Packet Sniffing and Spoofing Lab

CS5460 Assignment 7 Due: December 8 2017

Austin Derbique A01967241

2.1 Task 1: Writing Sniffing Program

Problem 1: Inside of sniffex.c are many different functions. The most important, however are the pcap library calls.

On Line 529: pcap_lookupdev: Finds default capture device

On Line 538: pcap_lookupnet: Gets net mask for device

On Line 551: pcap_open_live: Opens device and begins sniffing

On Line 558: pcap_datalink: Returns type of device that capture is happening on

On Line 564: pcap_compile: Creates filter of expression and parses filter

On Line 571: pcap_setfilter: Sets and installs filter

On Line 578: pcacp_loop: Loops for filtering packets

On Line 580: pcap_freecode: frees up code memory

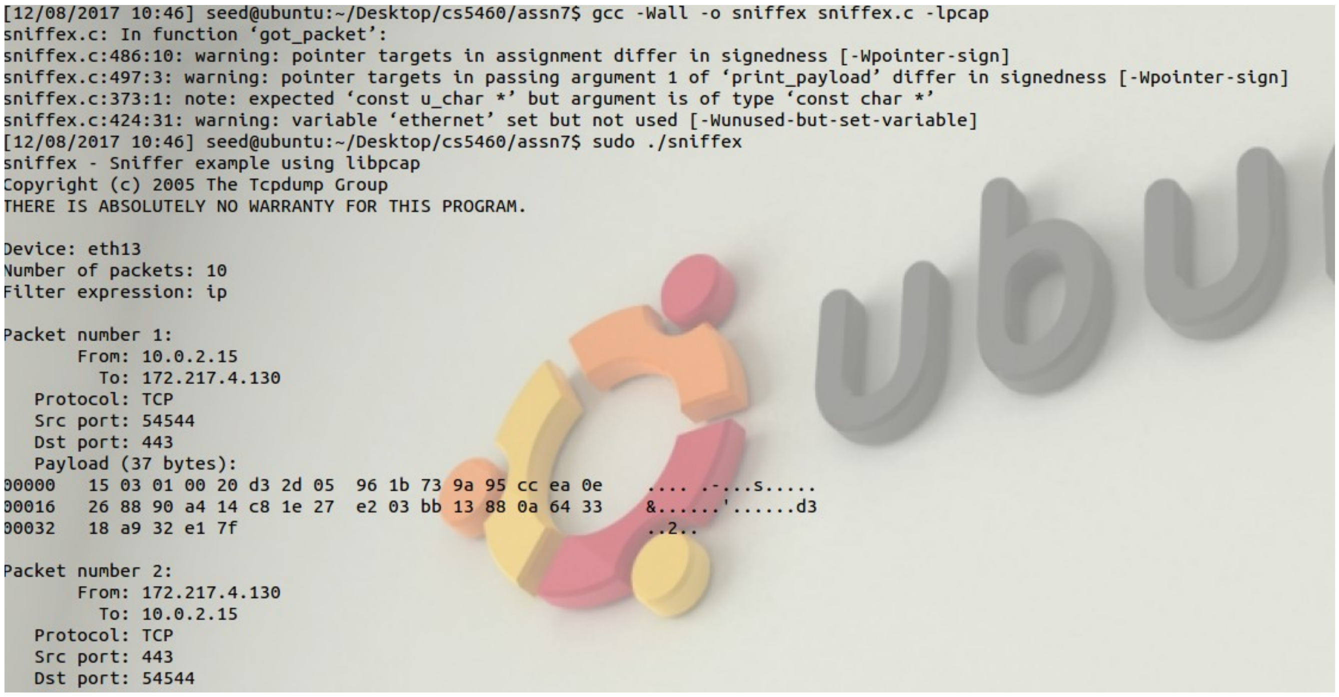
On Line 581: pcpc_close: closes handle and session

Problem 2: You need root privilege in order to have full access to looking at network traffic. This can be promiscuous if you are not authorized to have root access. It is likely to fail when the program searches for the device. This is in the pcap_lookupdev function. Finding the capture device.

Problem 3: Promiscuous mode seems to show all the traffic not only going to the network controller but data flying past as well. When turned off, the only data seen is the data intended for your network controller This I found on line 551 of sniffex.c with the boolean expression to turn promiscuous mode on or off.

