

CS5460 – Assignment 5



March 28, 2016

Varun GATTu

A02092613

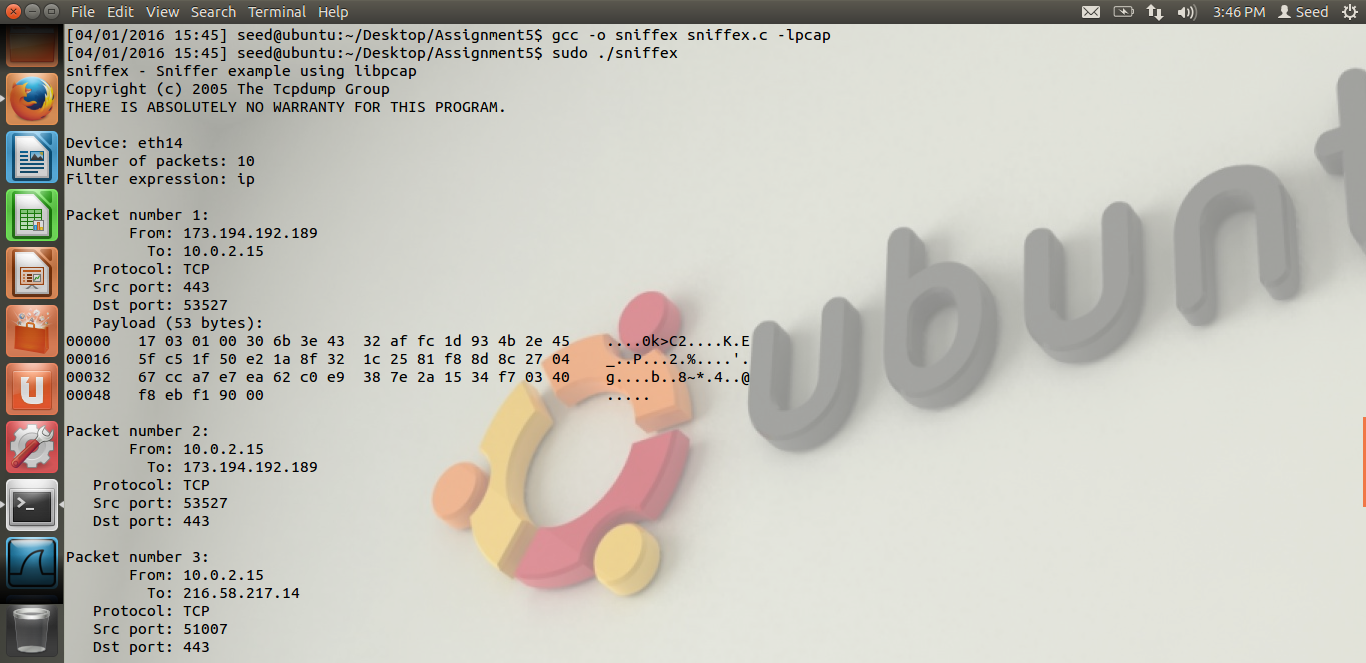
**Packet Sniffing and Spoofing Lab**

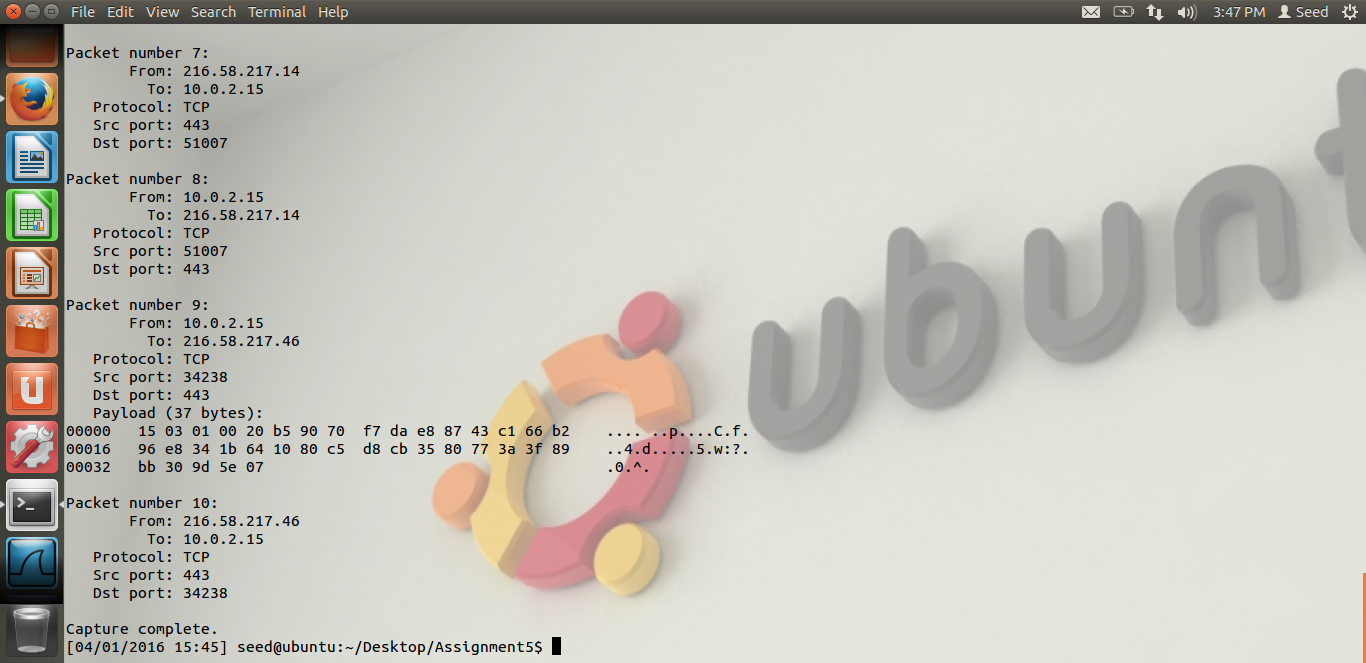
**2. Lab Tasks**

**2.1. Task 1: Writing Packet Sniffing Program**

**Task 1a: Understanding Sniffex**

Here, I downloaded the program from the provided link and compiled it with –lpcap extension. Then, I executed the program from the root and produced the following output:





**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 detailed explanation like the one in the tutorial.

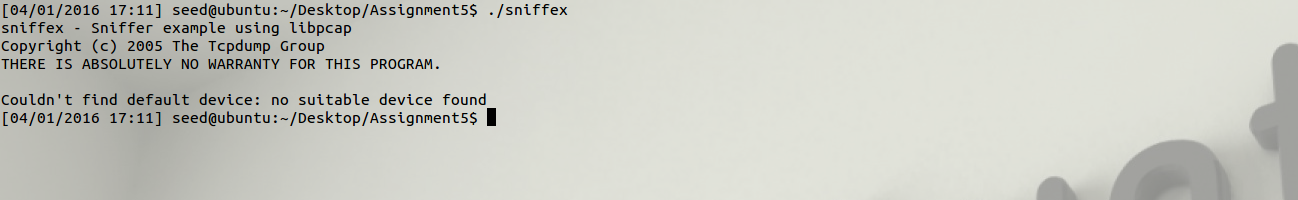
**Answer:**

1. Setting the device: this can be done in two ways:
   1. Device details are provided as a string when executing (argv)
   2. Use pcap\_lookupdev() function to set a device on its own and failure results in saving the error message to errbuff
2. Open the device for sniffing: this can be done using pcap\_open\_live() with the following arguments to it
   1. The device that is specified in step a
   2. Snaplen is the number of bytes to be captured by pcap
   3. Promisc is to set the promiscuous mode
   4. To\_ms is the read timeout in milliseconds
   5. Ebuf is to store error messages
3. Filtering the traffic: filter is applied after the expression is compiled.
   1. Pcap\_compile() compiles the expression
   2. Pcap\_setfilter() applies the filter
4. Actual sniffing: this can be done in two ways – capture single packet at a time or capture a set of packets using loops. The below functions are self-explanatory
   1. Pcap\_next()
   2. Pcap\_loop()

