CNS LAB

ASSIGNMENT -2

Writing Programs to Sniff and Spoof Packets using pcap (C programs)

NAME : SIRI S

SEMESTER : 5

SECTION :H

SRN : PES1UG19CS485

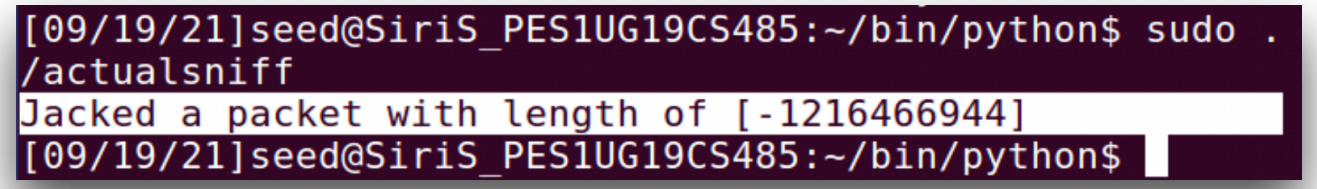
Screenshots of Execution:

ATTACKER: 10. 0. 2. 4

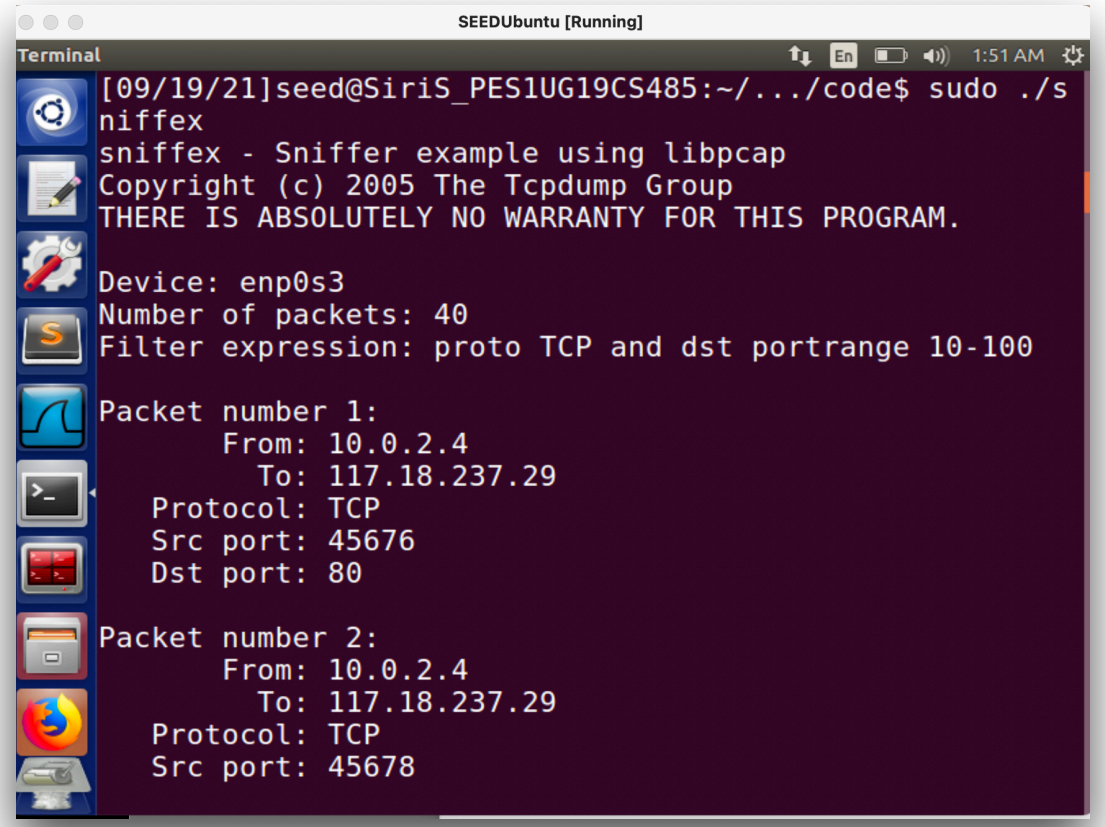
VICTIM: 10. 0. 2. 5

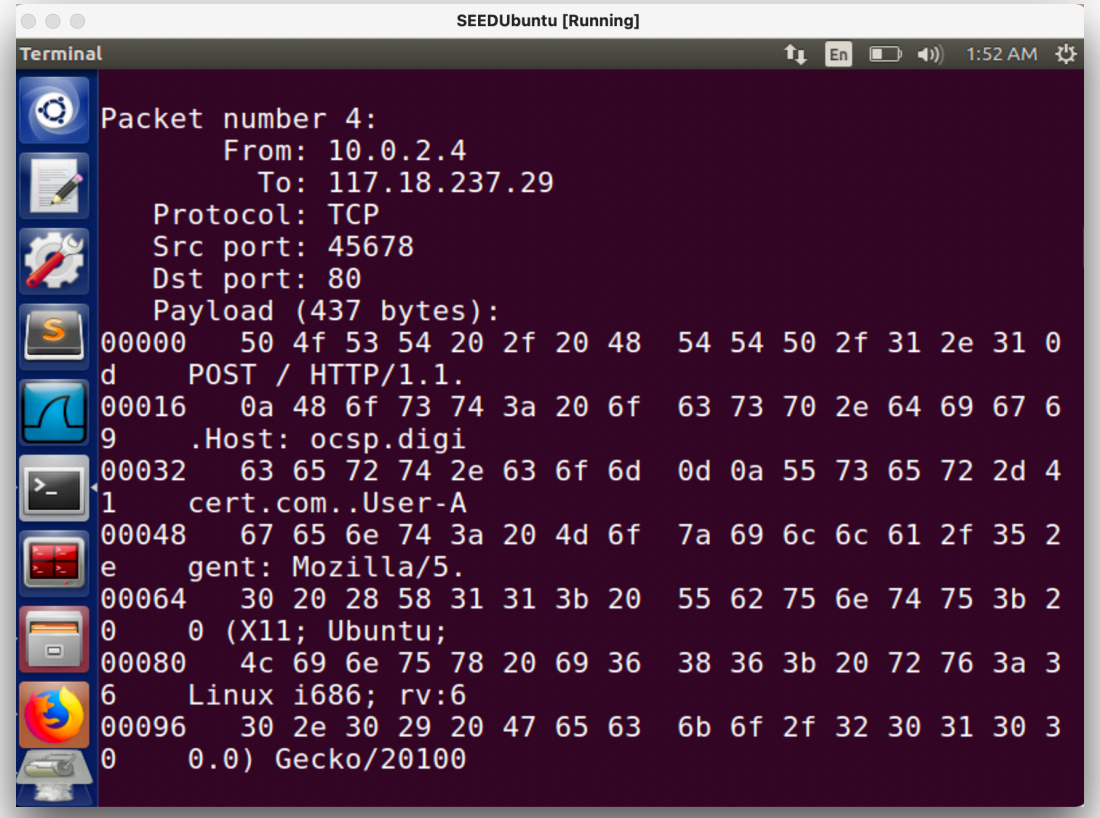
Task1: Sniffing-Writing Packet Sniffing Program

When we run the 'Actual Sniffer' code from www.tcpdump.org, we get the following



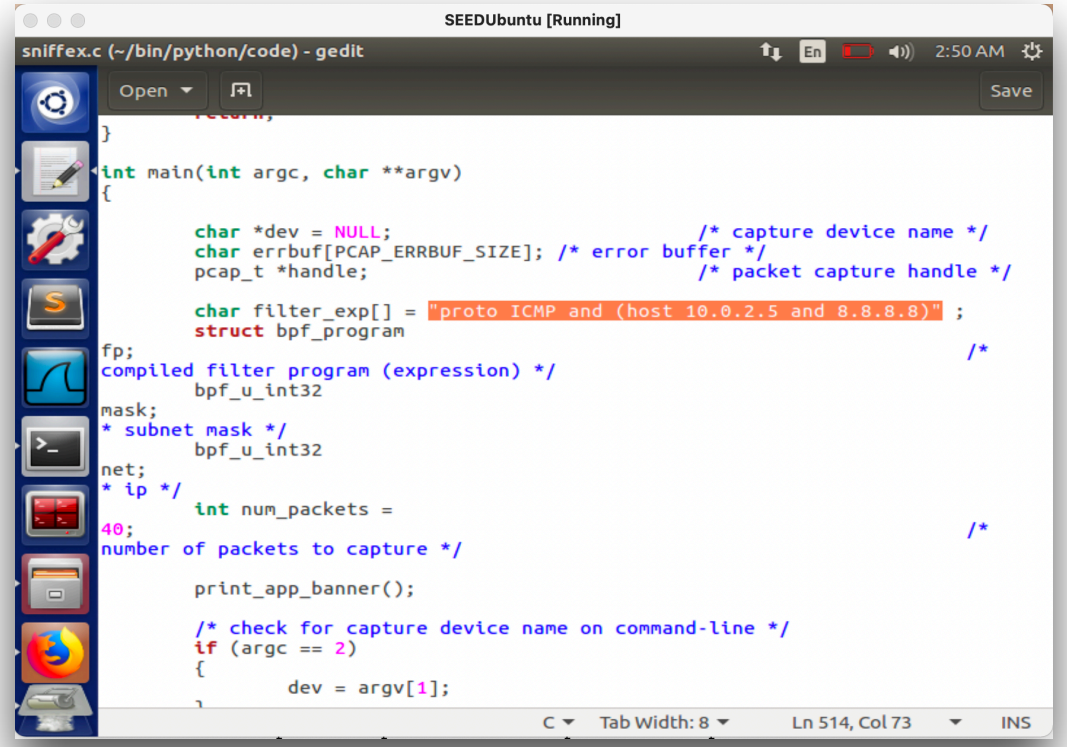
When we run the sniffer.c code from the attacker vm, the following packets are captured.