Task1.a: Understanding Sniffex. Downloading the sniffex.c file from:
http://www.tcpdump.org/sniffex.c
Command used to compile: gcc -Wall -o sniffex sniffex.c -lpcap
Command used to execute: sudo ./sniffex

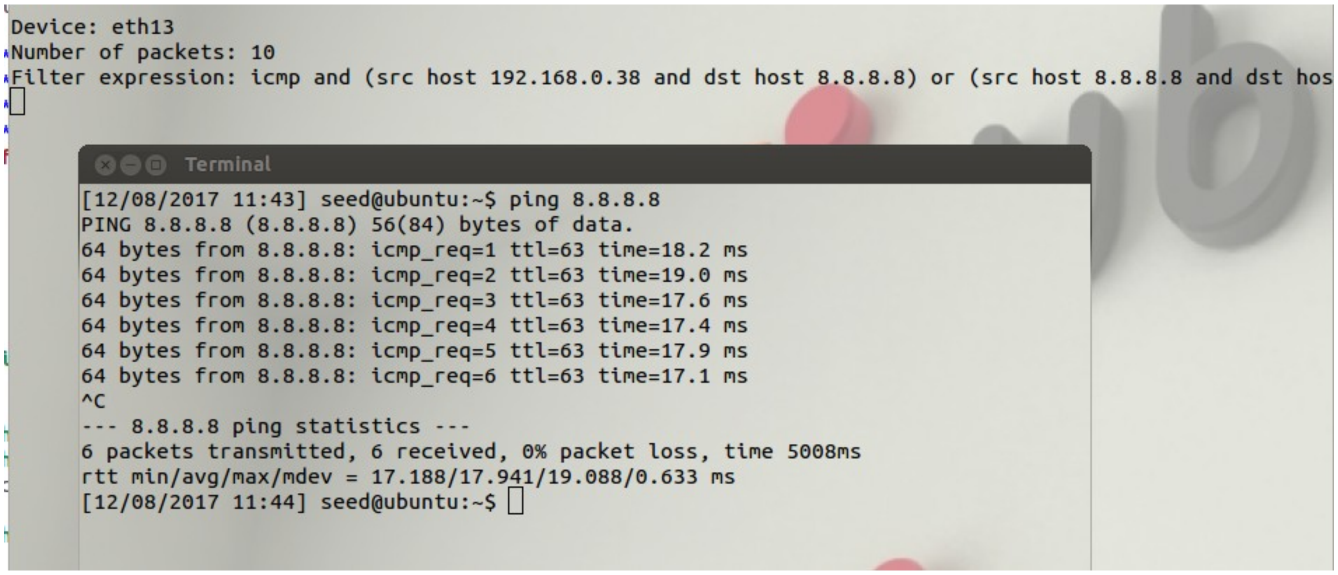
Output: Addiontally, piped txt output available in part2/task 1/sniffex_output.txt



```
[12/08/2017 10:46] seed@ubuntu:~/Desktop/cs5460/assn7$ gcc -Wall -o sniffex sniffex.c -lpcap
sniffex.c: In function ‘got_packet’:
sniffex.c:486:10: warning: pointer targets in assignment differ in signedness [-Wpointer-sign]
sniffex.c:497:3: warning: pointer targets in passing argument 1 of ‘print_payload’ differ in signedness [-Wpointer-sign]
sniffex.c:373:1: note: expected ‘const u_char *’ but argument is of type ‘const char *’
sniffex.c:424:31: warning: variable ‘ethernet’ set but not used [-Wunused-but-set-variable]
[12/08/2017 10:46] seed@ubuntu:~/Desktop/cs5460/assn7$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth13
Number of packets: 10
Filter expression: ip

Packet number 1:
       From: 10.0.2.15
         To: 172.217.4.130
   Protocol: TCP
   Src port: 54544
   Dst port: 443
   Payload (37 bytes):
00000   15 03 01 00 20 d3 2d 05  96 1b 73 9a 95 cc ea 0e    .... .-...s.....
00016   26 88 90 a4 14 c8 1e 27  e2 03 bb 13 88 0a 64 33    &......'......d3
00032   18 a9 32 e1 7f                                      ..2..

Packet number 2:
       From: 172.217.4.130
         To: 10.0.2.15
   Protocol: TCP
   Src port: 443
   Dst port: 54544
```

Task 1.b: Writing Filters. Using the help page on **http://www.tcpdump.org/manpages/pcap-filter.7.html**,

