # ./sniff**

**On Host A terminal** :  
**# telnet 10.9.0.6**

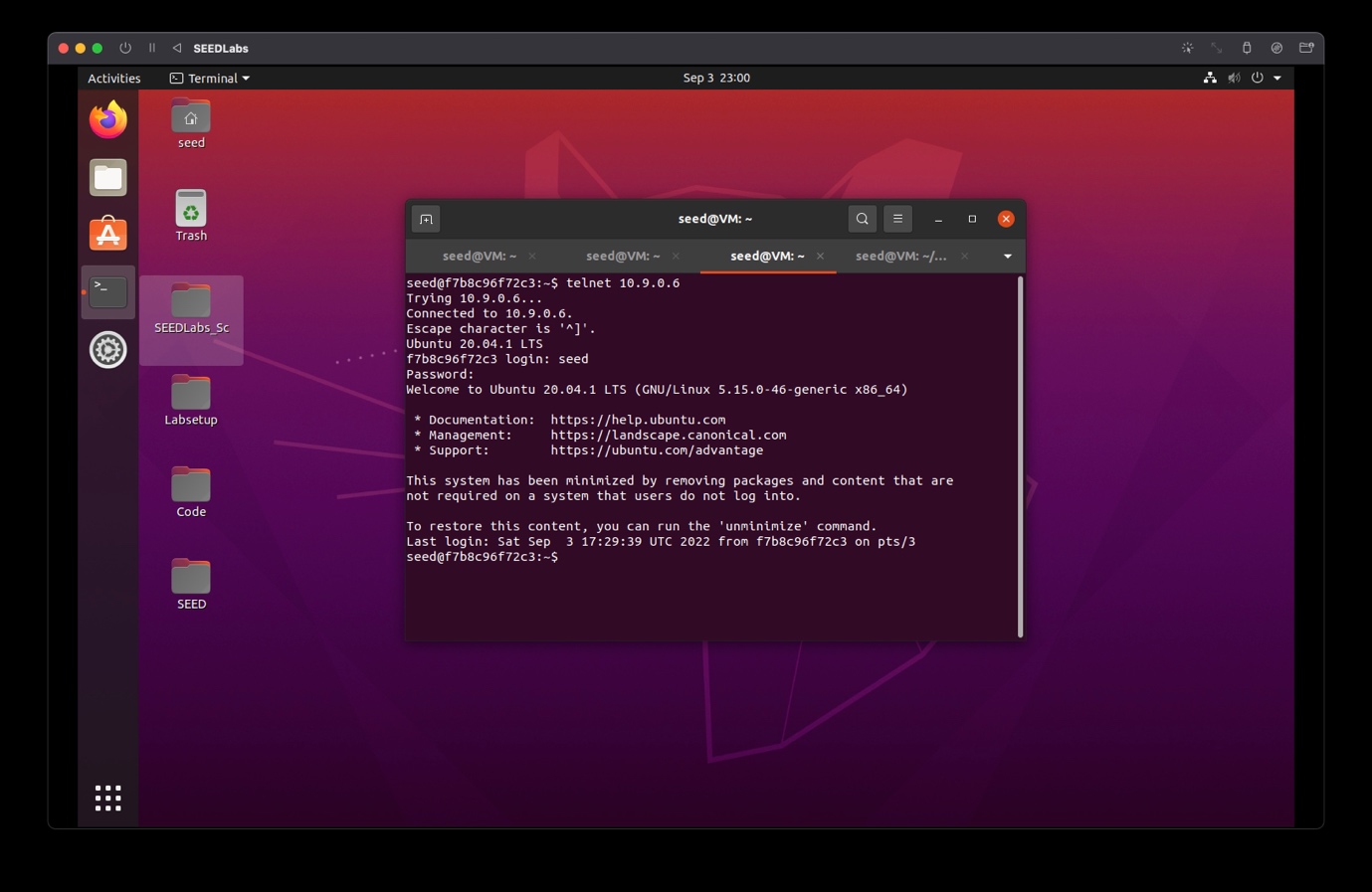
Provide screenshots of your observations.