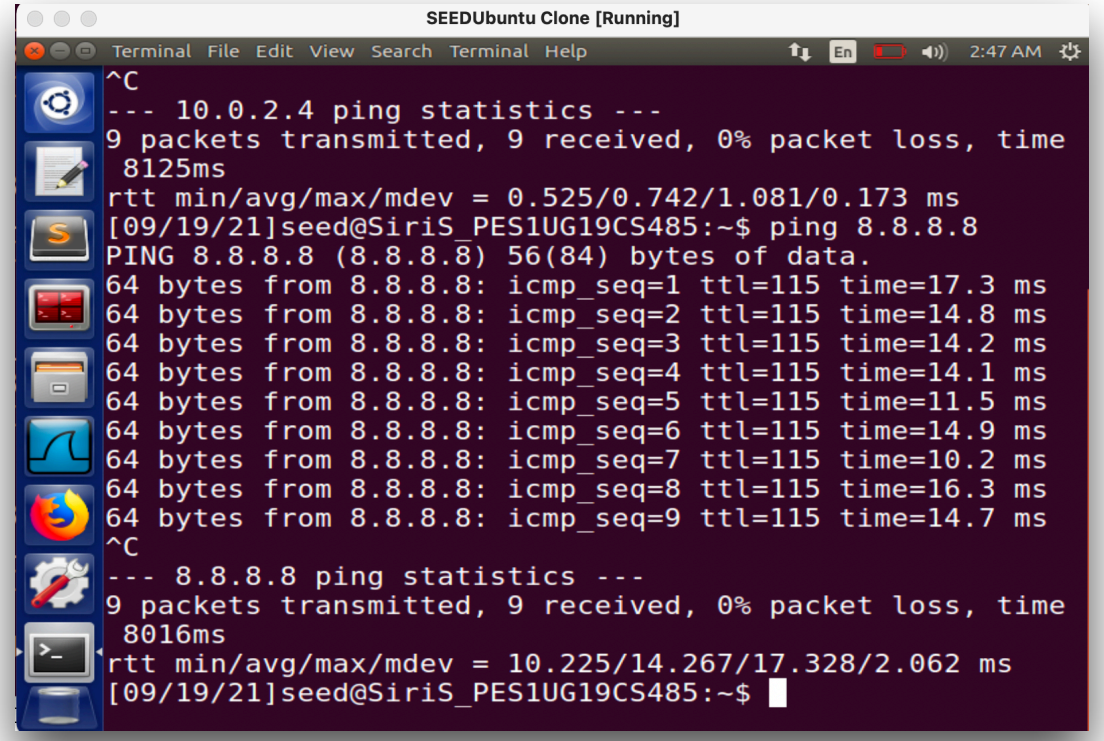


Task 1.1: Understanding how a Sniffer Works

Upon changing the sniffex.c code that is written using the pcap API and using the victim VM (10.0.2.5) to attack google (8.8.8.8)