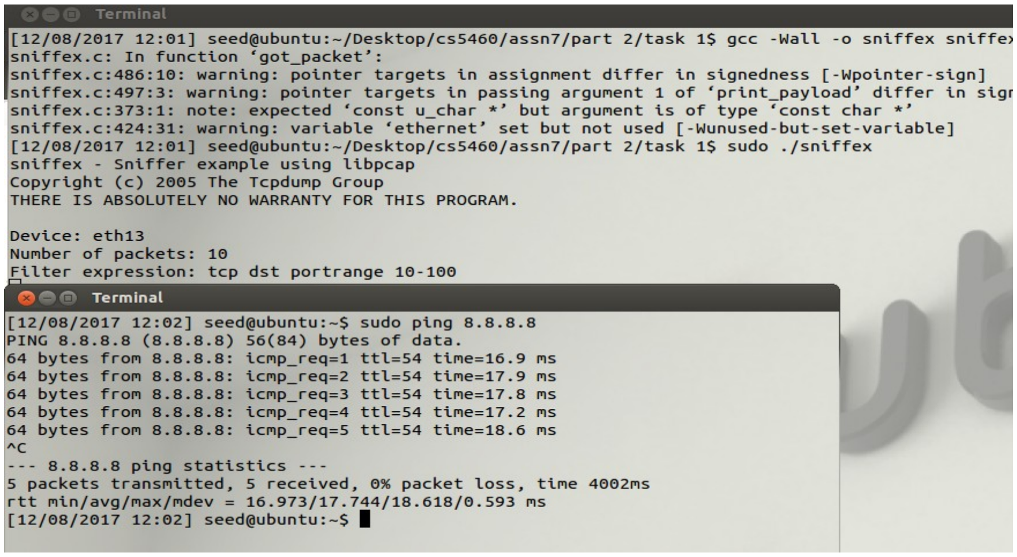
Capturing ICMP Packets between two Hosts:



```
Device: eth13
Number of packets: 10
Filter expression: icmp and (src host 192.168.0.38 and dst host 8.8.8.8) or (src host 8.8.8.8 and dst hos
```

```
[12/08/2017 11:43] seed@ubuntu:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=63 time=18.2 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=63 time=19.0 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=63 time=17.6 ms
64 bytes from 8.8.8.8: icmp_req=4 ttl=63 time=17.4 ms
64 bytes from 8.8.8.8: icmp_req=5 ttl=63 time=17.9 ms
64 bytes from 8.8.8.8: icmp_req=6 ttl=63 time=17.1 ms
^C
--- 8.8.8.8 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 17.188/17.941/19.088/0.633 ms
[12/08/2017 11:44] seed@ubuntu:~$
```

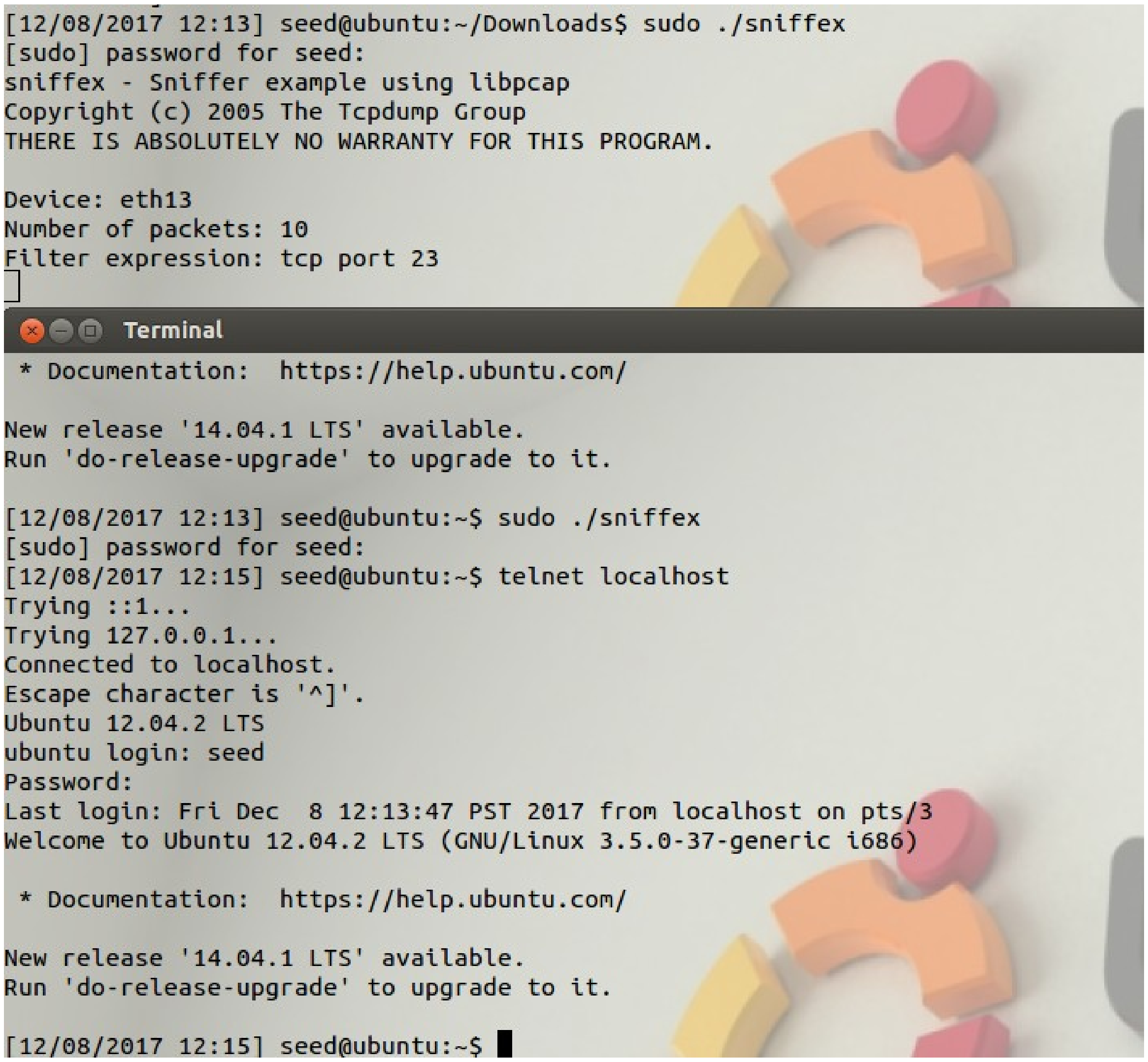
Capturing TCP packets that have destination port range from 10-100
Filter used: char filter_exp[] = "tcp dst portrange 10-100";



```
[12/08/2017 12:01] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 1$ gcc -Wall -o sniffex sniffex
sniffex.c: In function ‘got_packet’:
sniffex.c:486:10: warning: pointer targets in assignment differ in signedness [-Wpointer-sign]
sniffex.c:497:3: warning: pointer targets in passing argument 1 of ‘print_payload’ differ in sign
sniffex.c:373:1: note: expected ‘const u_char *’ but argument is of type ‘const char *’
sniffex.c:424:31: warning: variable ‘ethernet’ set but not used [-Wunused-but-set-variable]
[12/08/2017 12:01] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 1$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth13
Number of packets: 10
Filter expression: tcp dst portrange 10-100
```

```
[12/08/2017 12:02] seed@ubuntu:~$ sudo ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=54 time=16.9 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=54 time=17.9 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=54 time=17.8 ms
64 bytes from 8.8.8.8: icmp_req=4 ttl=54 time=17.2 ms
64 bytes from 8.8.8.8: icmp_req=5 ttl=54 time=18.6 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 16.973/17.744/18.618/0.593 ms
[12/08/2017 12:02] seed@ubuntu:~$
```

