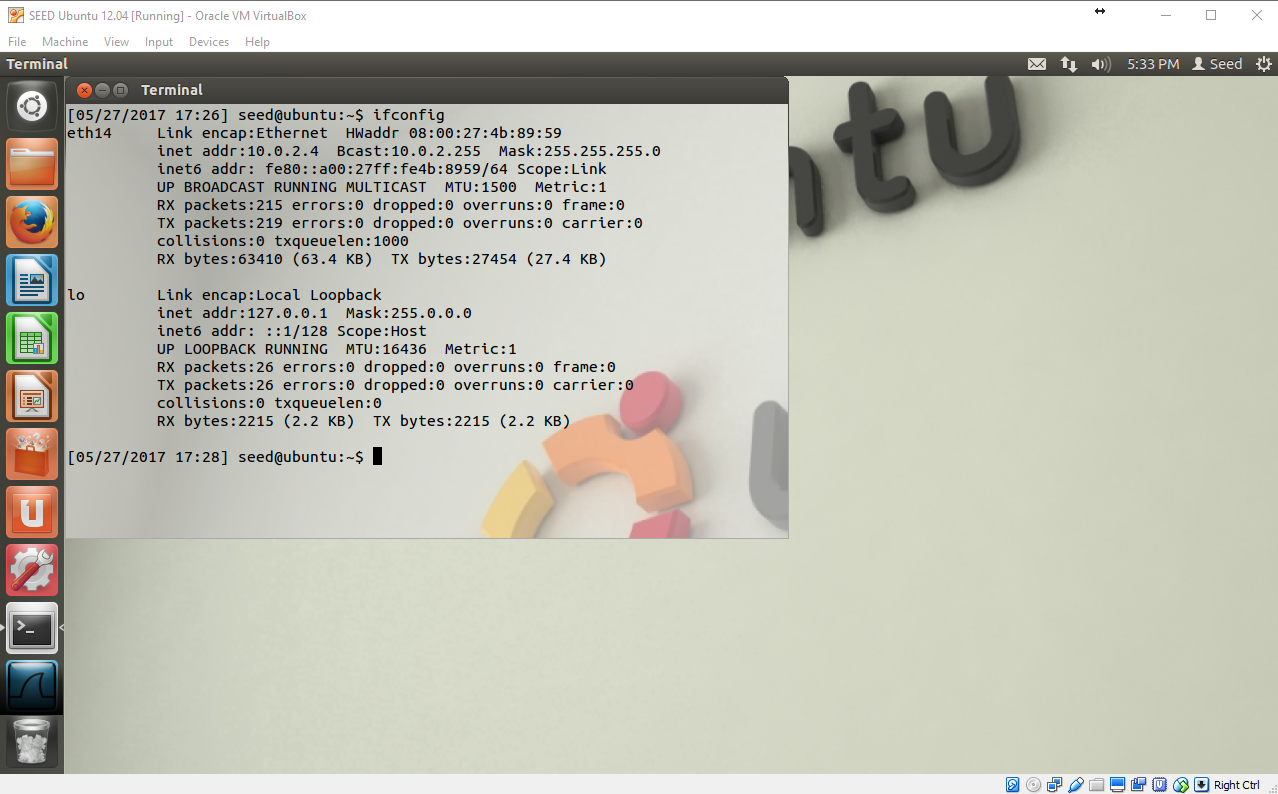
**Exercise 05**

**Name:** Jose Juan Sandoval

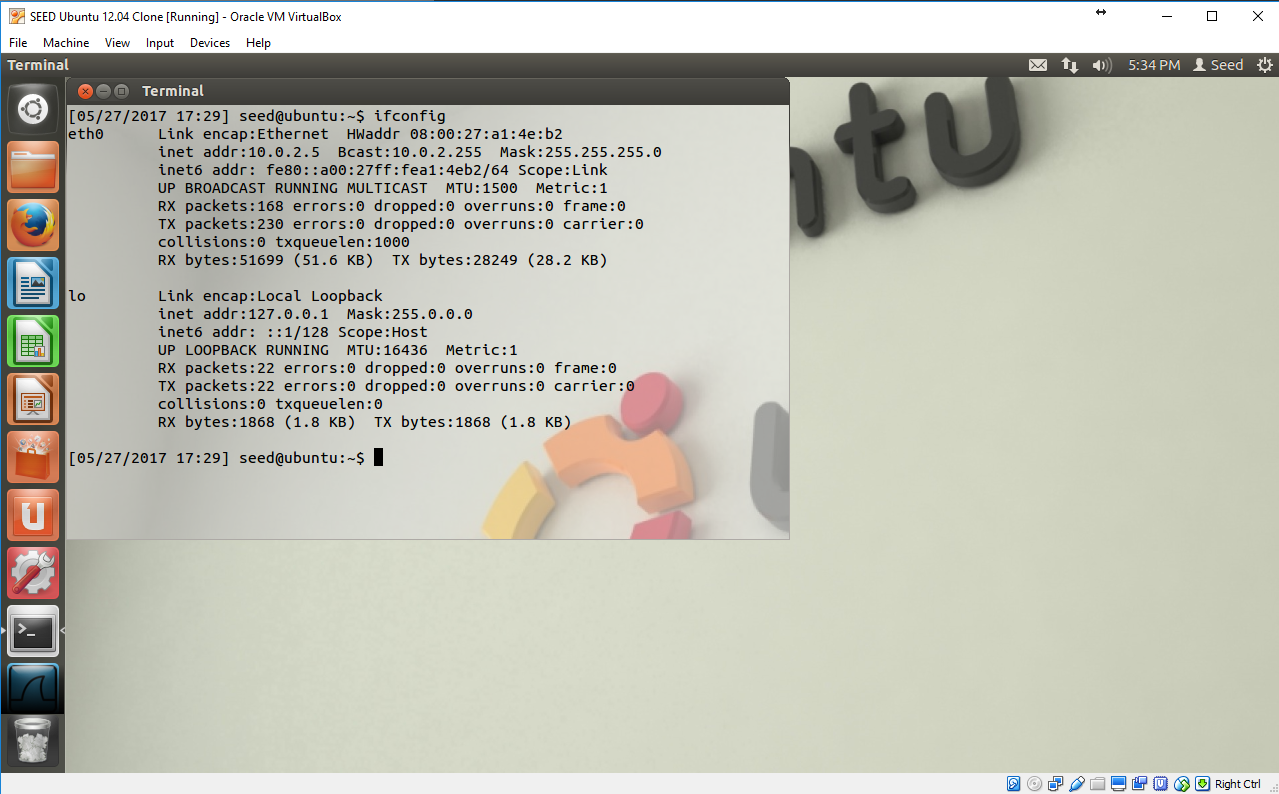
**Link to Project:** <https://github.com/Juanchiselo/