Attacker:





Host A:



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

page8image22556864

**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**

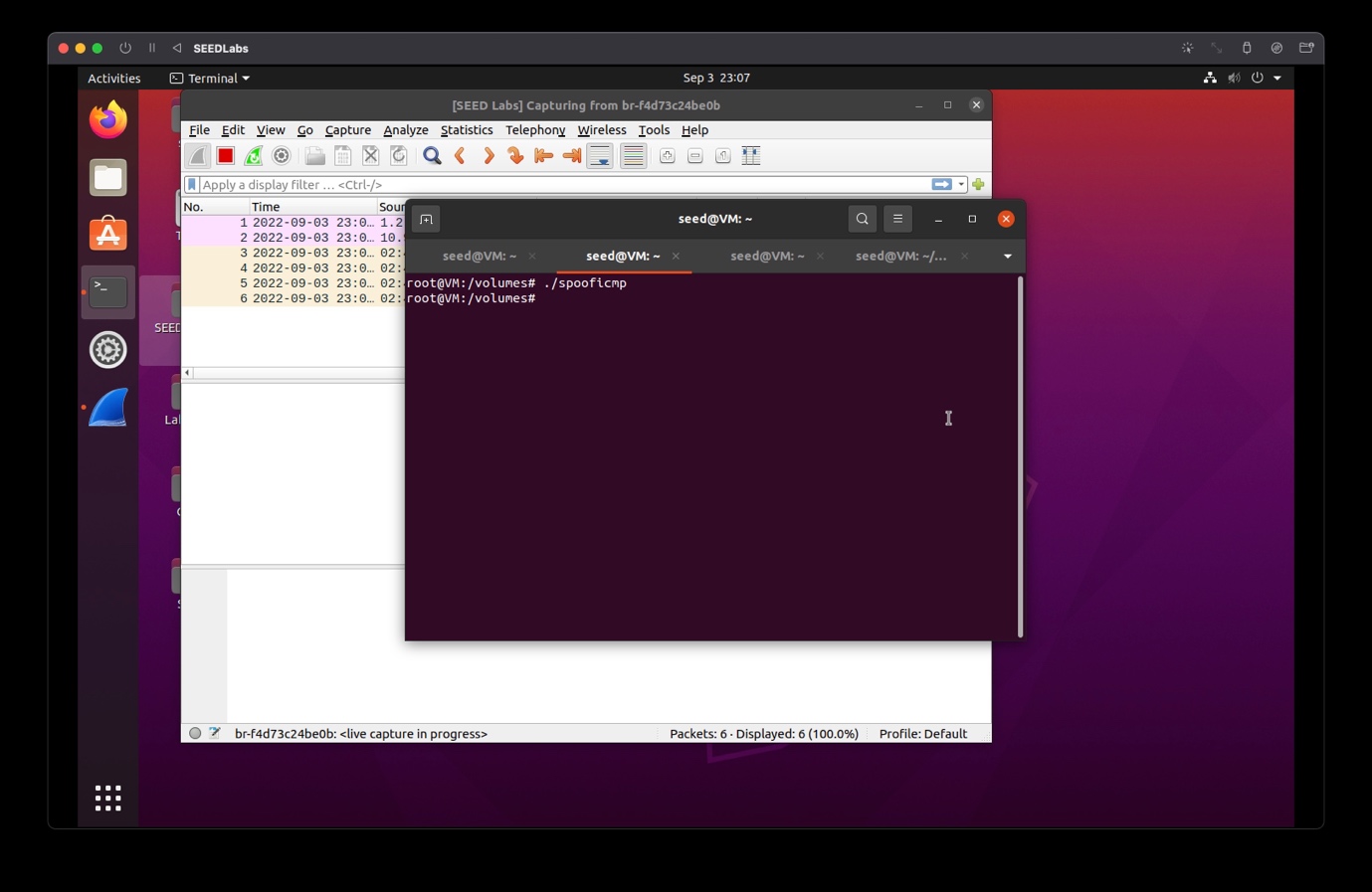
Open wireshark before executing the program and select the same interface in wireshark, as used in the code for each task ie. the attacker machines interface.c