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

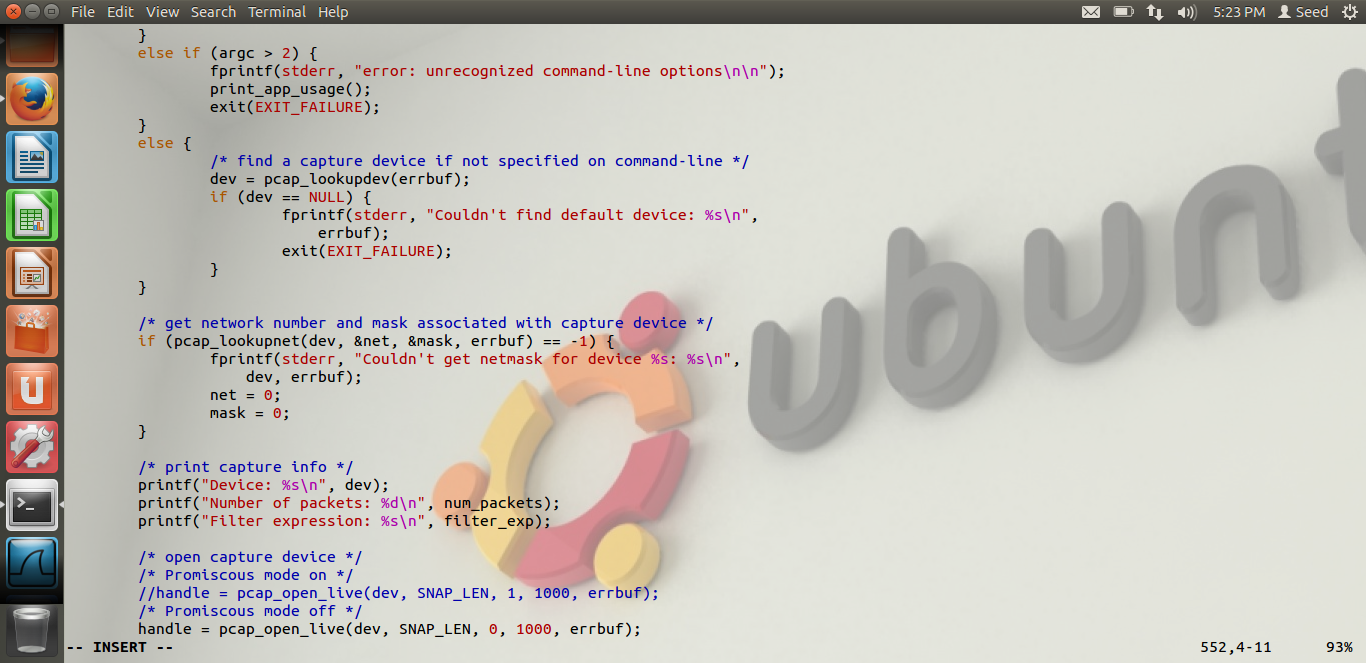
**Answer:** Sniffex needs root privileges to execute because it uses pcap\_lookupdev() which tries to select a device on its own to sniff. This function will not work if the program is not executed in root.

The output is shown below if tried to execute without root access:



**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.

**Answer:** Turning on or turning off promiscuous mode can be done in the function call pcap\_open\_live(). The argument is set to 0 when the mode is off and vice versa. I manually turned it off as shown here:



Then I again compiled and executed the program to get the following output:

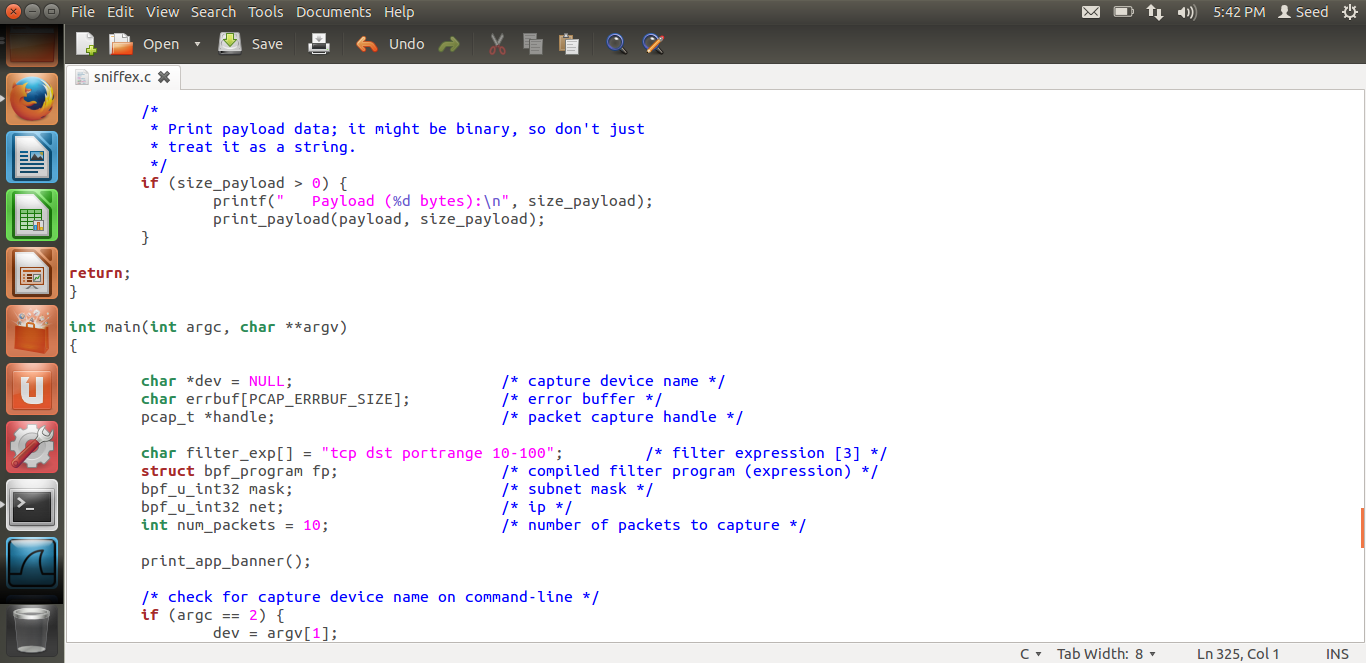


This shows that when the promiscuous mode is turned off, the sniffer program only captures packets from a single source and single destination IP address unlike when it was turned on, the program captured packets from multiple devices.

**Task 1b: Writing filters**

* Capture the ICMP packets between two specific hosts
* Capture the TCP packets with destination port range from 10-100

Here, I have changed the filter\_exp[] from “ip” to “tcp dst portrange 10-100” to capture all the packets that are sent only to TCP ports with range 10-100. The below screenshots show how and what happened:

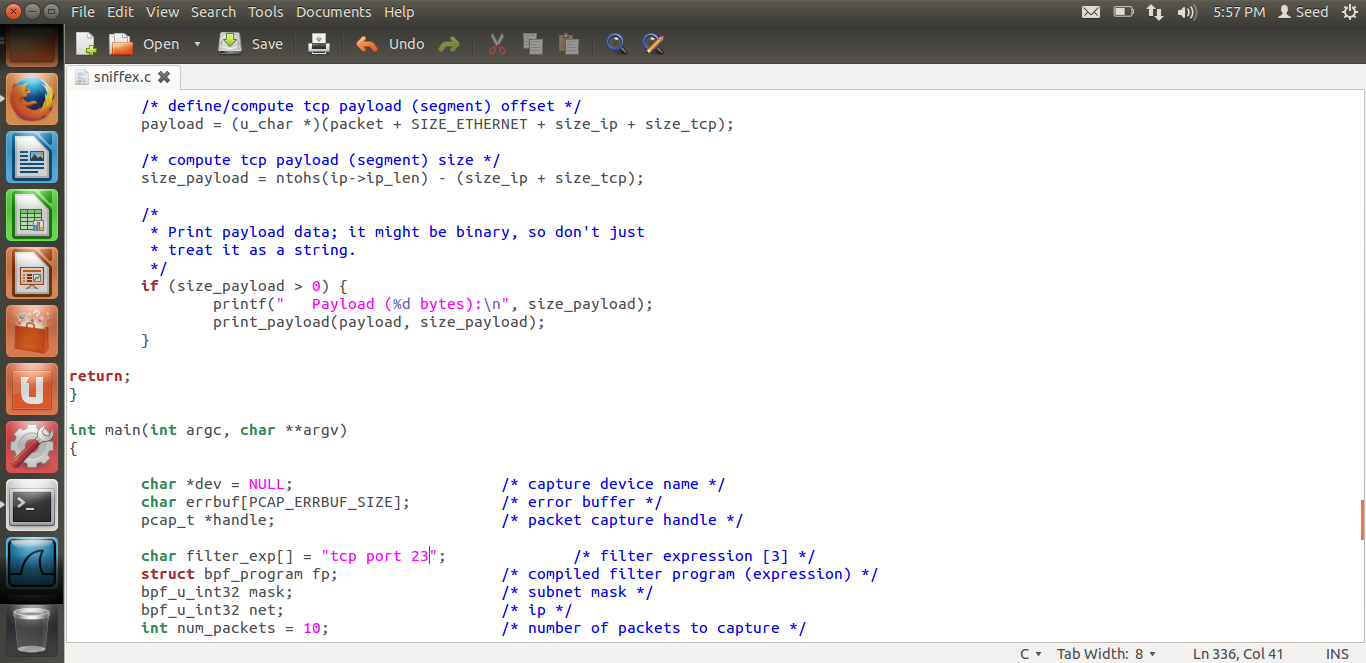




It shows that all the captured packets have DST PORT with values 10-100 only.

**Task 1c: Sniffing passwords**

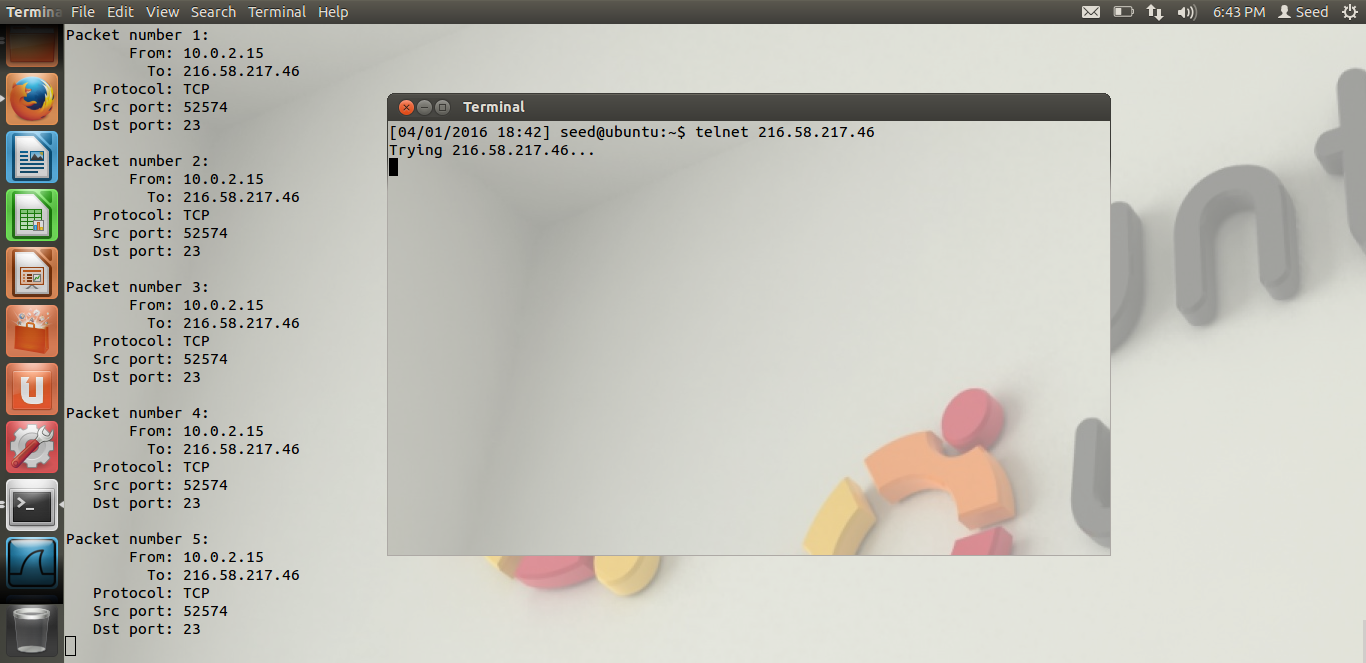
To perform this task, the filter expression needs to be only 23 because we are trying to capture all the telnet packets. Because TCP has a port number of 23. This can be done as follows:



We then start the telnetd server like the following:



Now, when executing the sniffex program, we invoke an IP address to capture the TCP packets on the flow like the follows:



**2.2 Task 2: Spoofing**

**Task 2.a: Write a Spoofing Program**

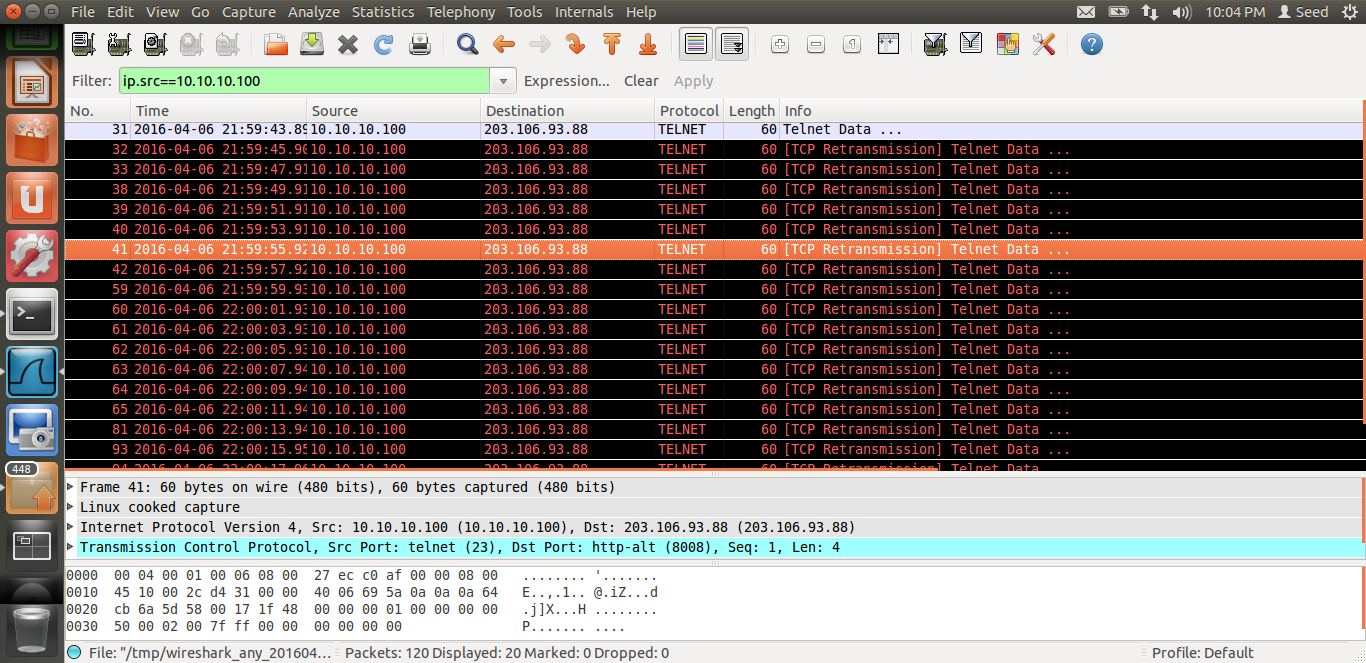
Here, I have used a program for spoofing which sends spoofed out packets to a destination with a different IP address. The output is similar to the below picture:



Here, the program takes 4 parameters as input through command line:

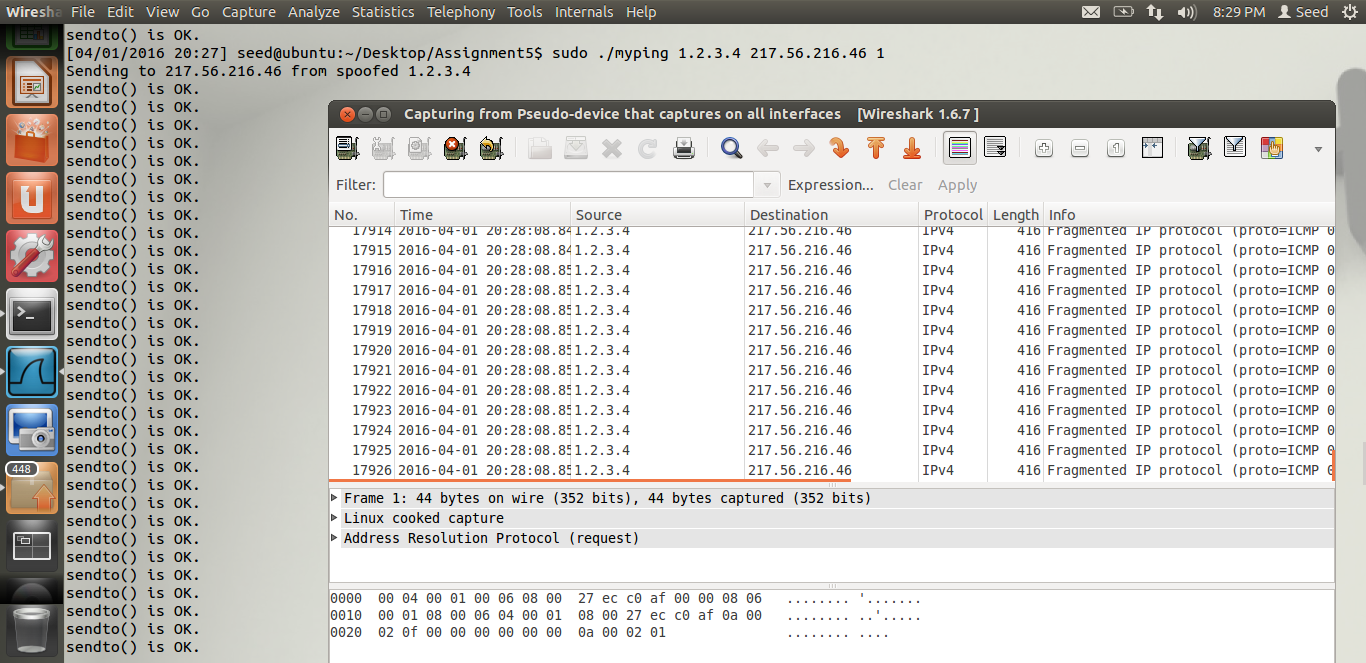
1. Source IP
2. Source port
3. Target IP
4. Target port

It means that the target IP received spoofed packets from the so called source ID which is not the genuine source. The Wireshark trace is shown here:



**Task 2.b: Spoof an ICMP Request**

Here, we try to spoof with an ICMP request. We try to ping the ICMP server to spoof the echo service it offers. The following output shows how it happens:



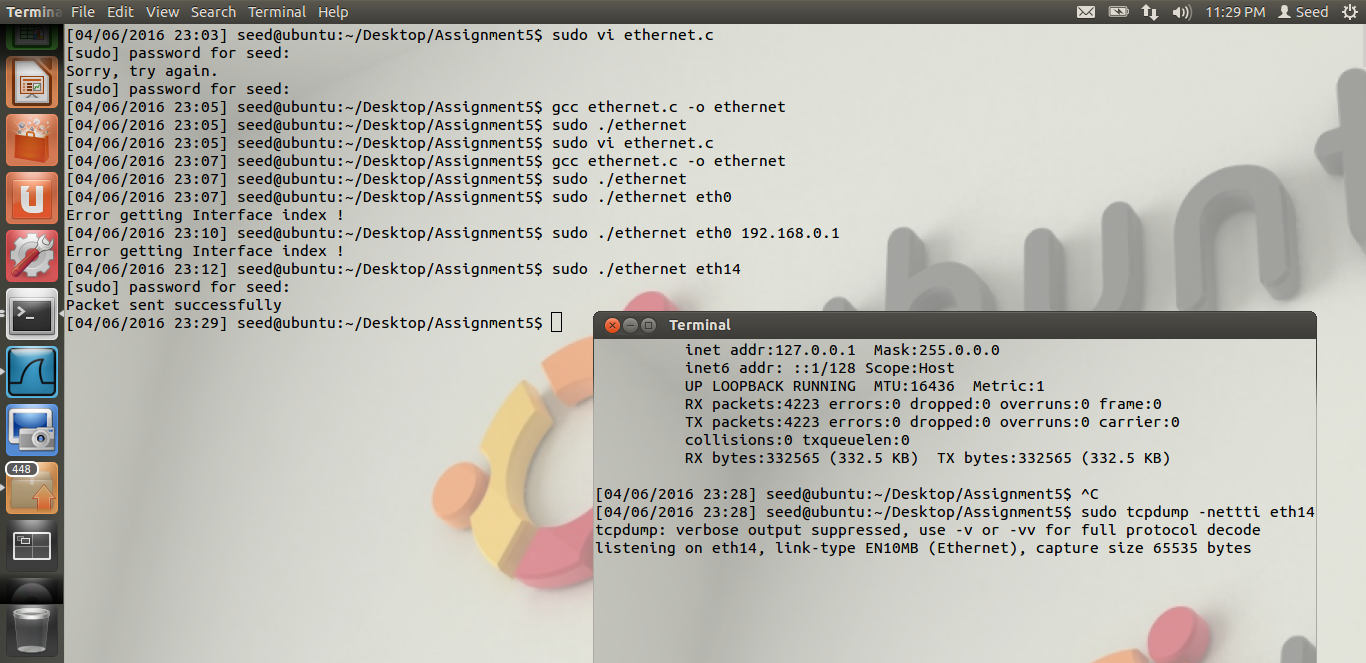
Here, the program takes 3 parameters as input through command line:

1. Source IP
2. Target IP
3. Number of packets to send

The Wireshark trace shows that the attack is working and runs successfully.