CS380/tree/master/Exercises/Exercise%2005>

**Problem 1: Verifying the Network**

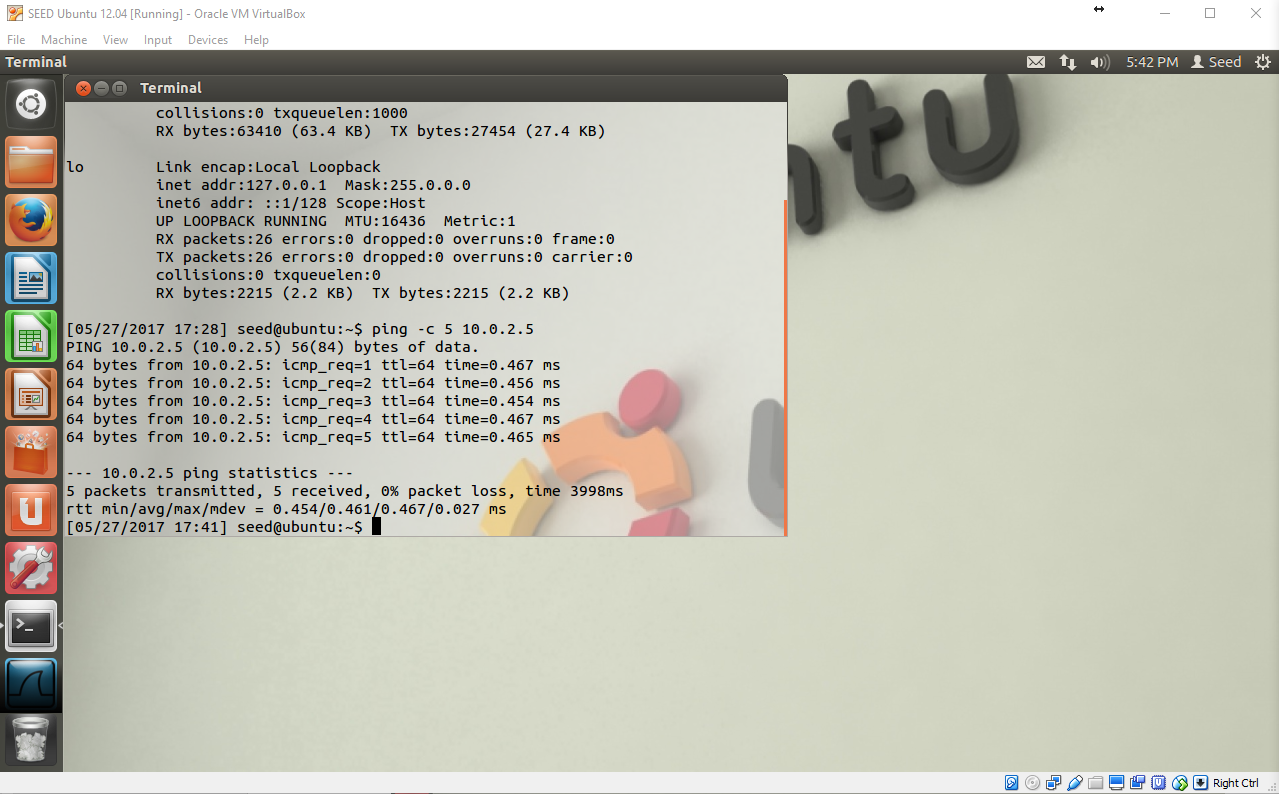
**VM1 –** Interface Name: eth14, Inet Address: 10.0.2.4