Task 1.c: Sniffing Passwords. Filter used: char filter_exp[] = "tcp port 23";

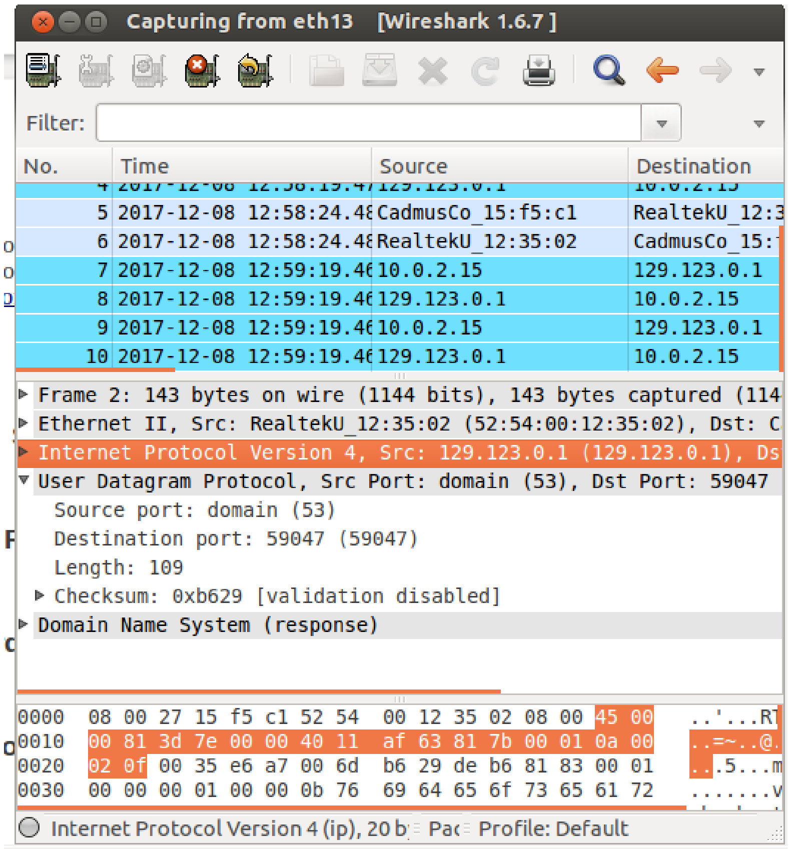


```
[12/08/2017 12:13] seed@ubuntu:~/Downloads$ sudo ./sniffex
[sudo] password for seed:
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth13
Number of packets: 10
Filter expression: tcp port 23
```

```
 * Documentation:  https://help.ubuntu.com/

New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

[12/08/2017 12:13] seed@ubuntu:~$ sudo ./sniffex
[sudo] password for seed:
[12/08/2017 12:15] seed@ubuntu:~$ telnet localhost
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Ubuntu 12.04.2 LTS
ubuntu login: seed
Password:
Last login: Fri Dec  8 12:13:47 PST 2017 from localhost on pts/3
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.5.0-37-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

[12/08/2017 12:15] seed@ubuntu:~$
```