**Task 2.c: Spoof an Ethernet frame**

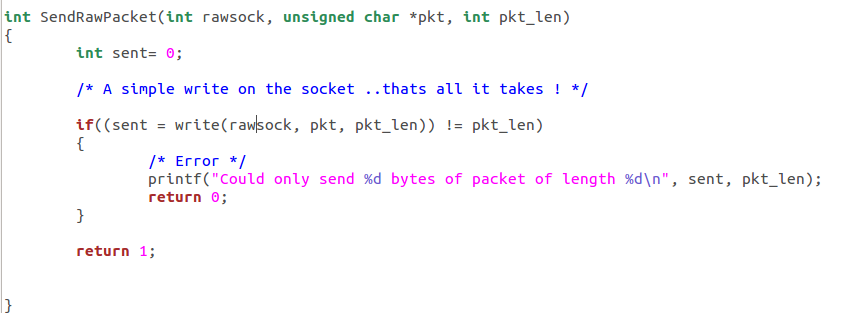
This was the toughest part to spoof an Ethernet frame. I have used the eth port which is already active on the system to get this work done. The eth port is used as the one that accepts packets and starts listening using tcpdump. Then, in the source program, I have used the eth port as a parameter with the input when I run the program. The output is as displayed:



**Questions:**

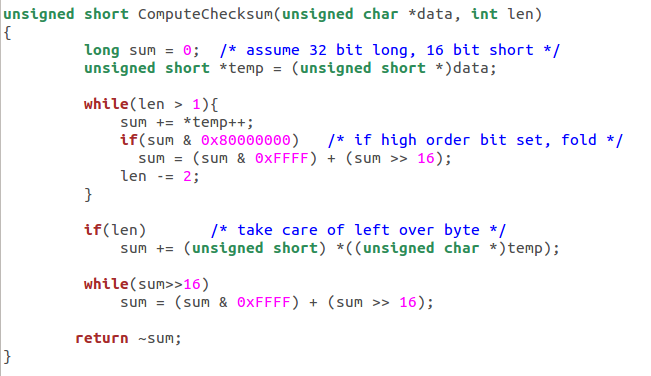
**Question 4)** Can you set the IP Packet length field to any arbitrary value, regardless of how big the packet is?

**Answer:** Yes, IP Packet can be initialized to any length no matter what. But, only the length of the raw socket size will be used from it. If the IP packet is 20 bytes and the Raw socket size is 10 bytes, the remaining 10 bytes will not be considered in the IP packet.



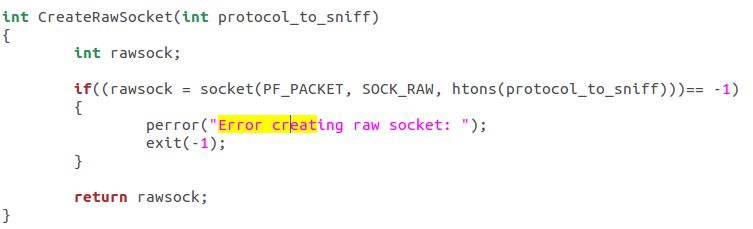
**Question 5)** Using the raw socket programming, do you have to calculate the checksum for the IP header?

**Answer:** Yes, we compute the checksum when doing socket programming.



**Question 6)** 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?

**Answer**: We need to run the program only in root privilege.



**2.3 Task 3: Sniff and then Spoof**

Here, the task is to combine both the above tasks to sniff and then spoof a target. For this, we need to VM’s cloned and then the second VM to act as the destination. We will have to compile with –lpcap as we did in the 1st task and then run the program with the port as a parameter. In the other VM, we should trigger the ping command to the so called destination to make sure we receive the packets from the first one.

The below screenshots show how:



