PROJECT 1: PACKET SNIFFING AND SPOOFING

SUBMITTED BY CHRISTIN WILSON

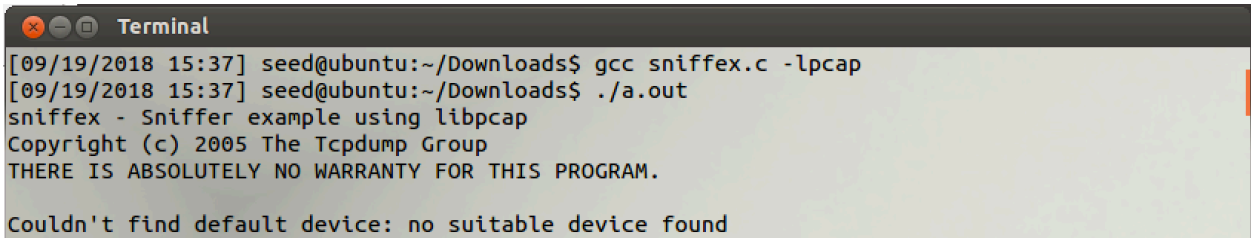
Problem 1: Please use your own words to describe the sequence of the library calls that are essential for sniffer programs. This is meant to be a summary, not detailed explanation like the one in the tutorial.

The sequence of library calls are:

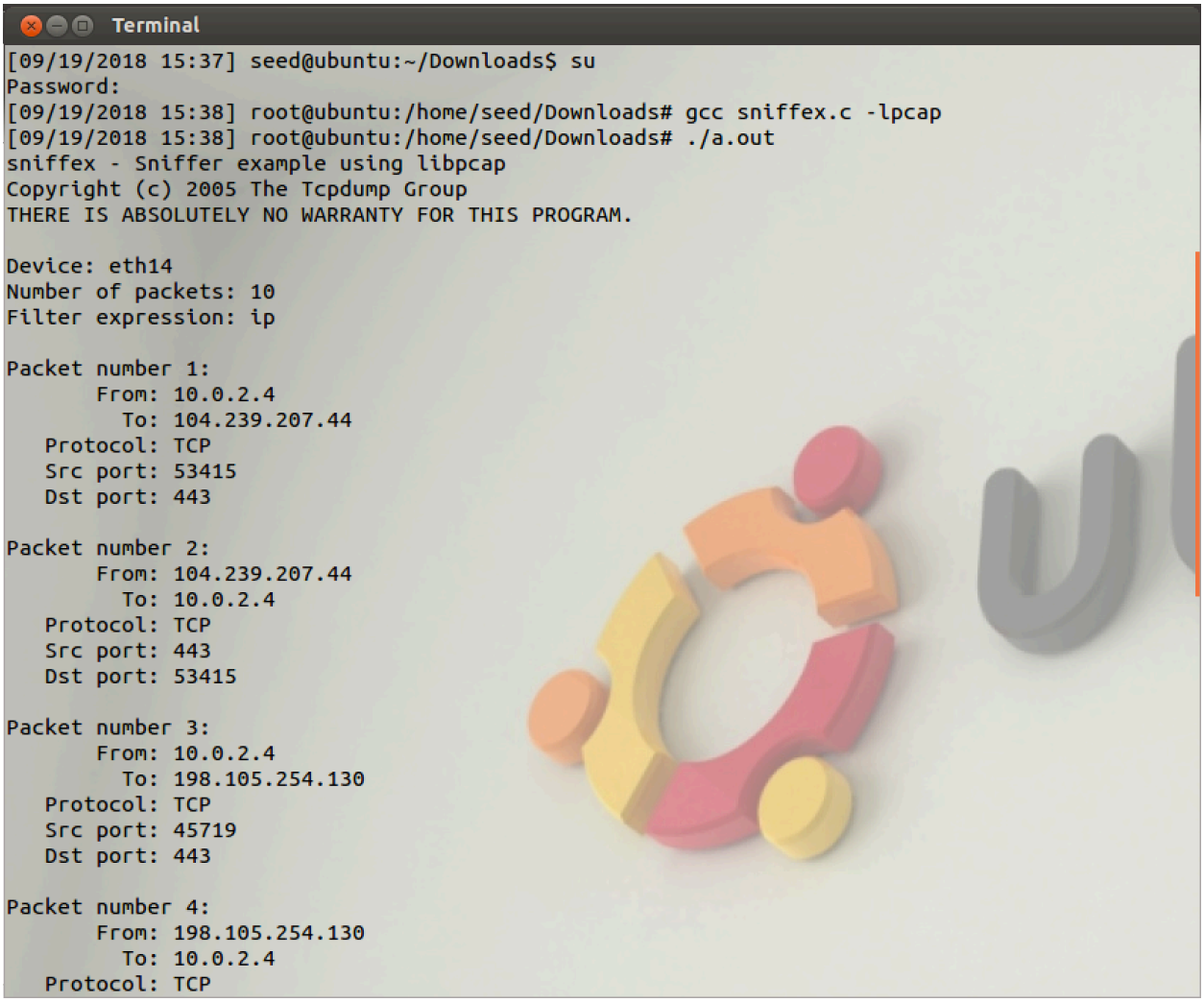
1) Setting up the device: The first step is to determine the interface to be sniffed on. Examples include eth0, x11 etc. It can be defined in a string in the code or by asking pcap to automatically pick an interface that will do the job.
2) Initialization: Sniffer tells pcap the device that we will be sniffing on as defined in the first step. Its possible to sniff on either one or multiple devices. These multiple devices are distinguished by naming the different sessions.
3) Filtering traffic: A rule set is created so that only special traffic is sniffed and it is compiled and applied. For example, if we want to sniff only HTTP traffic then we will sniff only the packets going to port 80. The rule set is kept in a string and is compiled. Pcap filters the traffic based on the rule set.
4) Execution: This is where the sniffer is finally executed. We tell pap to start the execution. Execution can be carried out in two techniques. In the first technique, a packet is sniffed and then analyzed whereas, in the second technique, we enter a loop until n packets are sniffed. Every time a packet is sniffed, a function that we have defined, will be called and it will perform some action on the packet.
5) Closing the session: After we are done with the requirements of sniffing, we close the session.

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

The **pcap_lookupdev()** function in sniffex.c needs root access to access the network interfaces. Network access isn't available without root access in Linux. If root access is not provided the results obtained is shown in the figure.



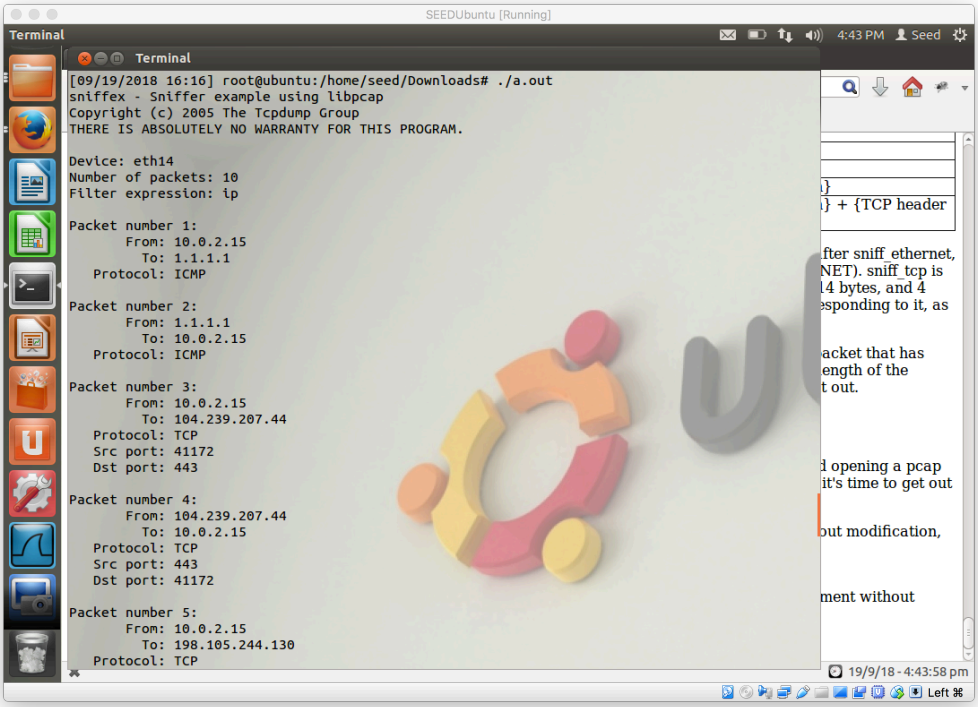
If root access is provided, the results obtained are:



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.