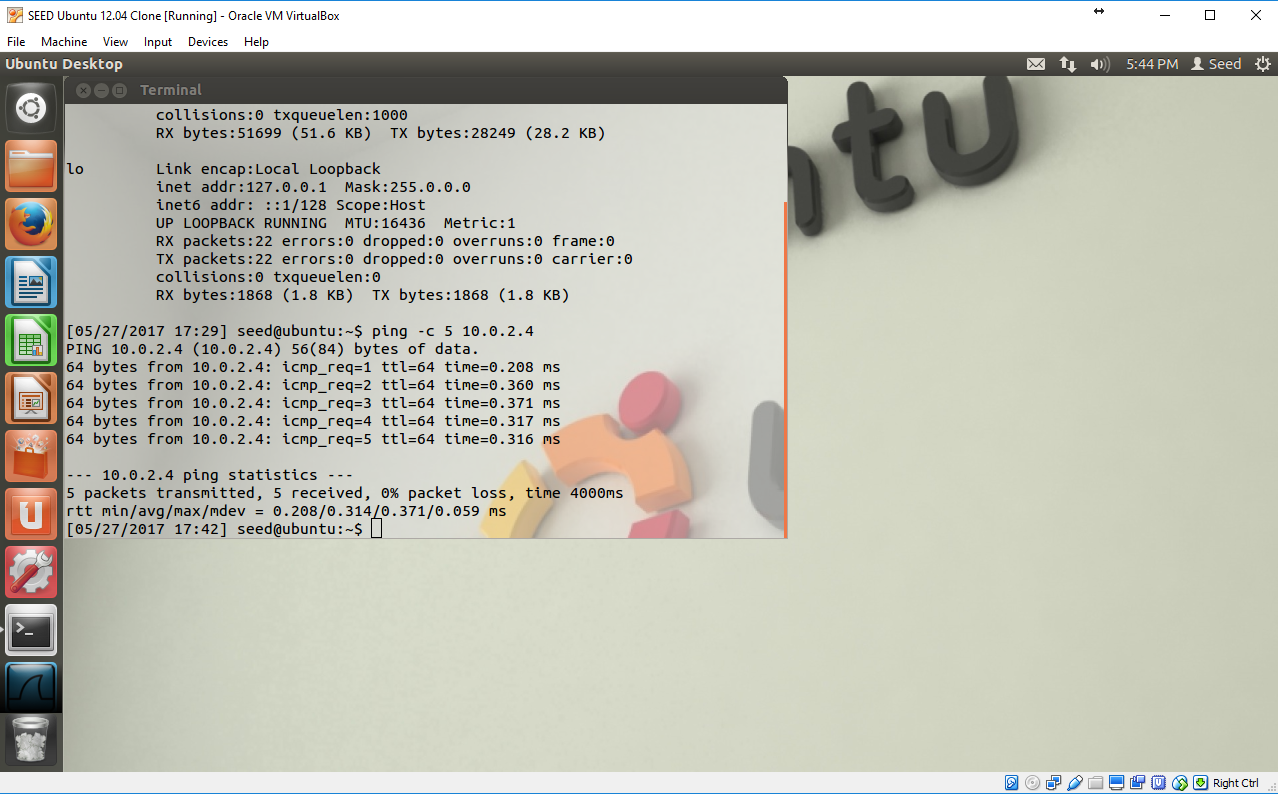
**VM2 –** Interface Name: eth0, Inet Address: 10.0.2.5



**VM1 -** Ping Command: ping –c 5 10.0.2.5



**VM2 –** Ping Command: ping –c 5 10.0.2.4



**Problem 2: Writing a Packet Sniffer**

To write our own packet sniffer we need to use the pcap library. The pcap library is used in our packet sniffer to:

1. Set the network interface that the packet sniffer will sniff on by using the pcap\_lookupdev function.
2. Open the device for sniffing by creating a sniffing session with the pcap\_open\_live function.
3. Determine the type of link-layer headers the device provides in order to process the packet contents later on. This determination is done by using the pcap\_datalink function.
4. Filter the packets we are sniffing on by setting and compiling a filter expression with the pcap\_setfilter and pcap\_compile functions respectively.
5. Capture the packets by using either the pcap\_next, pcap\_loop or pcap\_dispatch functions.
6. Read the packet contents by setting up a callback function that can parse the Ethernet, IP, TCP headers as well as the payload.