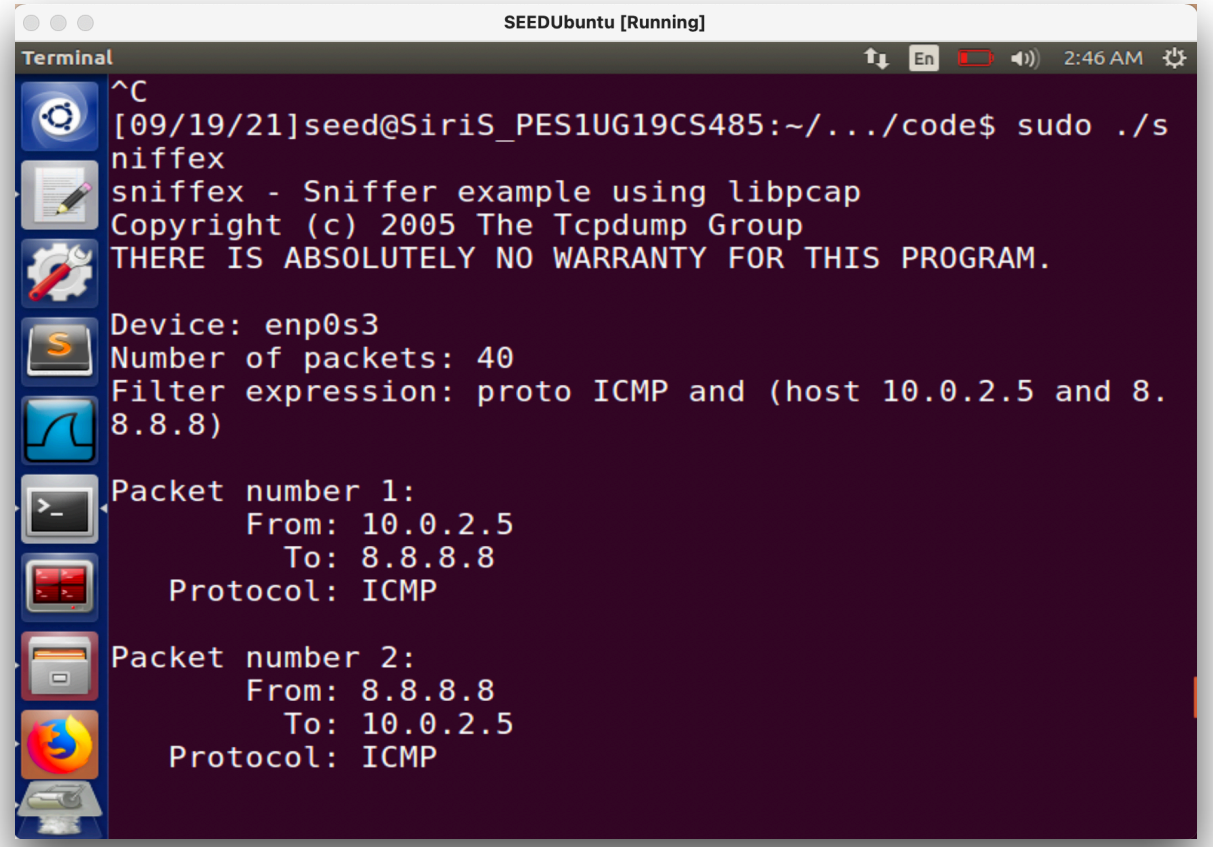


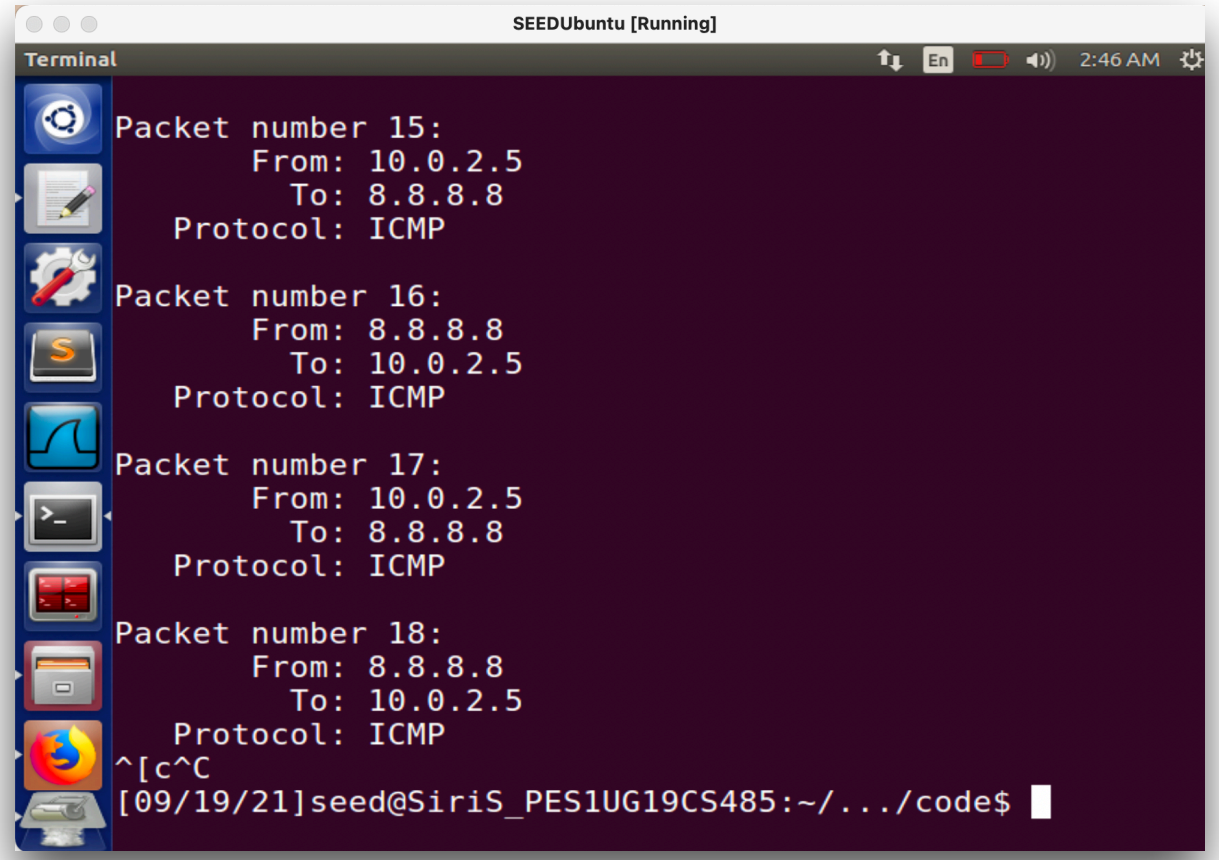
Changing the code



Pinging Google (destination address) from victim VM

The following icmp packets are captured in the attacker VM using pcap.





Problem 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 a detailed explanation like the one in the tutorial.

1. Setting up device
It can be defined using a string in the code for sniffer or sniffer decides the interface itself by picking up an active interface automatically.

2. Initialize Sniffing
Here in this step, after setting up a device for sniffing , sniffer Initialize PCAP and tell it to sniff on a particular device to create an environment for sniffing called as a session.

3. Traffic Filtering
For every session of sniffing you create you have to define any desire or rule, upon which packets are to be sniffed and analyzed. For example if you want to sniff HTTP traffic on a specific Interface of your computer , you will sniff TCP port 80 traffic on that interface (since http traffic uses port80), you will write your requirement in filter string and then compile it to apply the rule. This is called filtering and it is possible to use a blank filter but in that case it will be sniffing all packets and will be analyzing all fields.

4. Execution /actual sniffing
This is the execution part where sniffer is finally executed. Here we see that there are actually two main techniques to capture a packet, first is a packet is sniffed and then analyzed instantly whereas other is in which we enter into a loop that waits. for n number of packets to be sniffed before we go for analyzing part. It stores the results as asked by the user that is either to display the packets immediately or to save them in a file for future record.

5. Ending Session
End of the Session when you are done with the requirements of sniffing or you have enough data to analyze and to make any decisions or to make any perception about the network based on the analysis from the packets.

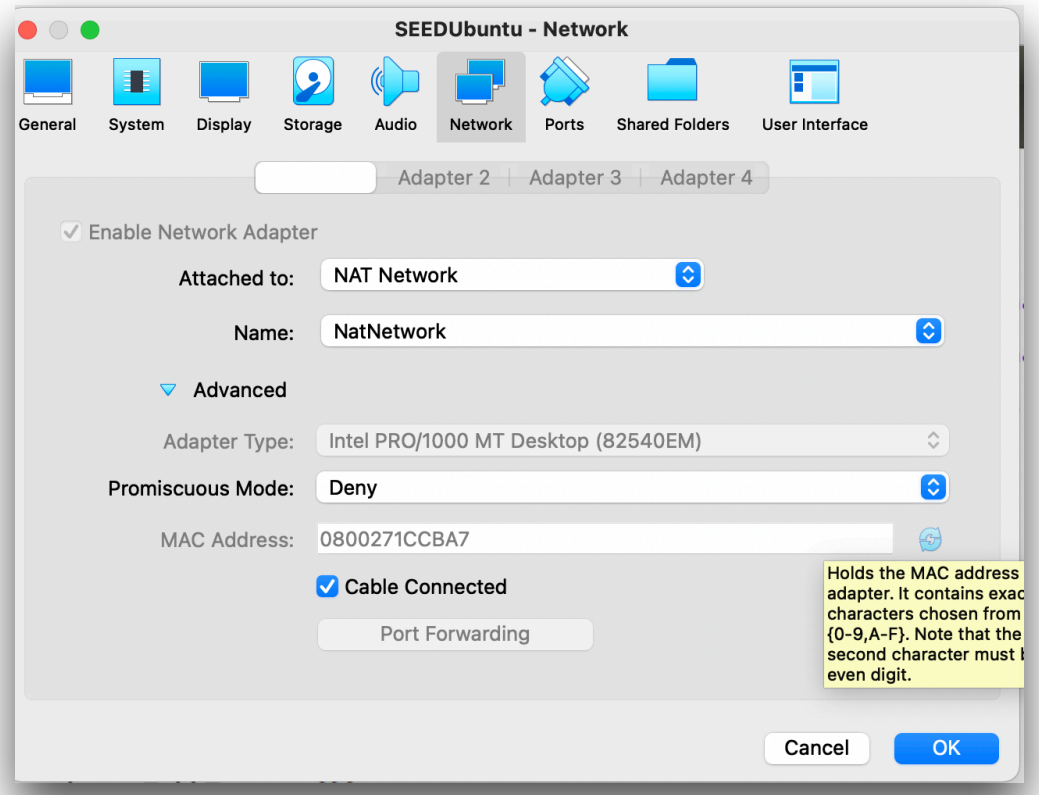
Problem 2: Why do you need the root privilege to run sniffex? Where does the program fail if executed without the root privilege?

Pcap_lookupdev() function needs root access because it wants to access network interfaces and it is impossible without root access in linux. Sniffer programs need raw sockets that allow direct sending of packets by the applications bypassing all applications in network software of operating system. And we need to be a root to create raw socket as we can't discover NIC until we are root.

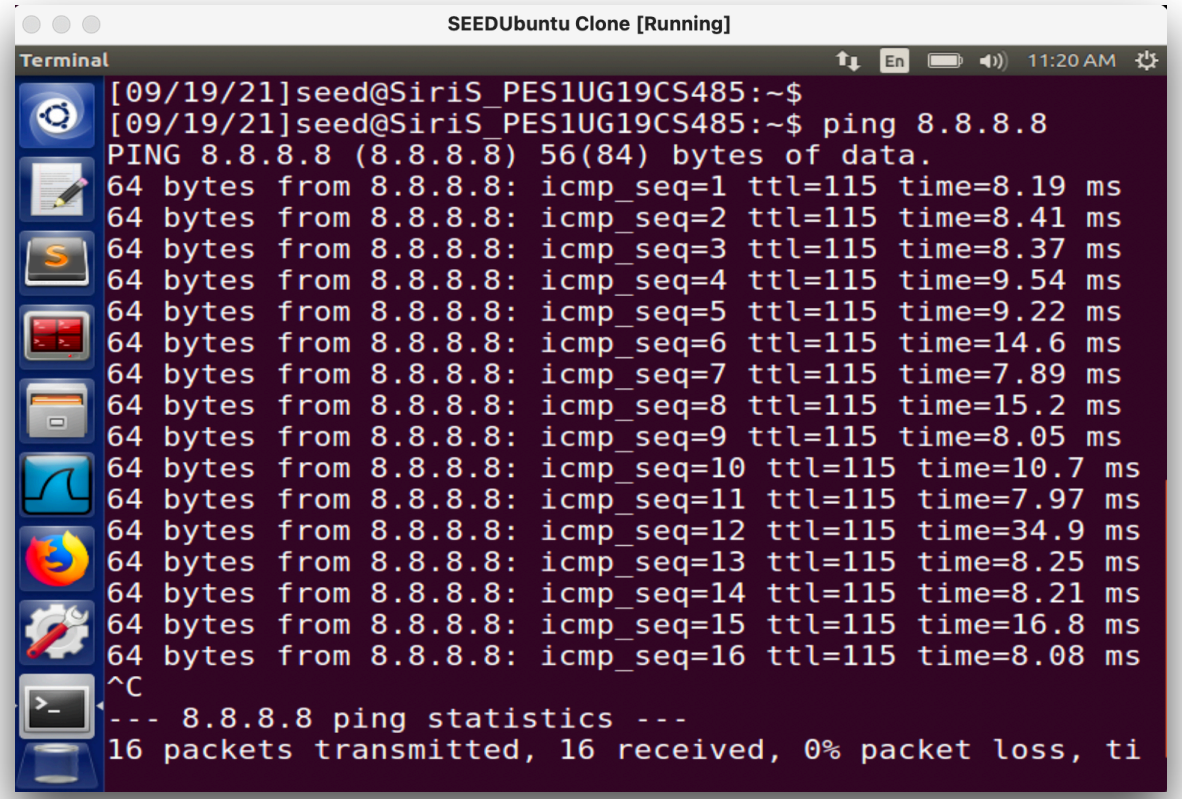
Problem 3: Please turn on and turn off the promiscuous mode in the sniffer program. Can you demonstrate the difference when this mode is on and off? Please describe how you demonstrate this

In really simple words promiscuous mode is one in which all the packets are sent to a computer or sniffed by sniffer and not only those which are addressed to it whereas in a non promiscuous mode only those packets are send to the computer or sniffed by sniffer which are addressed to it.

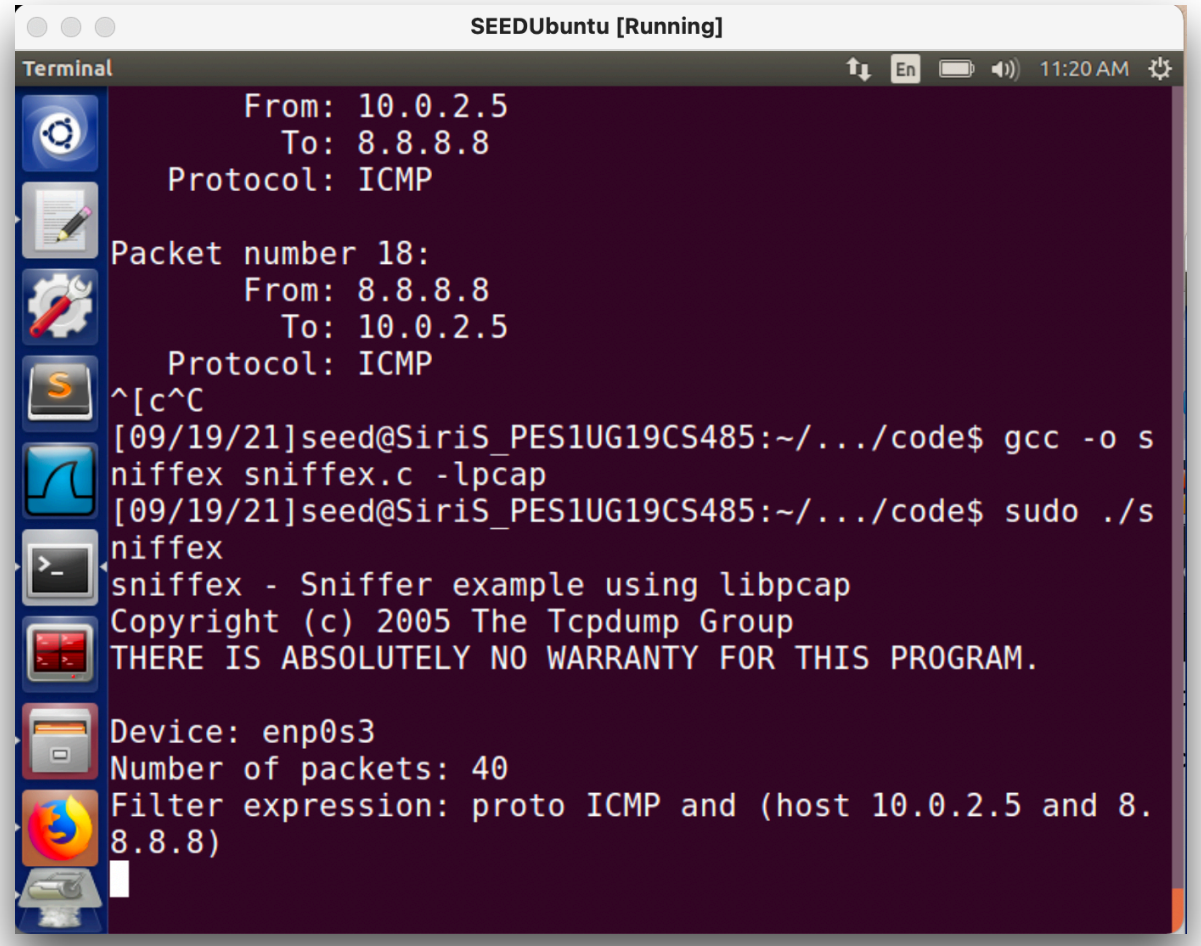
Screenshots:



Switching off the promiscuous Mode by enabling 'Deny' in network settings



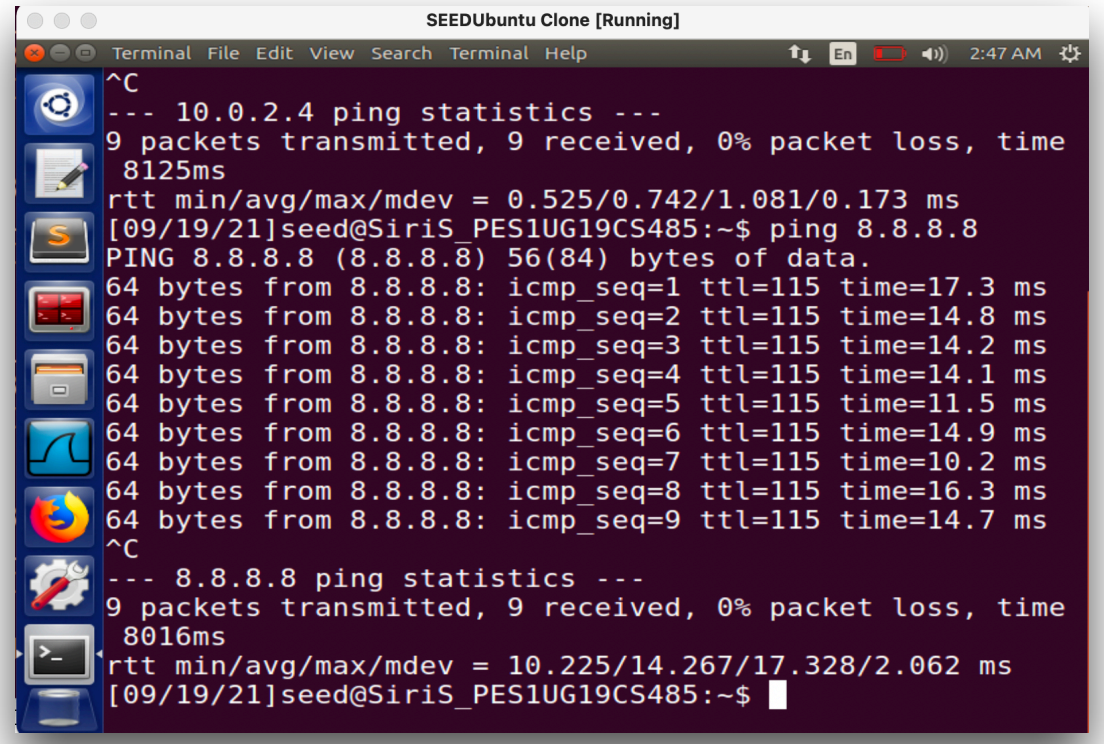
Pinging 8.8.8.8 from victim VM



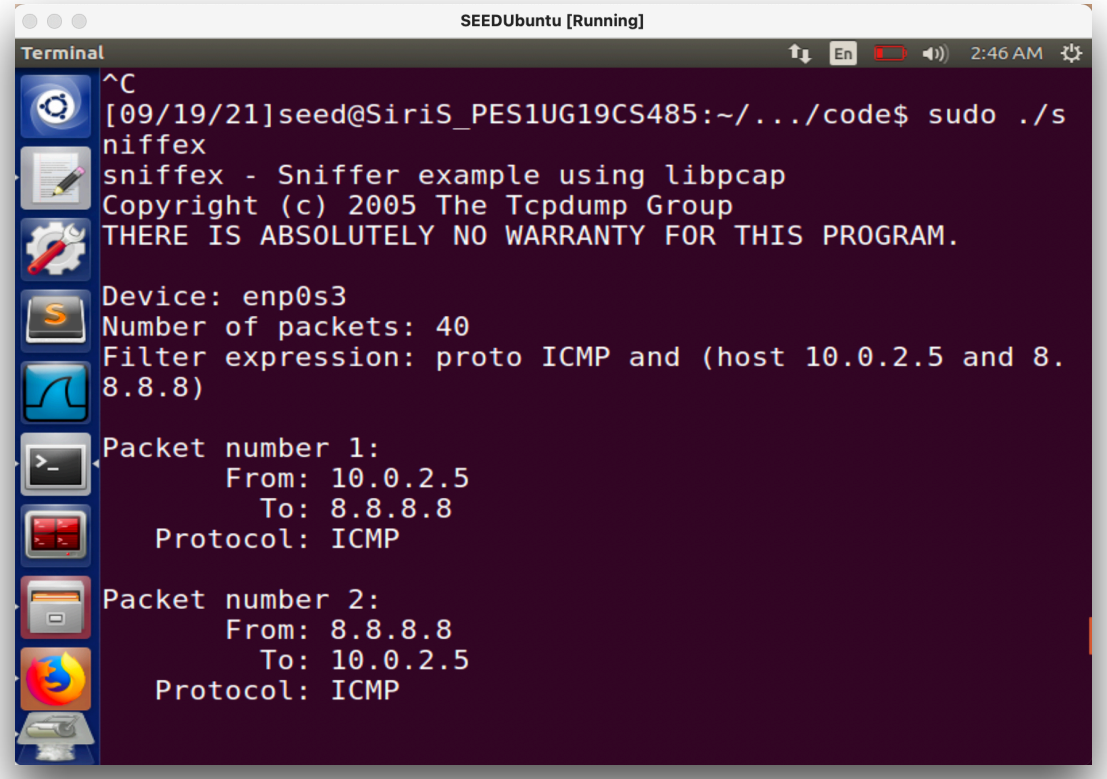
No capture of packets because promiscuous mode is off