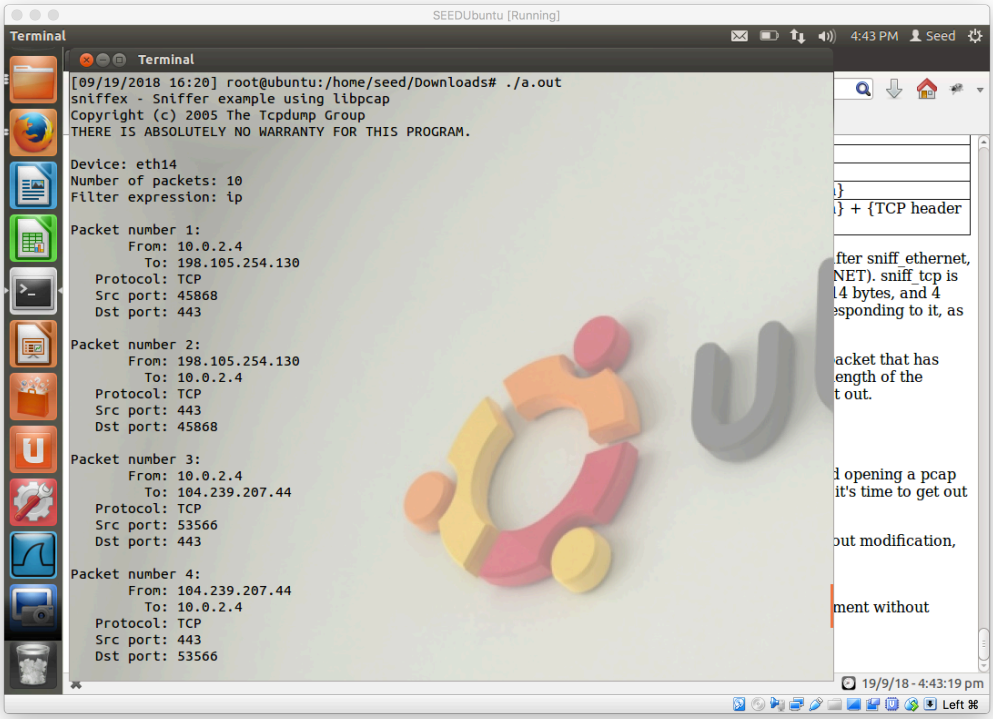
In non-promiscuous mode, the packets sniffed by the sniffer have to be addressed to it, whereas in promiscuous mode, The packets sniffed by the sniffer need not necessarily have to be addressed to it.

Demonstration: We have two VMs A&B. We run the sniffer program in A and we ping an IP address in B.
In promiscuous mode, the bit is set as 1 and A sniffs packets that are not addressed to it.
Code:
handle = pcap_open_live(dev, SNAP_LEN, **1**, 1000, errbuf);



In non-promiscuous mode, the bit is 0 and A sniffs packets that are only addressed to its Ip (10.0.2.4).