**Try running the packet sniffer using:**

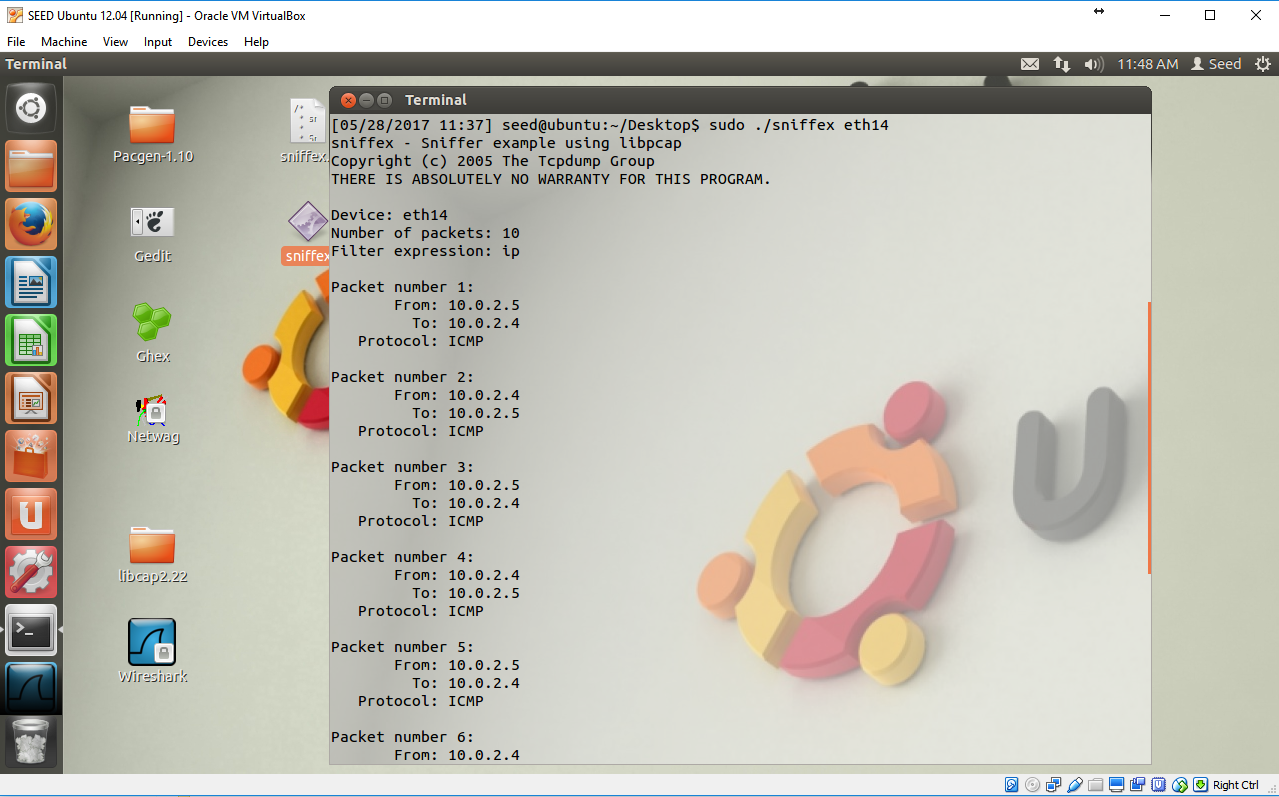
**$ ./sniffex ethX**

**where ethX is your network interface. What happens? Why does the program fail to start properly?**

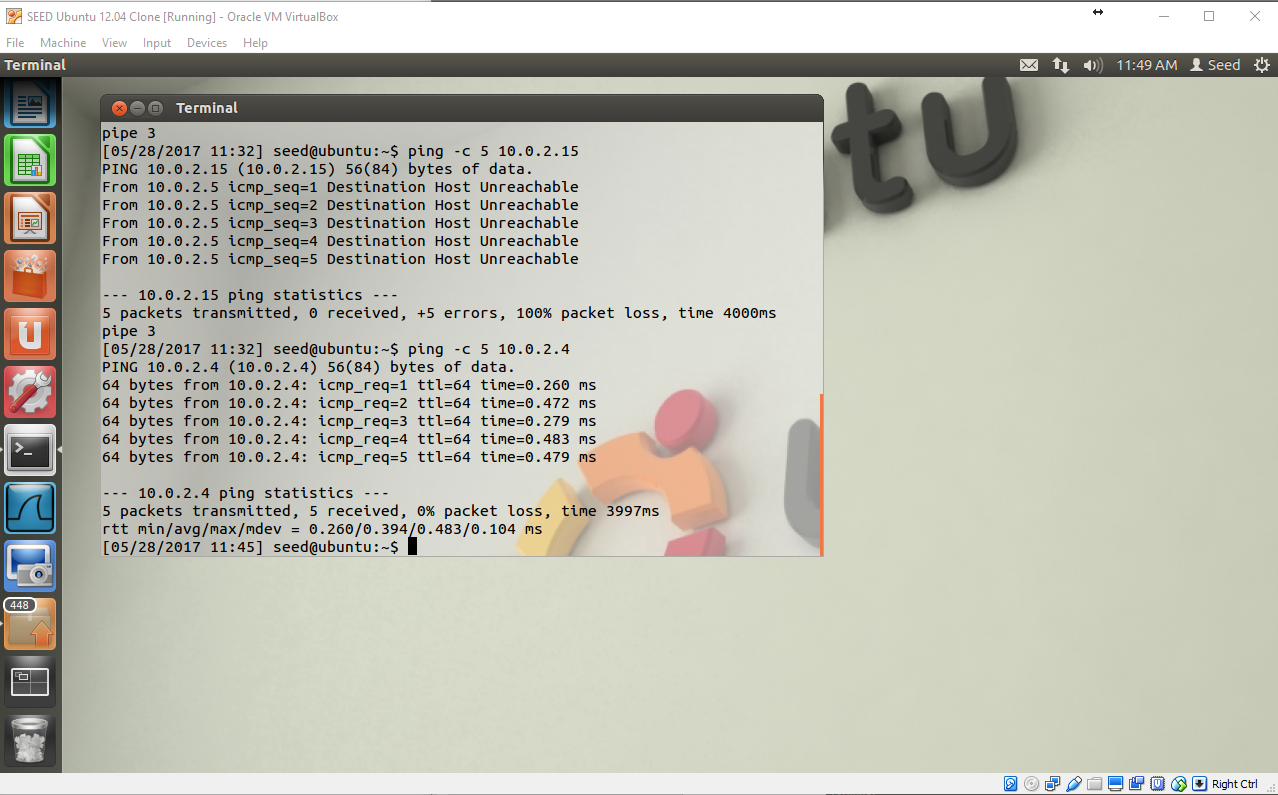
The packet sniffer failed to run because it couldn’t open the network interface for capturing due to not having the required permission to capture on that device.