2.2 Task 2: Spoofing

Fot Task 2.a, Writing a spoofing program, the assignment description says a packet program can be downloaded. For this, I found the following code: https://github.com/andrewjkerr/seed-packet-sniffing/blob/master/src/part-2/spoofer.c . Using wireshark,

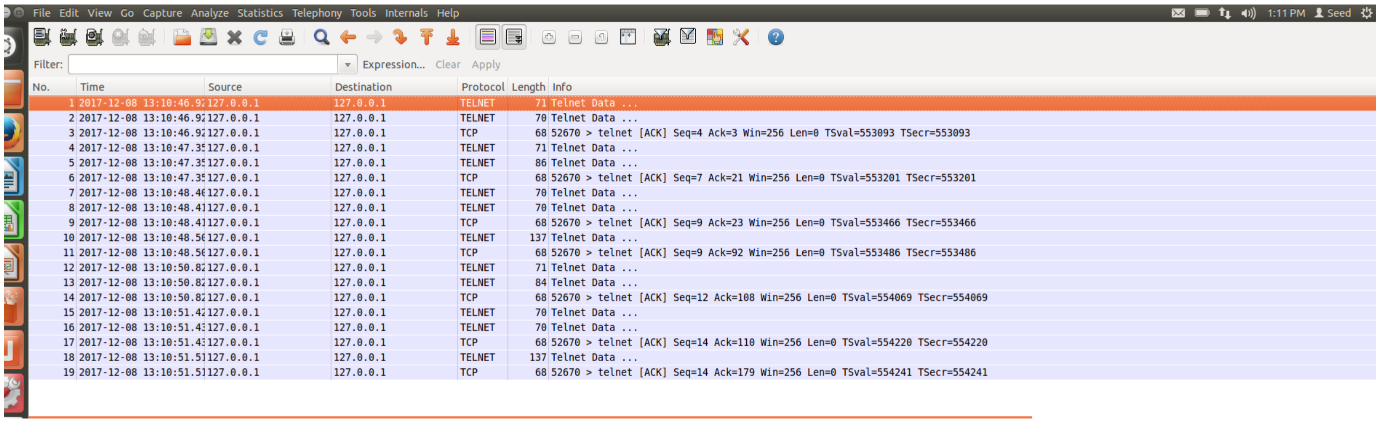


No.	Time	Source	Destination
4	2017-12-08 12:58:19.47	129.123.0.1	10.0.2.15
5	2017-12-08 12:58:24.48	CadmusCo_15:f5:c1	RealtekU_12:3
6	2017-12-08 12:58:24.48	RealtekU_12:35:02	CadmusCo_15:
7	2017-12-08 12:59:19.46	10.0.2.15	129.123.0.1
8	2017-12-08 12:59:19.46	129.123.0.1	10.0.2.15
9	2017-12-08 12:59:19.46	10.0.2.15	129.123.0.1
10	2017-12-08 12:59:19.46	129.123.0.1	10.0.2.15

```
▶ Frame 2: 143 bytes on wire (1144 bits), 143 bytes captured (114
▶ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: C
▶ Internet Protocol Version 4, Src: 129.123.0.1 (129.123.0.1), Ds
▼ User Datagram Protocol, Src Port: domain (53), Dst Port: 59047
    Source port: domain (53)
    Destination port: 59047 (59047)
    Length: 109
  ▶ Checksum: 0xb629 [validation disabled]
▶ Domain Name System (response)
```

```
0000  08 00 27 15 f5 c1 52 54  00 12 35 02 08 00 45 00   ..'...RT
0010  00 81 3d 7e 00 00 40 11  af 63 81 7b 00 01 0a 00   ..=~..@.
0020  02 0f 00 35 e6 a7 00 6d  b6 29 de b6 81 83 00 01   ...5...m
0030  00 00 00 01 00 00 0b 76  69 64 65 6f 73 65 61 72   .......v
```

Task 2.b Spoof an ICMP Echo Request.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
1:11 PM Seed
Filter:
Expression...
Clear
Apply