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

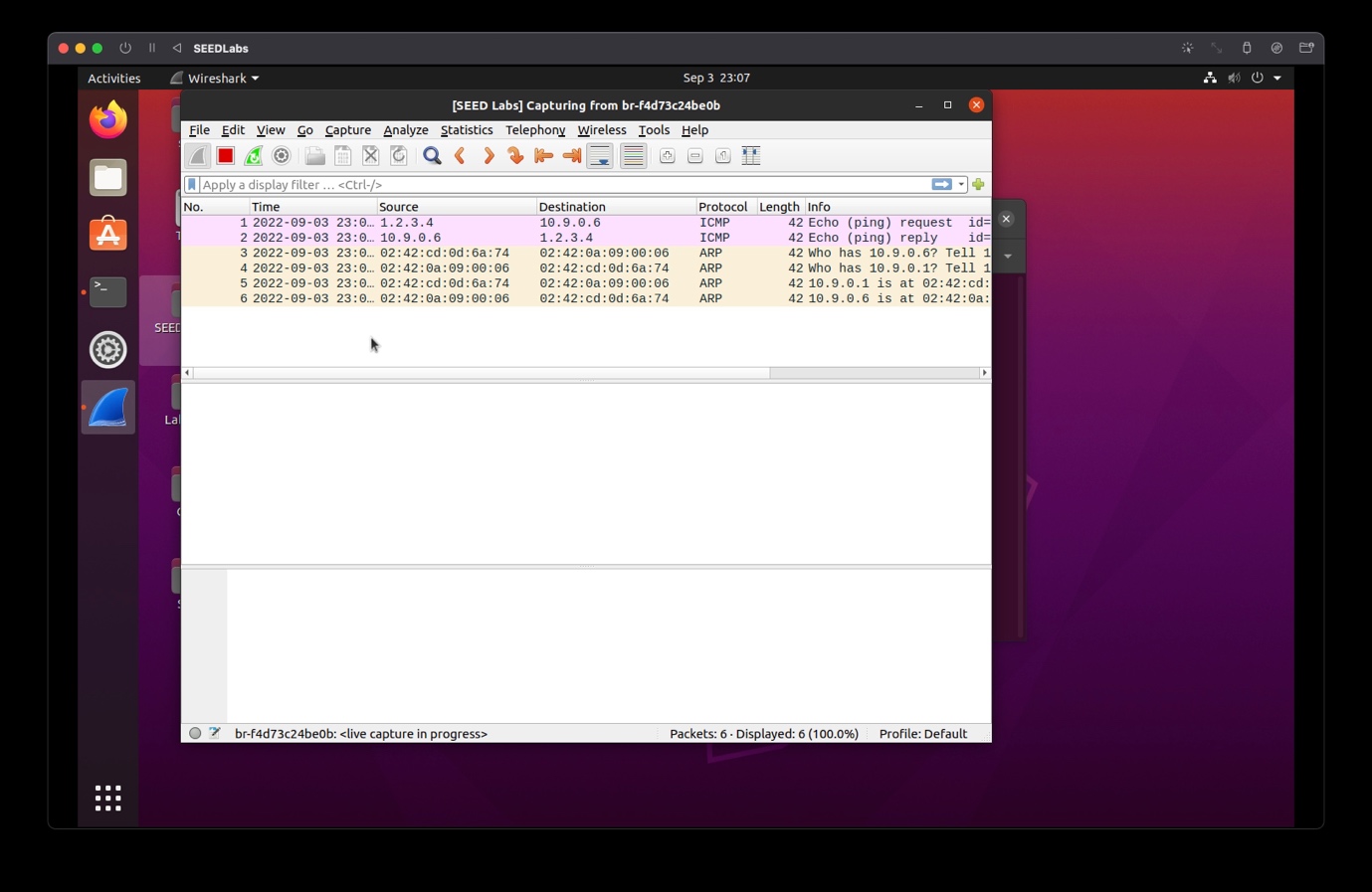
**On Attacker Machine terminal** : **# ./spooficmp**

Provide screenshots of your observations.

Attacker:



Wireshark:

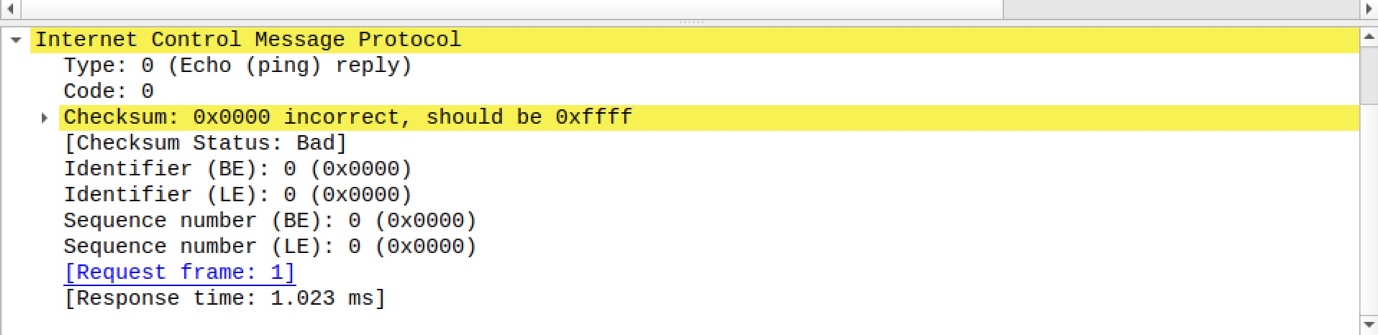


Observation: It appears that attacker spoofed the packet to be from source 1.2.3.4, but the checksum of the packet is incorrect

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.

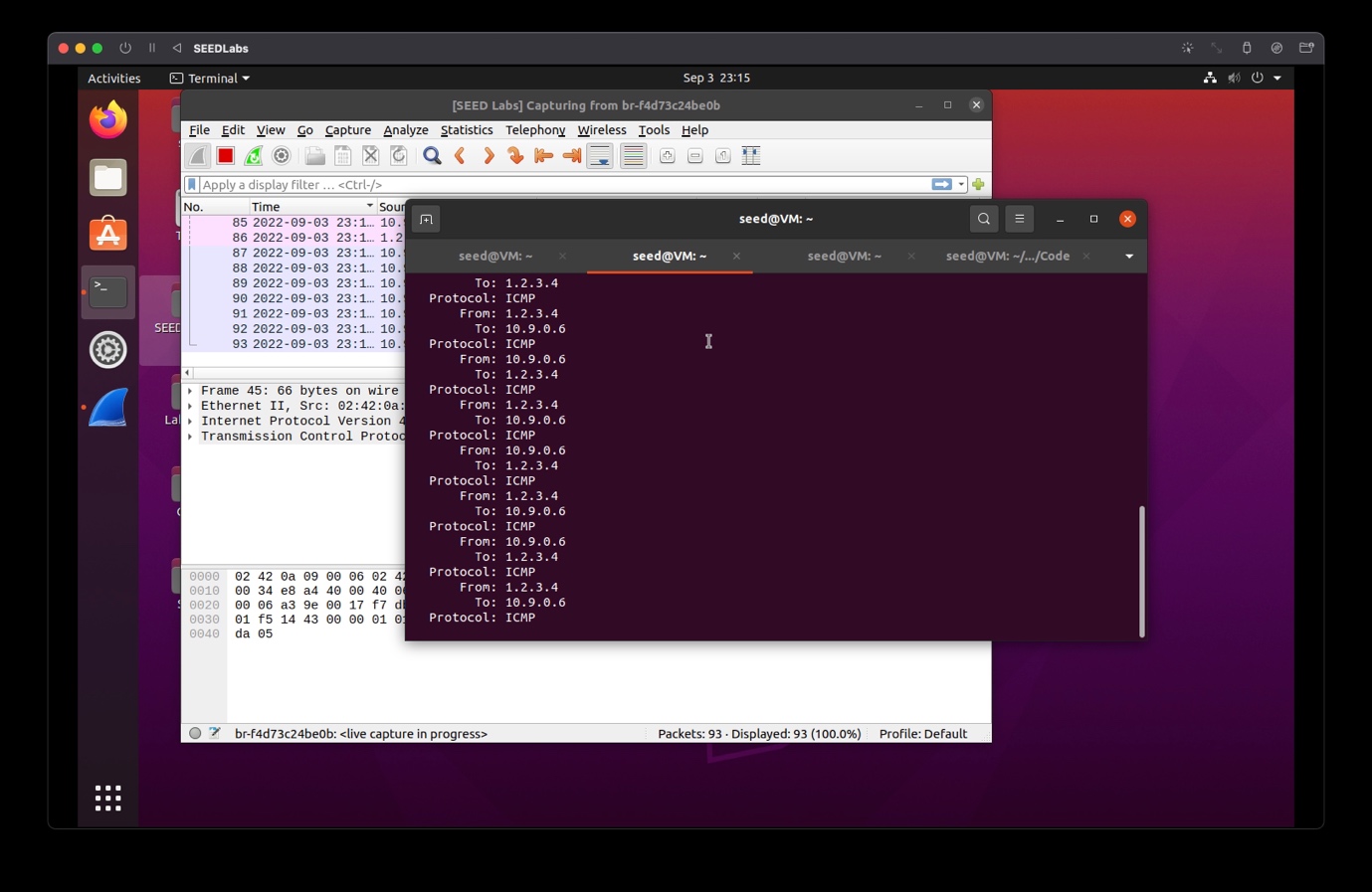
**On the host VM** :  
**# gcc -o sniffspoof Task2.3.c -lpcap  
# docker cp sniffspoof [Attacker machine docker container ID]:/volumes**