**While your packet sniffer is running on one machine, ping that machine from the second one. Document the results.**

**VM1 – Sniffing received and sent ICMP packets.**



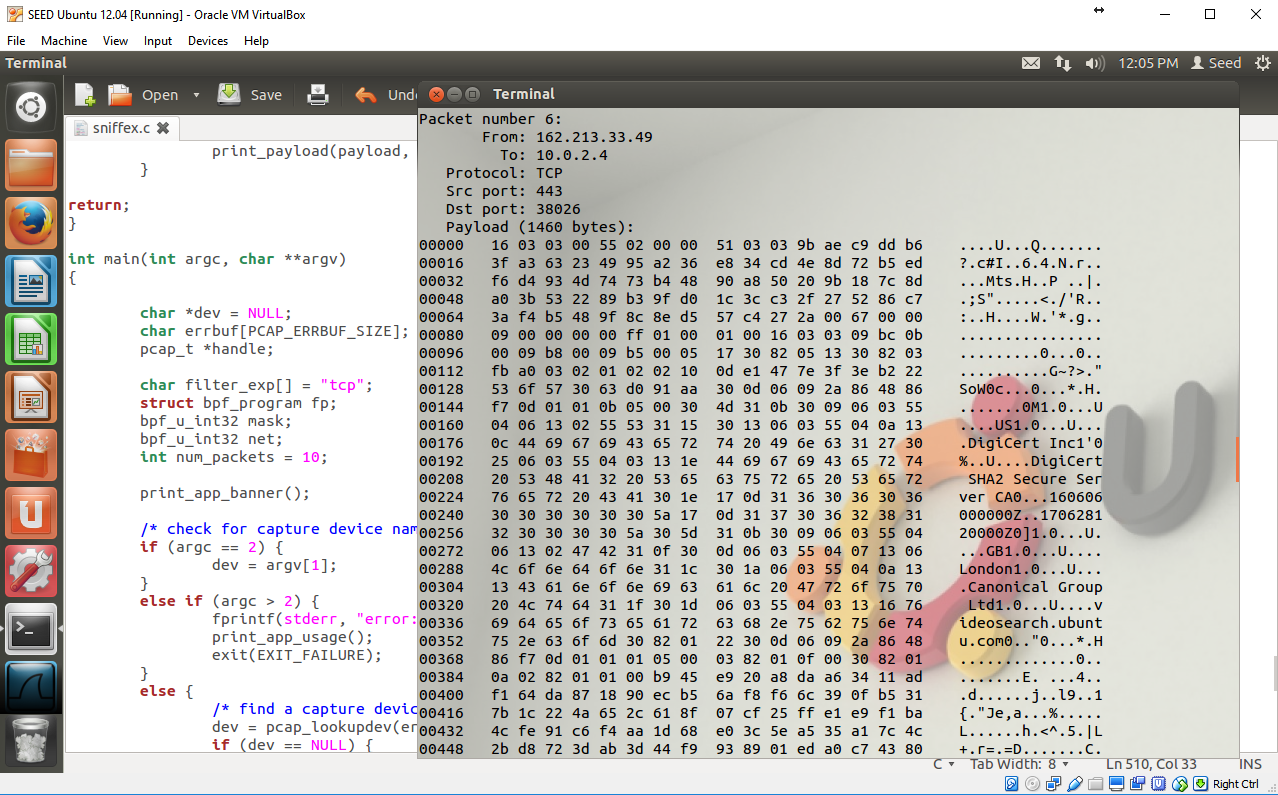
**VM2 – Running the Ping Command.**



**Now, modify the filter expression in your packet sniffer source code so that it only captures TCP packets. Redo the above experiment and document the results. What has changed?**

While the ping command was running on VM2 the packet sniffer on VM1 wasn’t capturing anything since the ping command sends ICMP packets and the packet sniffer was set to capture only TCP packets, but a few seconds the ping command was done, the packet sniffer started printing out some output.