No.	Time	Source	Destination	Protocol	Length	Info
1	2017-12-08 13:10:46.92	127.0.0.1	127.0.0.1	TELNET	71	Telnet Data ...
2	2017-12-08 13:10:46.92	127.0.0.1	127.0.0.1	TELNET	70	Telnet Data ...
3	2017-12-08 13:10:46.92	127.0.0.1	127.0.0.1	TCP	68	52670 > telnet [ACK] Seq=4 Ack=3 Win=256 Len=0 TSval=553093 TSecr=553093
4	2017-12-08 13:10:47.35	127.0.0.1	127.0.0.1	TELNET	71	Telnet Data ...
5	2017-12-08 13:10:47.35	127.0.0.1	127.0.0.1	TELNET	86	Telnet Data ...
6	2017-12-08 13:10:47.35	127.0.0.1	127.0.0.1	TCP	68	52670 > telnet [ACK] Seq=7 Ack=21 Win=256 Len=0 TSval=553201 TSecr=553201
7	2017-12-08 13:10:48.46	127.0.0.1	127.0.0.1	TELNET	70	Telnet Data ...
8	2017-12-08 13:10:48.41	127.0.0.1	127.0.0.1	TELNET	70	Telnet Data ...
9	2017-12-08 13:10:48.41	127.0.0.1	127.0.0.1	TCP	68	52670 > telnet [ACK] Seq=9 Ack=23 Win=256 Len=0 TSval=553466 TSecr=553466
10	2017-12-08 13:10:48.56	127.0.0.1	127.0.0.1	TELNET	137	Telnet Data ...
11	2017-12-08 13:10:48.56	127.0.0.1	127.0.0.1	TCP	68	52670 > telnet [ACK] Seq=9 Ack=92 Win=256 Len=0 TSval=553486 TSecr=553486
12	2017-12-08 13:10:50.82	127.0.0.1	127.0.0.1	TELNET	71	Telnet Data ...
13	2017-12-08 13:10:50.82	127.0.0.1	127.0.0.1	TELNET	84	Telnet Data ...
14	2017-12-08 13:10:50.82	127.0.0.1	127.0.0.1	TCP	68	52670 > telnet [ACK] Seq=12 Ack=108 Win=256 Len=0 TSval=554069 TSecr=554069
15	2017-12-08 13:10:51.42	127.0.0.1	127.0.0.1	TELNET	70	Telnet Data ...
16	2017-12-08 13:10:51.43	127.0.0.1	127.0.0.1	TELNET	70	Telnet Data ...
17	2017-12-08 13:10:51.43	127.0.0.1	127.0.0.1	TCP	68	52670 > telnet [ACK] Seq=14 Ack=110 Win=256 Len=0 TSval=554220 TSecr=554220
18	2017-12-08 13:10:51.51	127.0.0.1	127.0.0.1	TELNET	137	Telnet Data ...
19	2017-12-08 13:10:51.51	127.0.0.1	127.0.0.1	TCP	68	52670 > telnet [ACK] Seq=14 Ack=179 Win=256 Len=0 TSval=554241 TSecr=554241

Task 2.c: Spoof an Ethernet Frame. For this part, I used a ethernet frame sender with code modified from **https://gist.github.com/austinmarton/1922600**. This allows an attacker to send the spoofed ethernet frame while haiving 01:02:03:04:05:06 as the source address.

Question 4. Can you set the IP packet length field to an arbitrary value, regardless of how big the actual packet is?

Not necessarily. This is because the program will error out if the packet length field is too large. It should just cut off the packet and send an incomplete packet.

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

For raw socket programming, you do not need to calculate the checksum of the IP header because there are no checks. This would be a different story for regular sockets.

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?

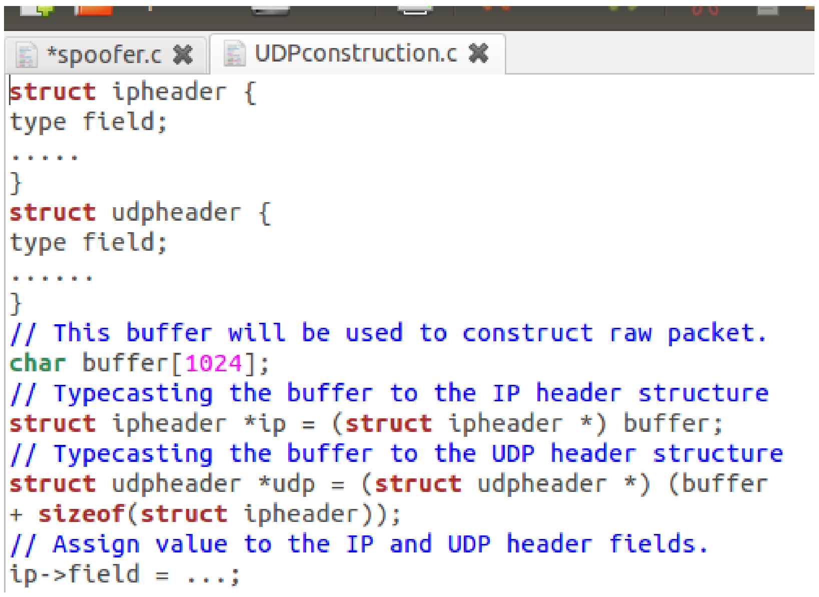
According to **http://opensourceforu.com/2015/03/a-guide-to-using-raw-sockets/**, it appears that you need root privilege to run programs on raw sockets because raw sockets require full access to the network controller.