Code:
handle = pcap_open_live(dev, SNAP_LEN, **0**, 1000, errbuf);



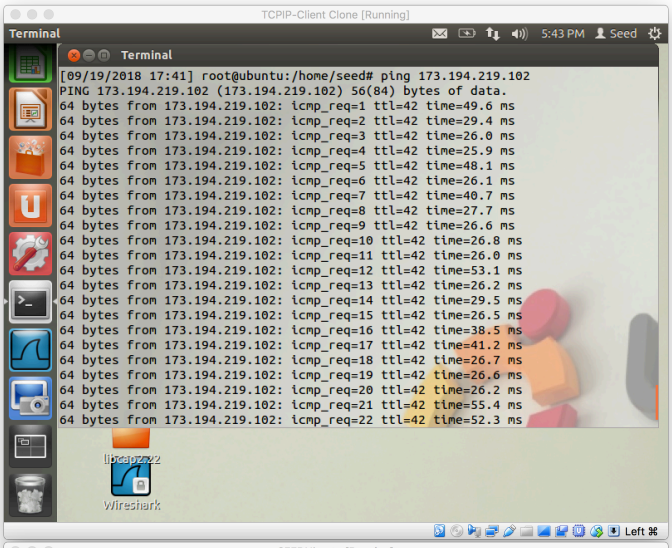
Problem 4: Please write filter expressions to capture each of the followings. In your lab reports, you need to include screendumps to show the results of applying each of these filters

- **Capture the ICMP packets between two specific hosts.**

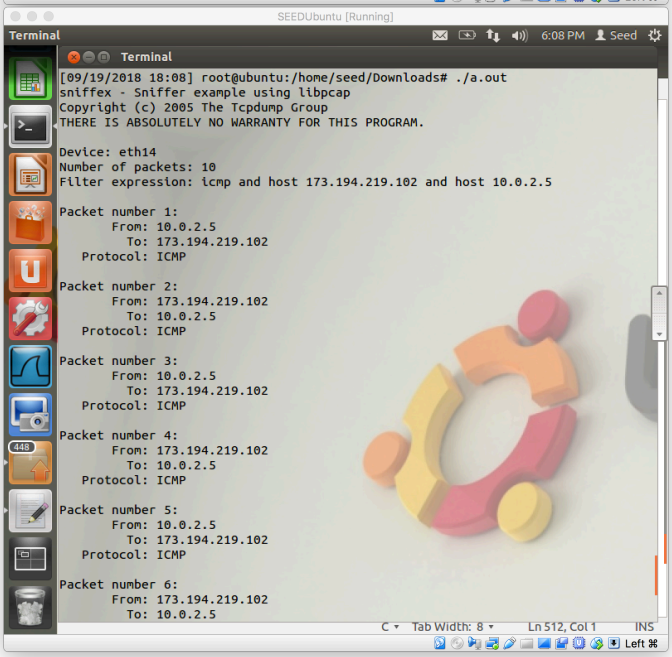
The IP address of the VM which was pinging the IP is 10.0.2.5.

Code used for filtering:
char filter_exp[] = “icmp and host 173.194.219.102 and host 10.0.2.5”;

VM B:



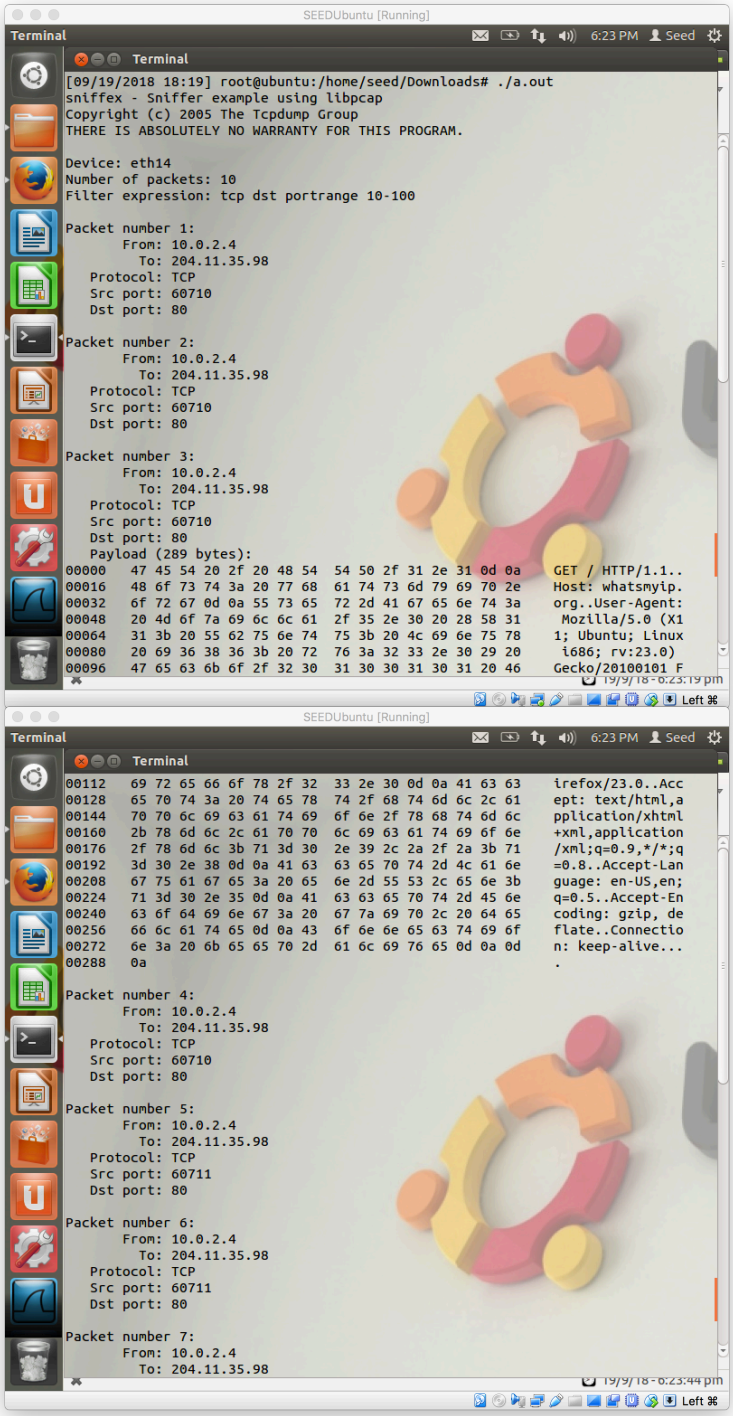
VM A:



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

Code used for filtering:
char filter_exp[] = "tcp dst portrange 10-100";

In order to sniff this, I used the web browser on the same VM and visited a page with uses HTTP which uses port 80 which is between 0-100. The sniffer sniffed a packet from 10.0.2.4 to 204.11.35.98 that was using port 80 as the destination port. To check for unacceptable ports, I visited google.com which uses HTTPS which uses port 443 and thus it was not sniffed.



Problem 6: Please use your own words to describe the sequence of the library calls that are essential for packet spoofing. This is meant to be a summary.

The four library calls for packet spoofing are:
1. Create a raw socket
2. Set socket option
3. Construct the packet
4. Send out the packet through the raw socket.

Create a raw socket:

A raw socket is a network socket used to send/receive raw packets. It can bypass the TCP/IP processing and directly send or receive the packet. The first step for spoofing is to create these raw sockets. These will be used to inject packets to the network. All the protocol headers will be filled by the program instead of by the kernel. There are two kinds of sockets that UNIX provides: SOCK_PACKET and SOCK_RAW.

Set Socket option:

Sockets are transport layer specific even though it is an interface to the IP header. We have to create 3 separate raw sockets, using IPPROTO_TCP, IPPROTO_UDP and IPPROTO_ICMP for listening to TCP, UDP and ICMP traffic.

Construct the packet:

Since we are entering our own packets, we need to know the structure of the protocols that need to be included. Packets should have headers for all the protocols that need to be included before injecting into the network for spoofing. The protocols can be network layer (IP) or transport layer (UDP or TCP).

Send out the packet through the raw packet:

With the information regarding the structure of the protocol headers and programming knowledge, we create the packets and inject it into the network. Certain parameters are defined which will indicate that this packet was created for spoofing.

Problem 7: 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?

For spoofing, the user creates packets with headers in which all the data is entered by the user itself. This is possible only when raw sockets are used. To receive the raw packets on an interface, we need root privilege. This is a security precaution. Also the packets that a normal user sends isn't customizable to that level. The OS will set some fields including the protocol header fields. So if we execute the program without root privilege we wont be able to create the raw packets and inject it using the raw sockets.