The screenshot below shows some sample output that started printing out:



It appears the host was contacting a website with the digicert.com domain name.

**Problem 3: Password Sniffing**

**On the machine where you are running the packet sniffer, start the telnet service using:**

**$ sudo service openbsd-inetd start**

**Telnet is a service that allows a user to login to a system remotely.**

**After starting the telnet service, from the other VM run the following command:**

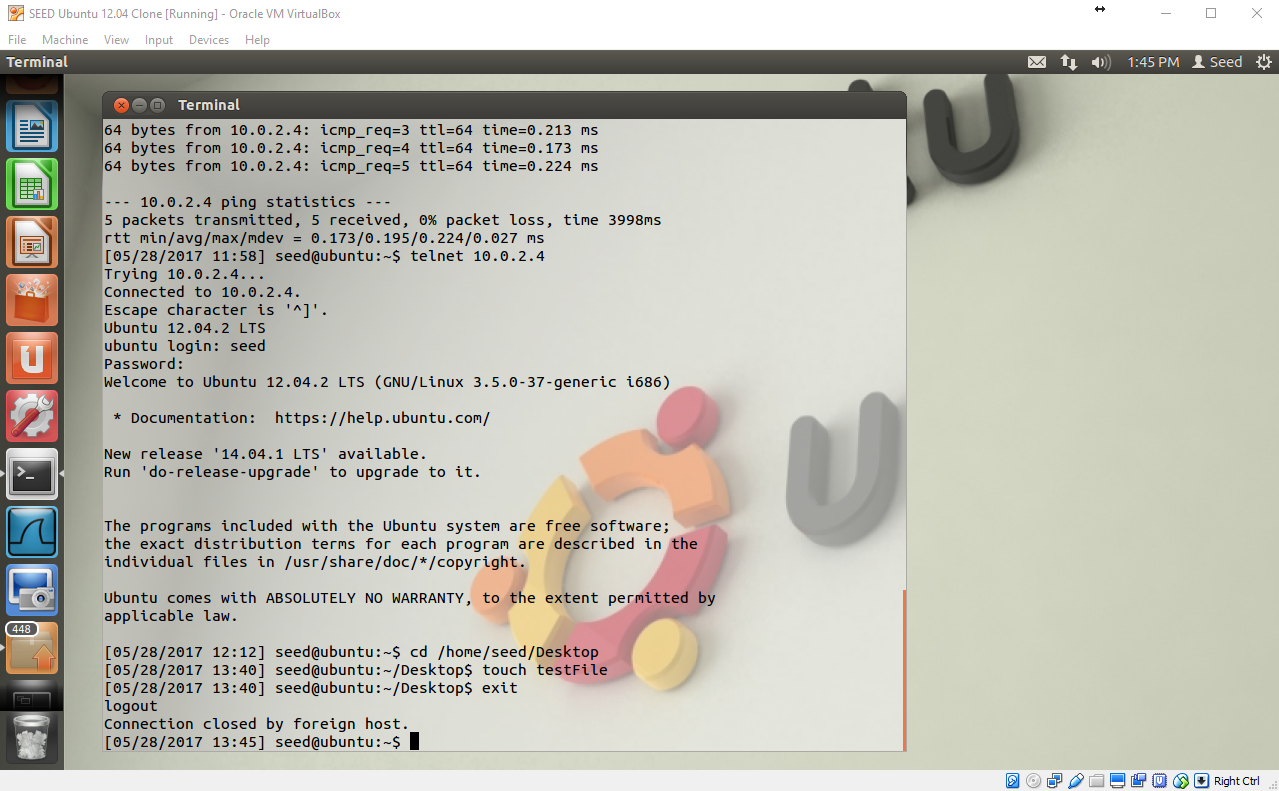
**$ telnet x.x.x.x**

**where x.x.x.x is the IP address of the machine running the telnet server. After some time, you should see a login prompt. Enter the login credentials of the user on the first VM and you will be logged in.**

