Task 1.a

**Problem 1:**

The general idea of the sniffer programs are to first

Set the Device -> Open the device for sniffing -> Sniff -> Filter traffic being sniffed by the device

**Problem 2:**

The sniffer program needs root privileges because it is accessing a low-level OS system in the machine. Without this commands like pcap\_lookupdev() and pcap\_open\_live() will fail due to security issues. More so, the sniffing program is calling a system API which will always require root privileges.

**Problem 3:**

Promiscuous to 1: The output can sniff packets that are not being sent to the IP with the sniffer. (i.e can sniff the “air”)

Promiscuous to 0: The output will only print things that are being sent to the IP with the sniffer turned on.

Task 1.b

Capturing ICMP

IP of sniffer: 192.168.15.5

IP of hosts: 192.168.15.5 192.168.15.6

Filter Expression: “(icmp) and ((net 192.168.15.5) or (net 192.168.15.6))”

`A screenshot of a cell phone

Description automatically generated

.6 pinging .5

A screenshot of a cell phone

Description automatically generated

.5 pinging .6

Capturing TCP packets ports 50-100

IP of sniffer: 192.168.15.5

IP of hosts: 192.168.15.5 192.168.15.6

Filter Expression: “(tcp) and (dst portrange 50-100)”

A screenshot of a cell phone

Description automatically generated

.6 pinging .5 on ports 40,50,60,70,80

A close up of a logo

Description automatically generated

Netcat command used to ping .5

Task 1.c

Capturing passwords on Telnet

Telnet server: 192.168.15.5

Testnet client: 192.168.15.6

Filter Expression: “port 23” (23 is the port that telnet uses)

A screenshot of a cell phone

Description automatically generated

.6 connecting to .5 via telnet and entering username + password

A screenshot of a cell phone

Description automatically generated

Password figure one (cs5)

A screenshot of a cell phone

Description automatically generated

Password figure 2 (28pa)

A screenshot of a cell phone

Description automatically generated

Password figure 3 (ss)

Task 2.a:

Compile spoof.c : “gcc -o spoof spoof.c”

Run spoof

“sudo ./spoof 0” ICMP

“sudo ./spoof 1” Ethernet

Task 2.b:

Spoofing machine: 192.168.15.5

Spoofed machine: 192.168.15.10

Destination machine: 192.168.15.6

Tcpdump machine: 192.168.15.6

The reason it says cs528vm-2.local is because I am sending the spoofed message to the IP running tcpdump. If I sent it to something like 8.8.4.4 you would see “192.168.15.10 > 8.8.4.4”

A screenshot of text

Description automatically generated

Task 2.c:

Spoofing machine: 192.168.15.5

Spoofed machine: 192.168.15.10

Destination machine: 192.168.15.6

MAC address: 01:02:03:04:05:06

Tcpdump machine: 192.168.15.6

The reason it says “01:02:03:04:05:06 > Broadcast is because I am making the dst\_mac be ff:ff:ff:ff:ff:ff. aka broadcast. If I changed this to something like 1f:23:0f:1f:3f:5f you would see that instead of broadcast. I also explained why it says 192.168.15.10 > cs528vm-2.local above.

A screenshot of text

Description automatically generated

Task 2.Question.1:

No, the length of the IP packet length field must be the size of the packet, sendto() will throw a error if this is not the case.

Task 2.Question.2:

No, the system will automatically fill this in for us.

Task 2.Question.3:

We need root privilages because when we try to make a socket, “sd = socket(AF\_INET, SOCK\_RAW, IPPROTO\_RAW);” the system will throw an error saying “Permission Denied”. If you could create sockets without root, programs could spoof different address and build their own packets causing a huge security problem.

Task 2.Question.4:

First create a raw socket, then add the flags as you see necessary. Then create our packet structs and fill them in with values that you want. Use the sendto() command to send your custom packet to whatever destination IP you would like. This all must be done under root as explained in Task 2.Question.3.