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

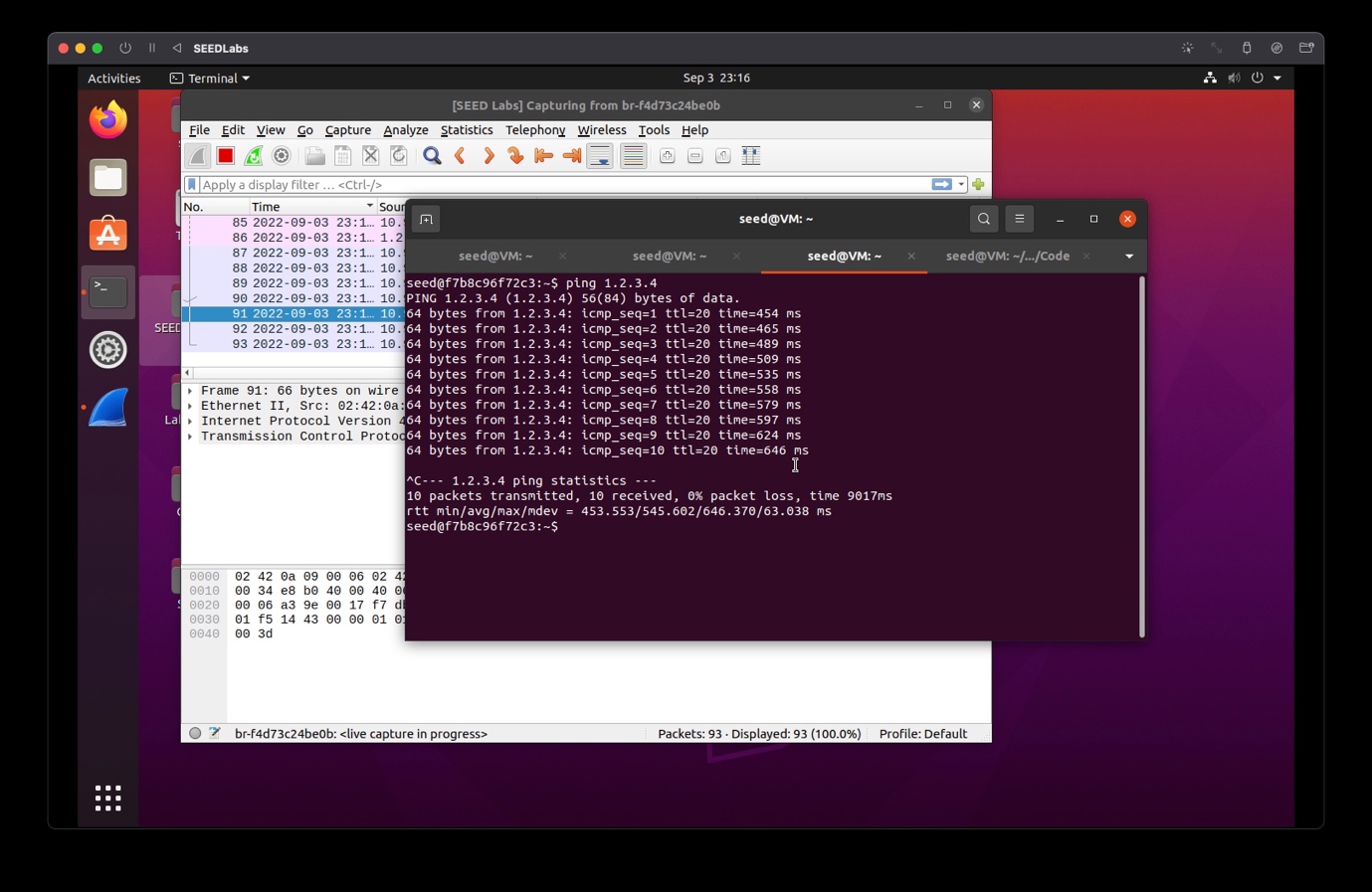
**On the Host A terminal ping 1.2.3.4 # ping 1.2.3.4**

Provide screenshots of your observations.