**cd to the user's Desktop folder and create a new text file, then use the exit command to logout. From the first VM, verify that the text file has been created and document all of these actions.**

**VM1 – Starting Up the Telnet Service and Showing the testFile File Remotely Created by VM2.**



**VM2 – Remoting Into VM1 Using Telnet, Navigating to the Desktop of VM1, Creating the testFile File and Logging Out.**

**Now, on the machine running the telnet server start your packet sniffer. You may need to modify it to sniff more than just 10 packets. Then repeat the process above from the second VM. Pay attention to the packet contents in the packet sniffer and you should find the password of the user logging in. Document the results.**

**VM1 – Sniffing the TCP Packets Received and Sent Back to VM2.**

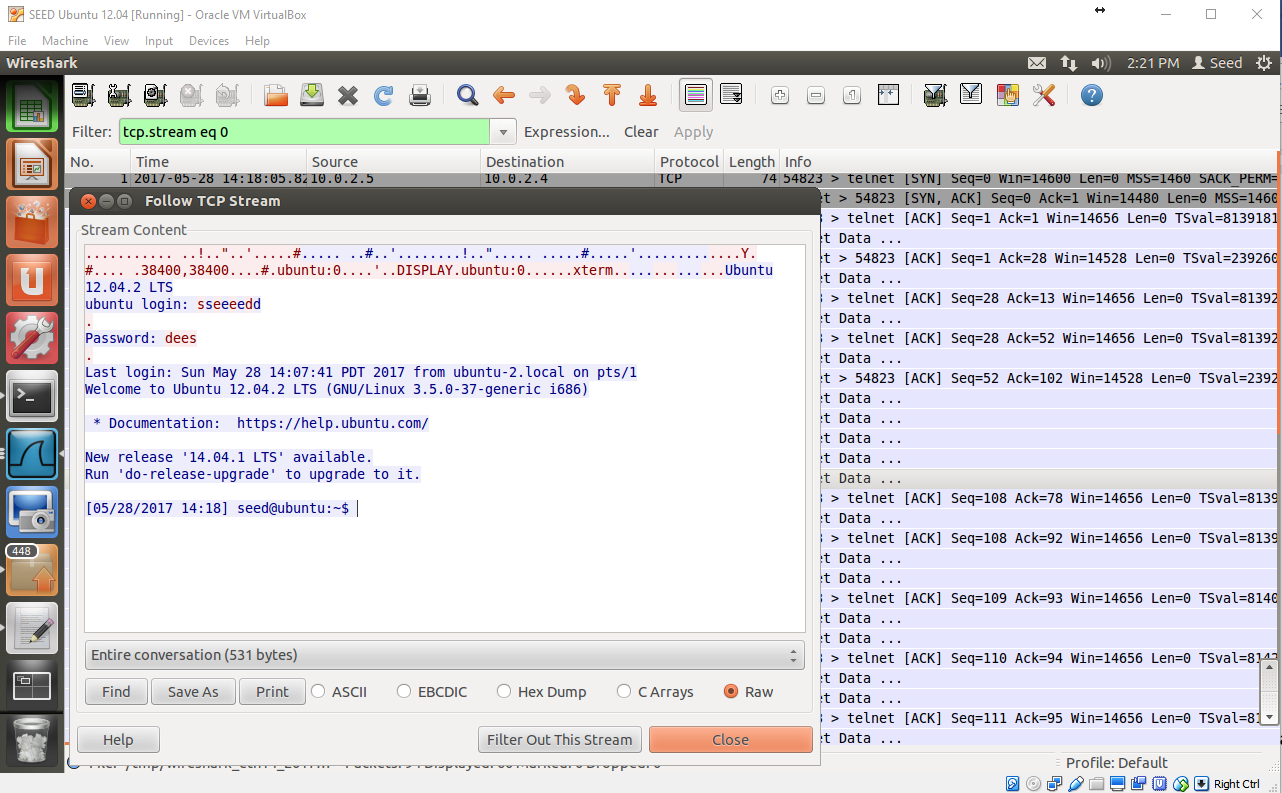


**VM2 – Remoting to VM1 and Entering Username and Password.**



What I noticed is that the packet sniffer on VM1 was able to capture the password in plaintext and all of the other strings that were sent back and forth between the machines.