Task 1.2: Writing Filters

i) Capture the ICMP packets between two specific hosts

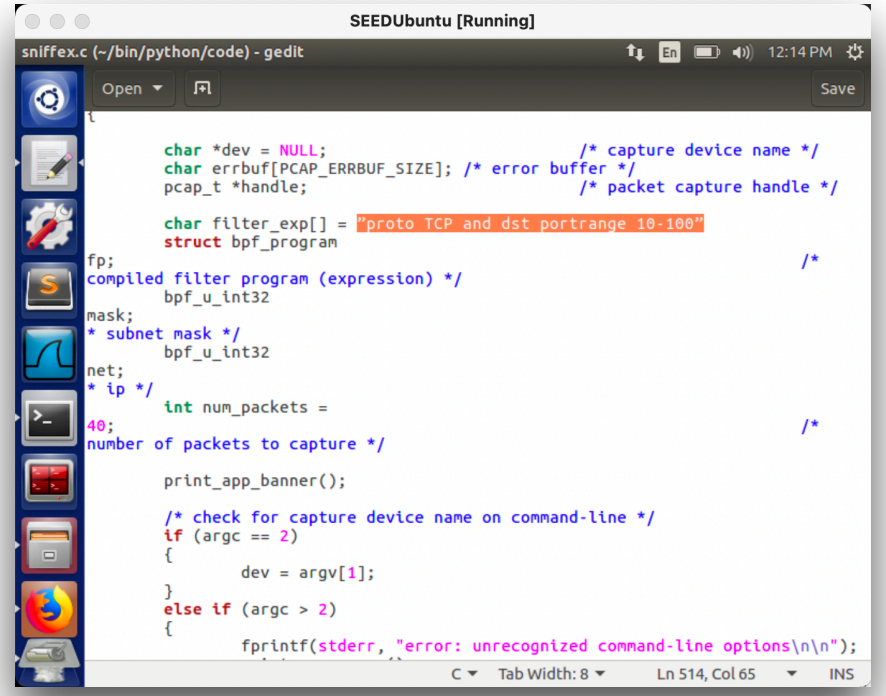


Pinging Google (destination address) from victim VM

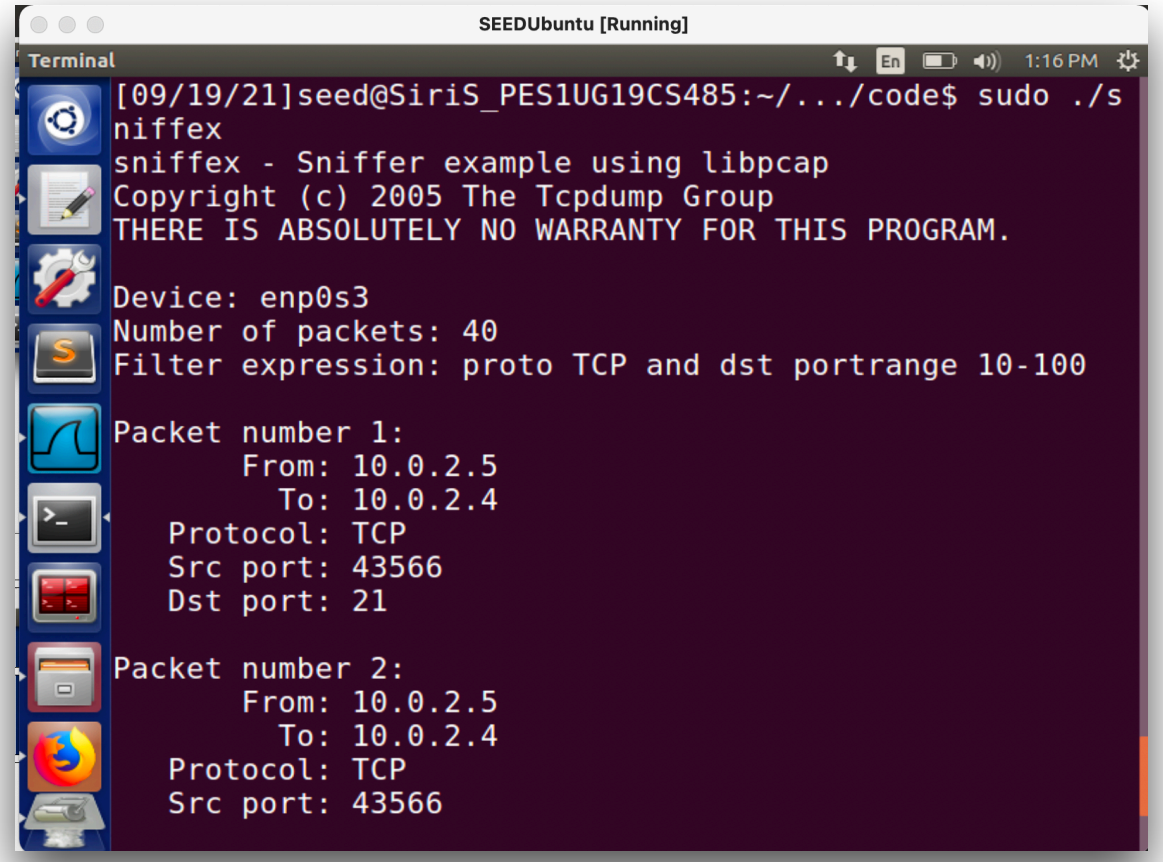


ICMP packets captured by the attacker

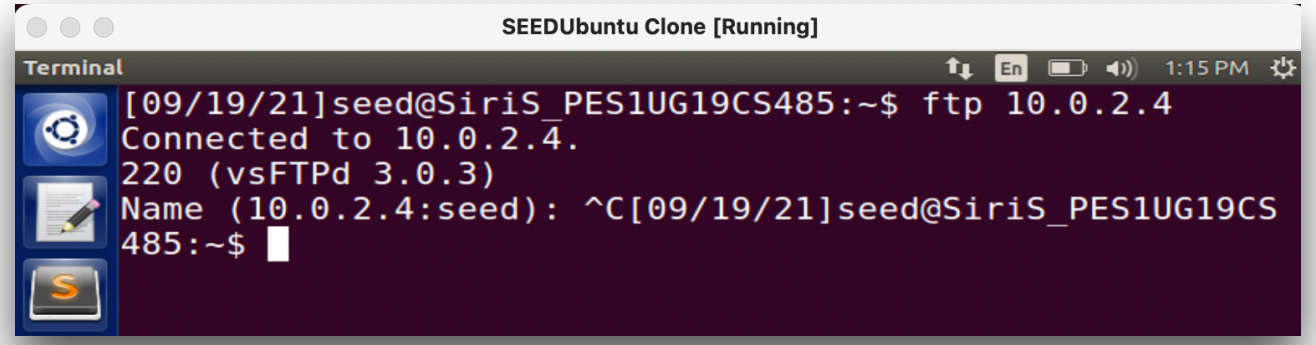
ii) Capture the TCP packets that have a destination port range from to sort 10 - 100.