2.3 Task 3: Sniff and then Spoof

For this task, I was unsuccessful in combining the sniffing and spoofing techniques to implement a sniff-and-then-spoof program. I successfully got the two Vms on the same LAN. This Is shown below in the following screenshot.

3.1 Filling in Data in Raw Packets

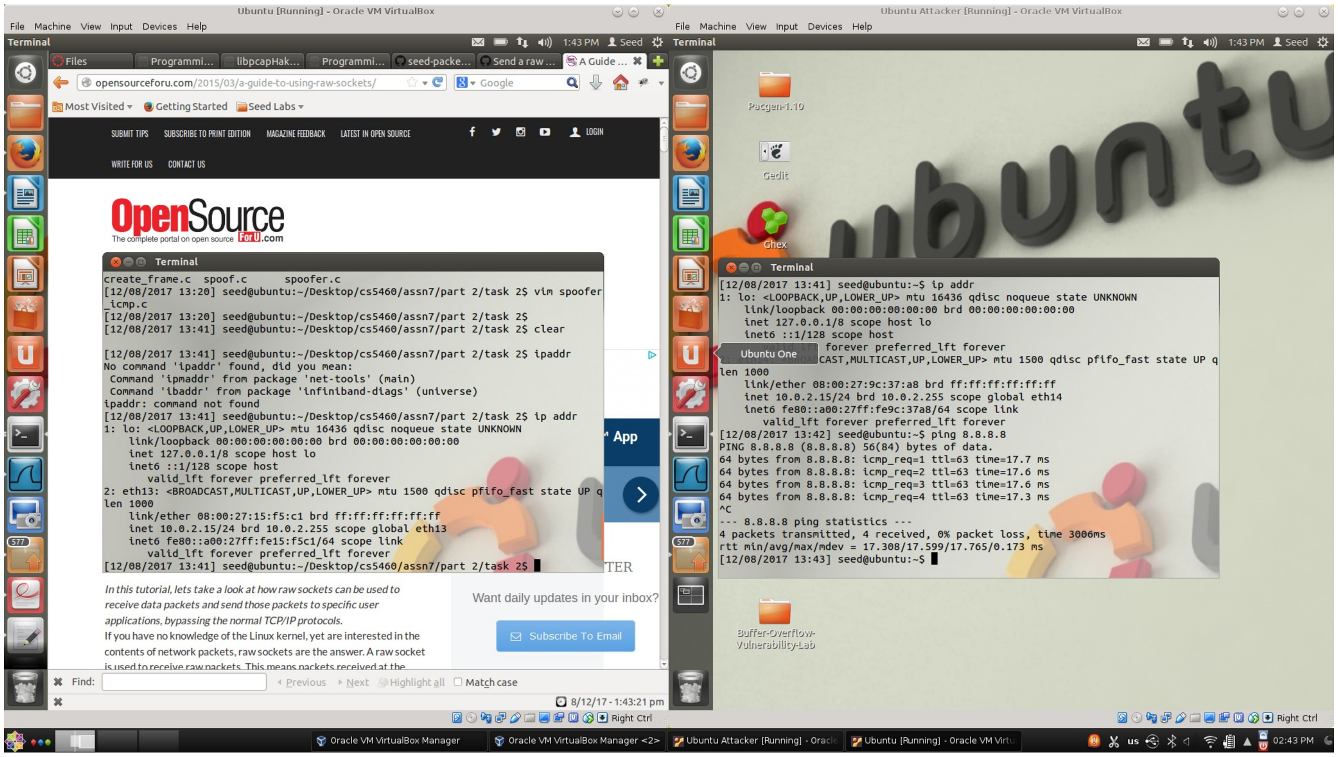
For this part I took the provided code and ran it. Files can be found in the 3.1 firectory of part 3.



```
struct ipheader {
type field;
.....
}
struct udpheader {
type field;
......
}
// This buffer will be used to construct raw packet.
char buffer[1024];
// Typecasting the buffer to the IP header structure
struct ipheader *ip = (struct ipheader *) buffer;
// Typecasting the buffer to the UDP header structure
struct udpheader *udp = (struct udpheader *) (buffer
+ sizeof(struct ipheader));
// Assign value to the IP and UDP header fields.
ip->field = ...;
```