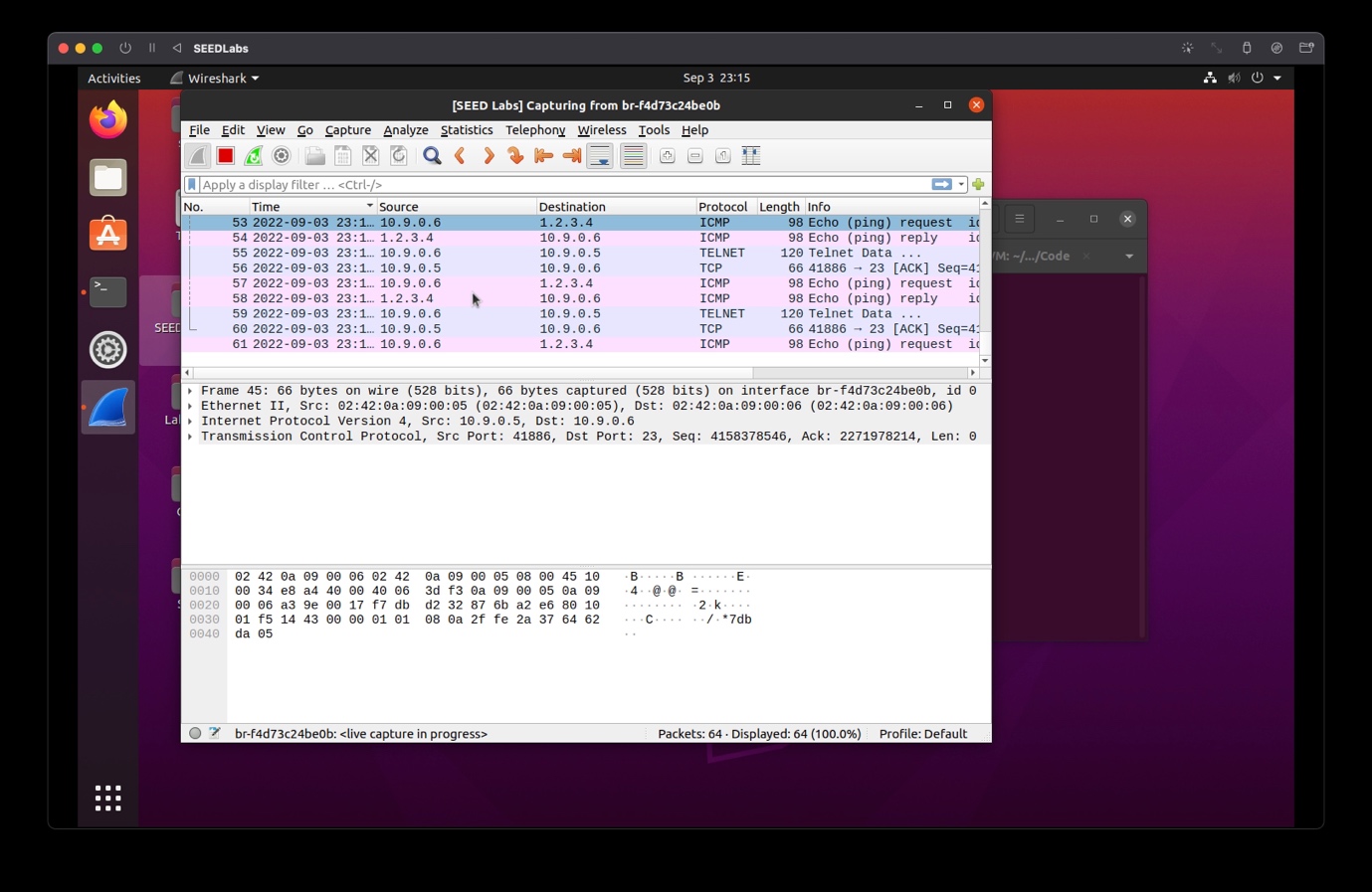
Attacker:



Host A:



Wireshark:



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