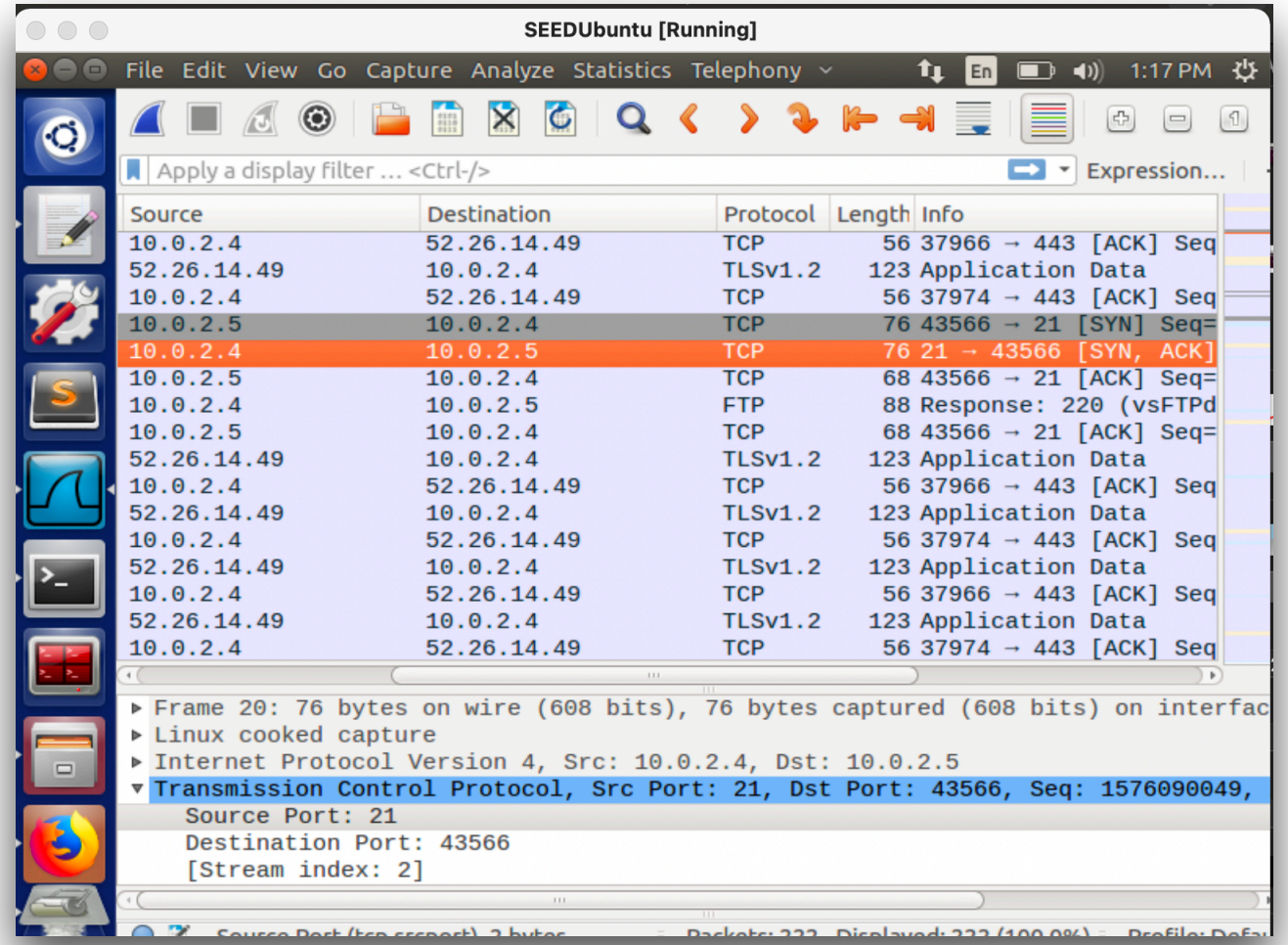
After changing the sniffed.c code



Running the code to capture packets

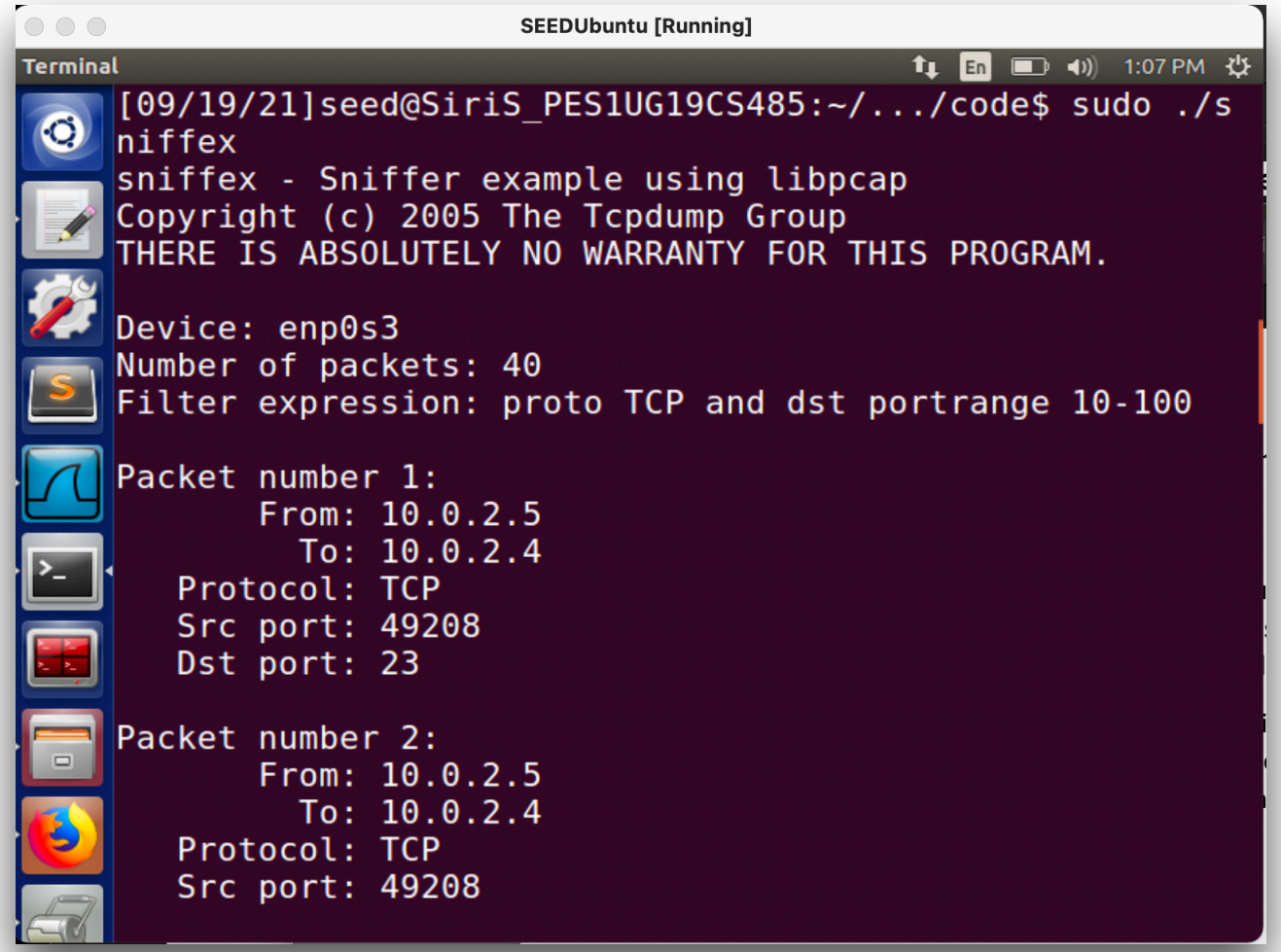


Sending ftp packets to the destination machine

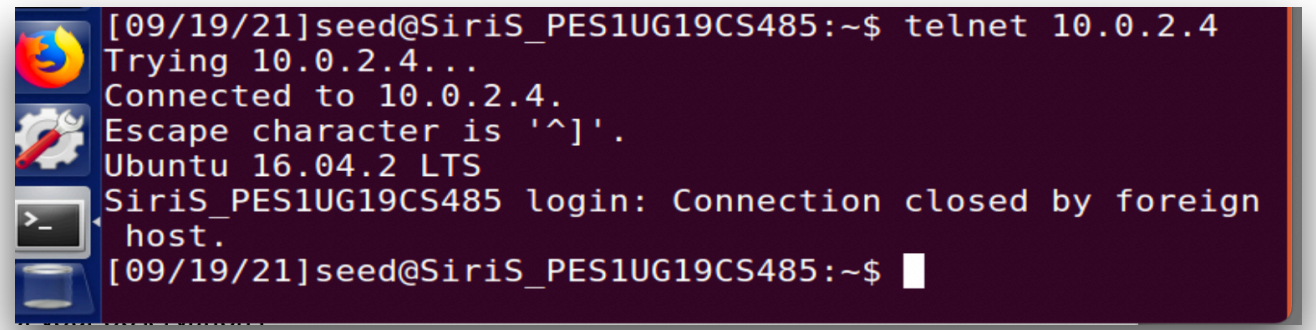


Wireshark screenshot of the packets captured along with source (21) and destination port address