**Finally, repeat the process again but use Wireshark instead of your own packet sniffer. Can you still locate the user's password? Document your results.**



Yes, I was able to locate the user’s password with Wireshark but only after I right clicked on a telnet packet and choose “Follow TCP Stream”. Before doing that I was very confused as to which packets to look at to see the single characters as they were shown in our own packet sniffer.

**After these experiments, discuss your thoughts about the security of using telnet as a method of remotely accessing a system**.

After these experiments, I don’t think it is very secure to use telnet as a method to remotely access a system because the password is sent in plaintext and it can be easily captured with a packet sniffing tool.

**Problem 4: SSH**

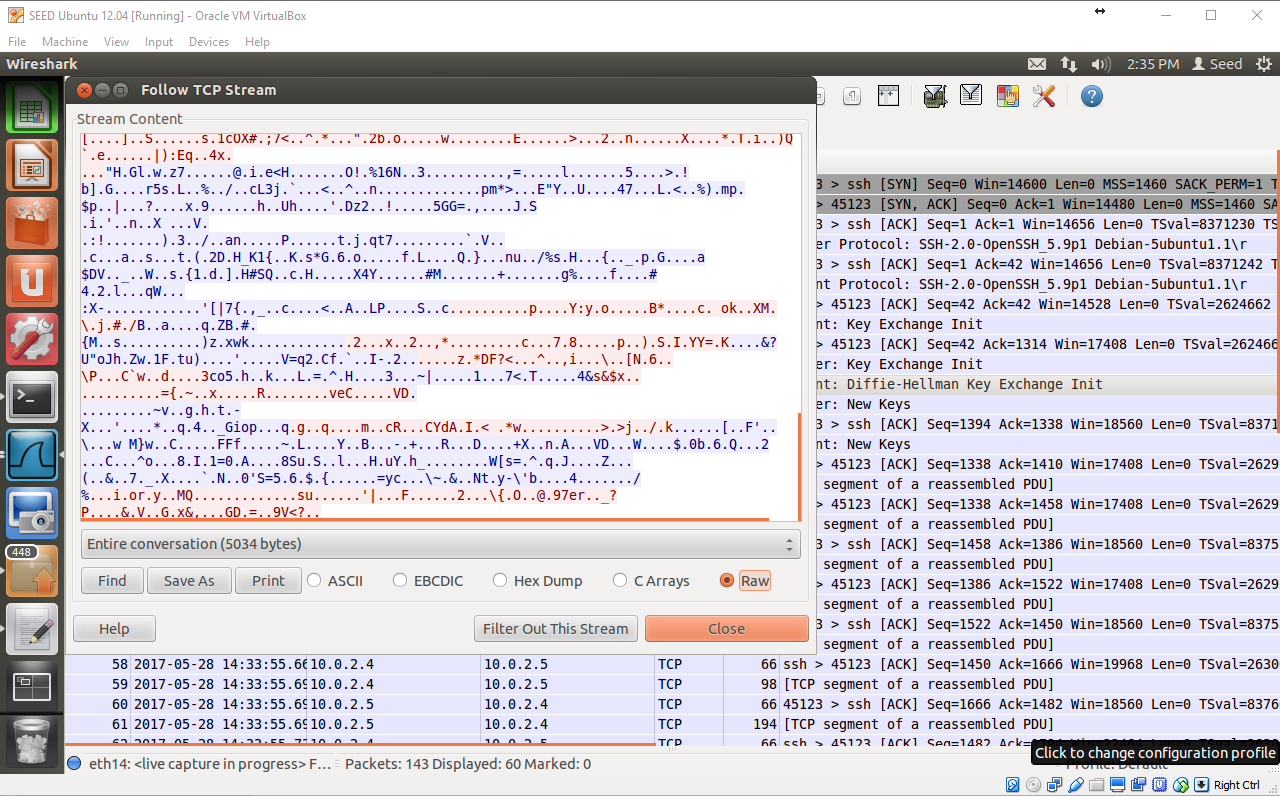
**From the machine where you were running your packet sniffer, have Wireshark listen on the network interface. From the other machine, we will now login using SSH (Secure SHell). SSH is like telnet in that it allows us to remotely login to a system. However, SSH was designed to be secure and uses encryption to prevent eavesdropping.**

**To login, use the following command:**

**$ ssh seed@x.x.x.x**

**where x.x.x.x is the IP address of the other VM (where Wireshark is running). You might get a warning**

**about an ECDSA key fingerprint, simply type yes and then Enter to continue. You should then be prompted for a password and finally be logged in to the system. After logging in successfully, you can exit. Once you have completed these steps, go back to Wireshark on the first VM and try to locate the SSH traffic. Can you find the user's password? Document the results from Wireshark.**



This time I could not find the user password because it is encrypted when using SSH. Looking at the output captured by Wireshark all I can see mumble jumble.