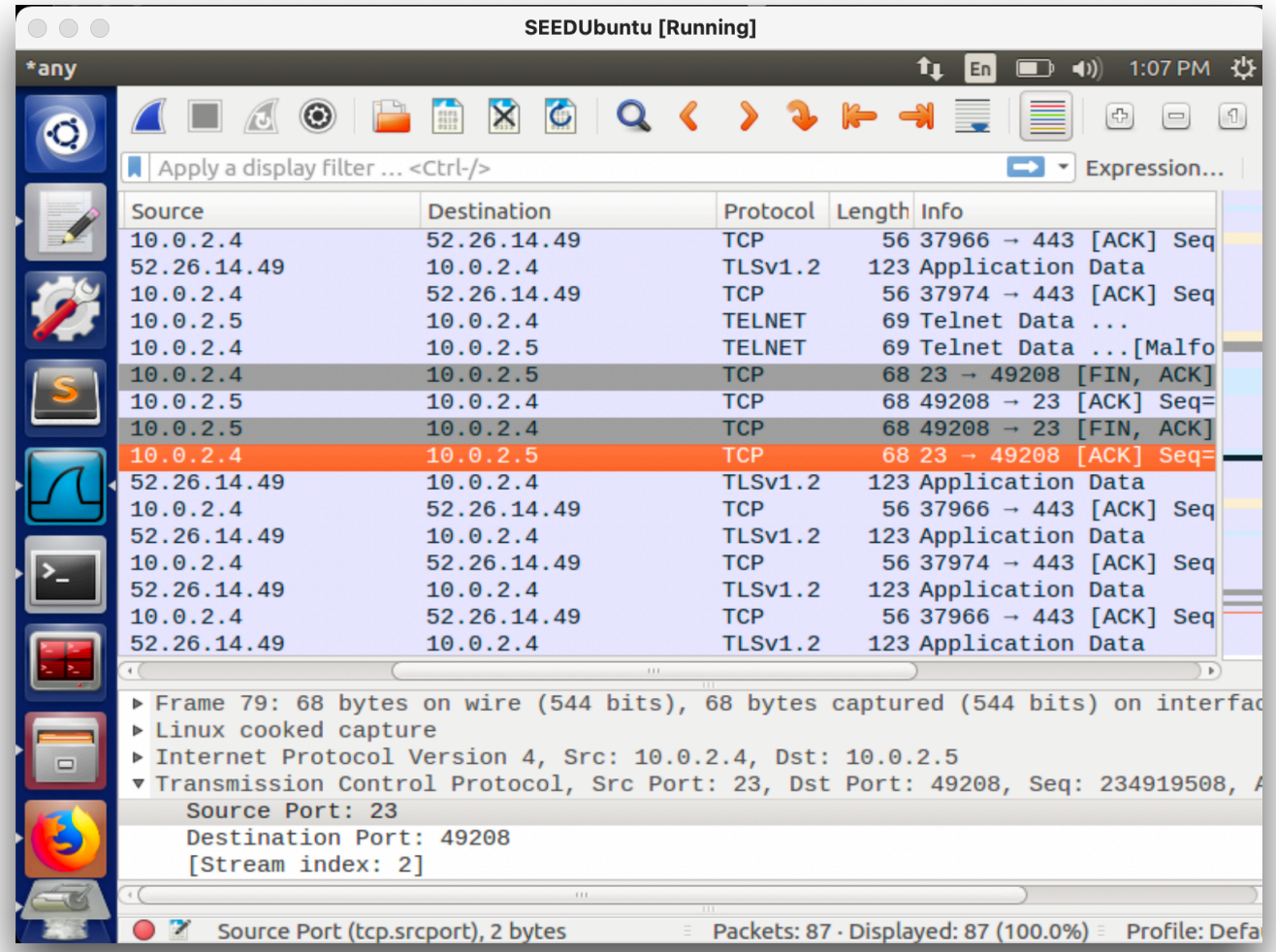
Task 1.3: Sniffing Passwords



Running the code which has a port range that includes port 23



Running telnet on the victim machine using the attacker machines ip address

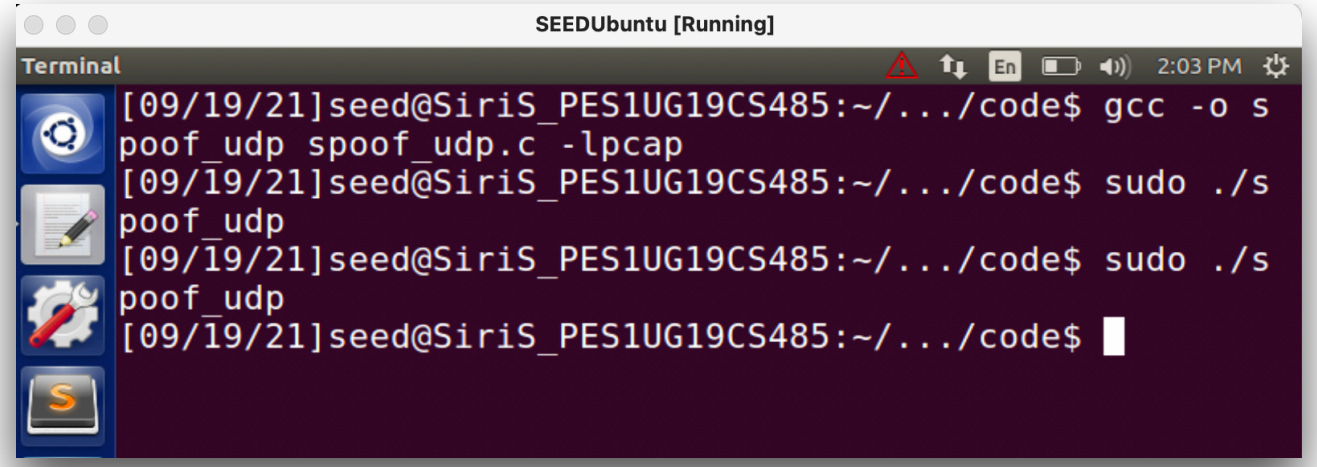


Wireshark screenshot showing the TCP packets captured in port 23. We will connect to a telnet server (running in our VM) and get the password of the user.

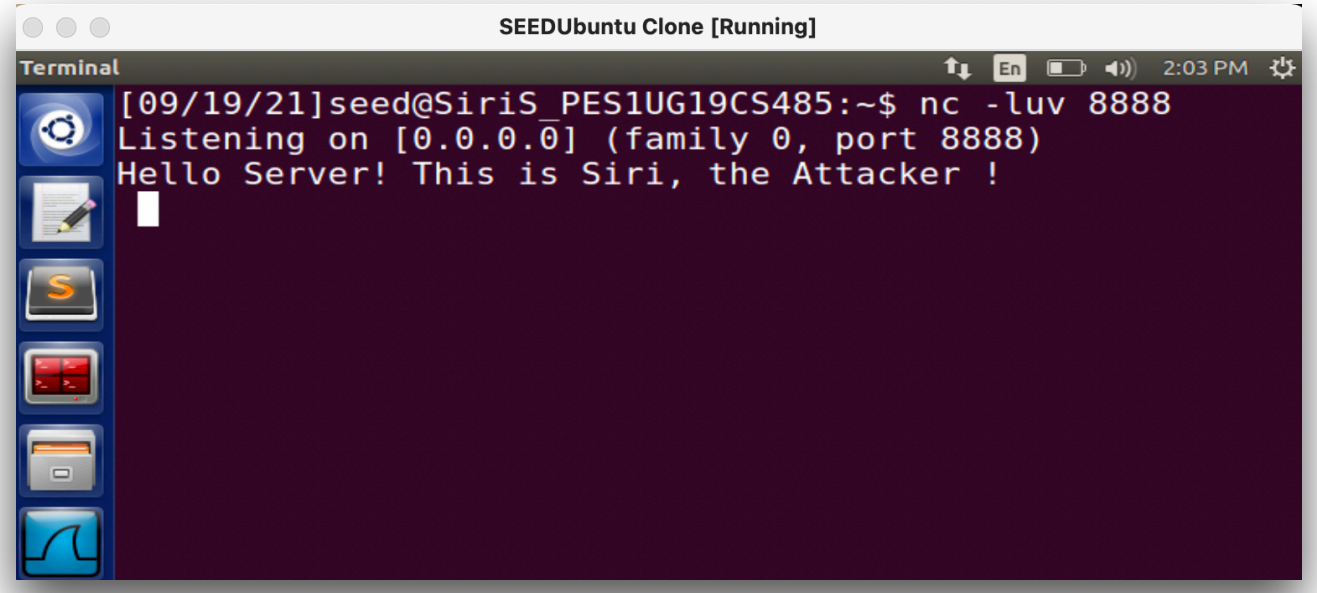
Task 2: Spoofing

Task 2.1 - A Writing a spoofing program:

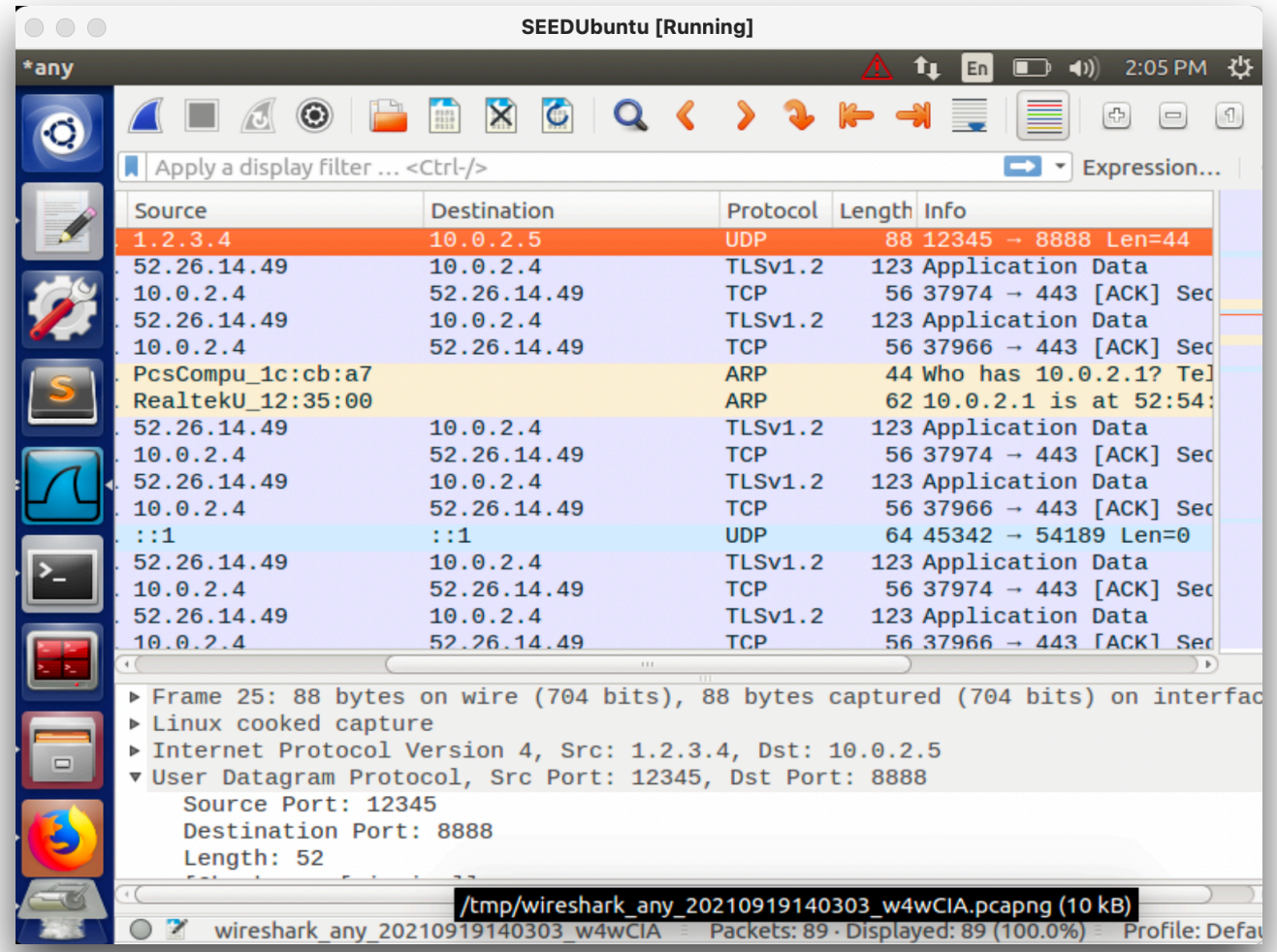
Upon running spoof_udp.c after changing the victim ip address



Running the code

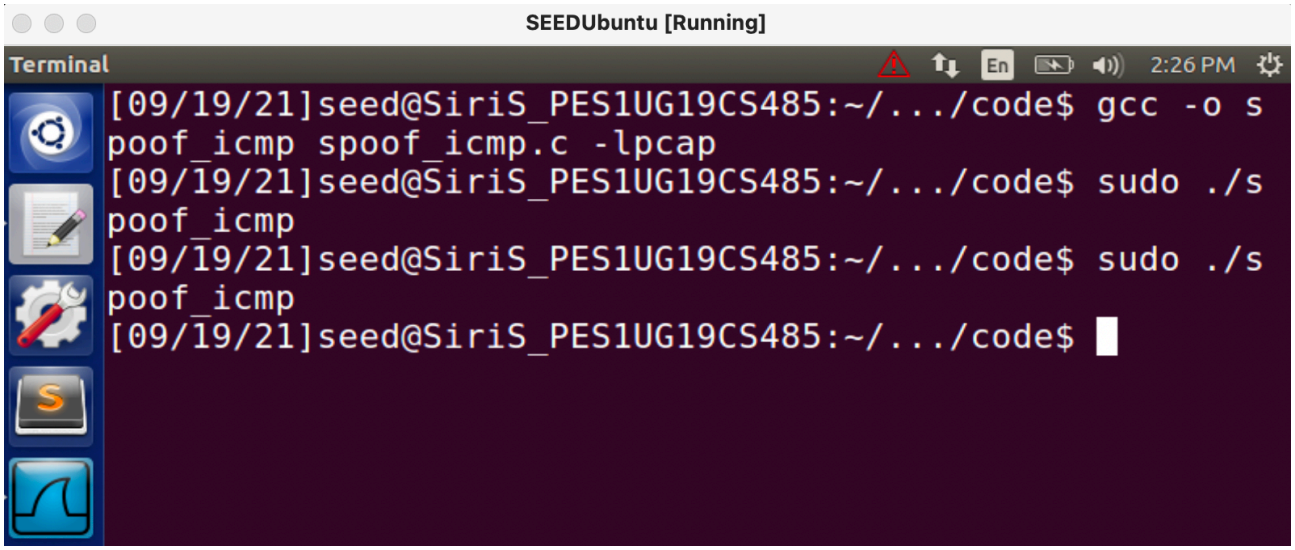


Victim machine opens a listener at the user end to listen to the connections and we see that the user machine receives the UDP packet sent by the spoof program.

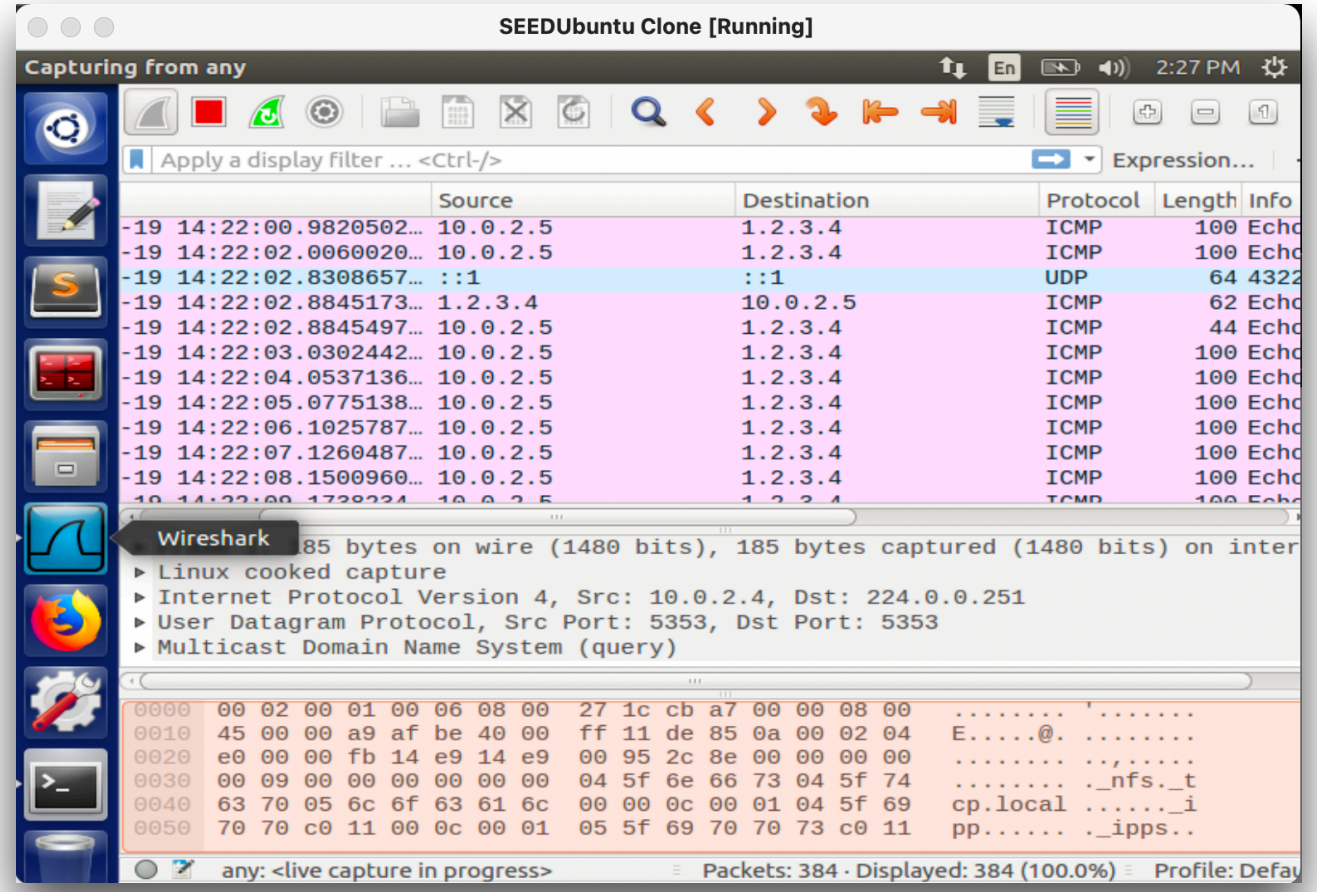


Wireshark screenshot to show the data that's captured due to spoofing

Task 2.2 – Spoof an ICMP Echo Request



Running the code for spoof_icmp



Wireshark capturing data