3.2 Network/Host Byte Order and the Conversions

Below is a picture of the Host VM and a secondary VM.



```
create_frame.c  spoof.c      spoofer.c
[12/08/2017 13:20] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 2$ vim spoofer
_icmp.c
[12/08/2017 13:20] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 2$
[12/08/2017 13:41] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 2$ clear

[12/08/2017 13:41] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 2$ ipaddr
No command 'ipaddr' found, did you mean:
 Command 'ipmaddr' from package 'net-tools' (main)
 Command 'ibaddr' from package 'infiniband-diags' (universe)
ipaddr: command not found
[12/08/2017 13:41] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 2$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP q
len 1000
    link/ether 08:00:27:15:f5:c1 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth13
    inet6 fe80::a00:27ff:fe15:f5c1/64 scope link
       valid_lft forever preferred_lft forever
[12/08/2017 13:41] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 2$
```

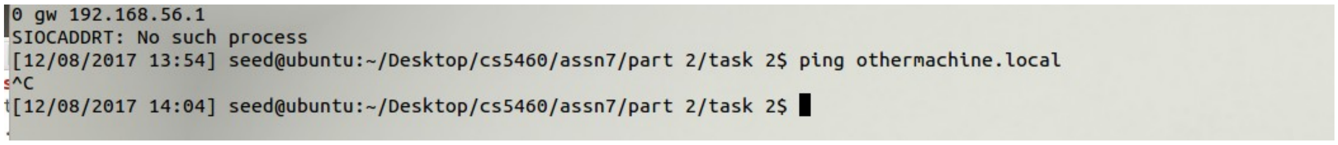
In this tutorial, lets take a look at how raw sockets can be used to receive data packets and send those packets to specific user applications, bypassing the normal TCP/IP protocols.
If you have no knowledge of the Linux kernel, yet are interested in the contents of network packets, raw sockets are the answer. A raw socket is used to receive raw packets. This means packets received at the

```
[12/08/2017 13:41] seed@ubuntu:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth14: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP q
len 1000
    link/ether 08:00:27:9c:37:a8 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth14
    inet6 fe80::a00:27ff:fe9c:37a8/64 scope link
       valid_lft forever preferred_lft forever
[12/08/2017 13:42] seed@ubuntu:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=63 time=17.7 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=63 time=17.6 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=63 time=17.6 ms
64 bytes from 8.8.8.8: icmp_req=4 ttl=63 time=17.3 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 17.308/17.599/17.765/0.173 ms
[12/08/2017 13:43] seed@ubuntu:~$
```

My CPU is a x64 so I believe it does not follow the Little Endian format. As for the packet buffer, I am unsure what this part of the assignment is asking. I am confident, however that whatever goes on on my machine is being converted properly to a network byte order format.

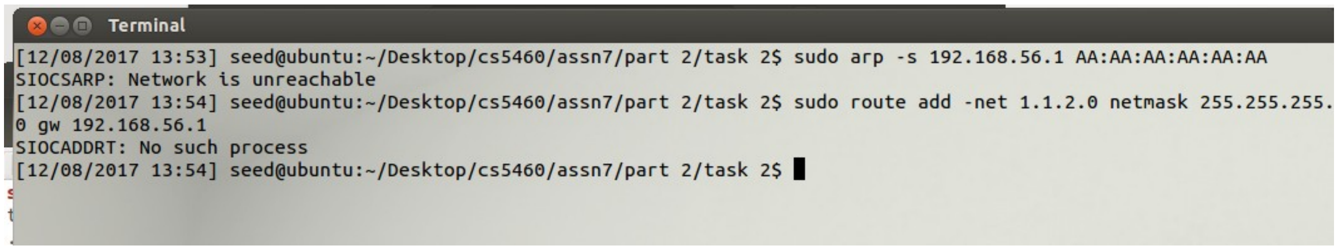
3.3 VirtualBox Network Configuration for Task 3

Attempted Ping:



```
0 gw 192.168.56.1
SIOCADDRT: No such process
[12/08/2017 13:54] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 2$ ping othermachine.local
^C
[12/08/2017 14:04] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 2$
```

Adjust rules by spoofing MAC address.



```
[12/08/2017 13:53] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 2$ sudo arp -s 192.168.56.1 AA:AA:AA:AA:AA:AA
SIOCSARP: Network is unreachable
[12/08/2017 13:54] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 2$ sudo route add -net 1.1.2.0 netmask 255.255.255.
0 gw 192.168.56.1
SIOCADDRT: No such process
[12/08/2017 13:54] seed@ubuntu:~/Desktop/cs5460/assn7/part 2/task 2$
```

After this, I followed the directions in the assignment description of changing the VM's to be on a bridged network with promiscuous mode turnd on. After this, they were able to see each other.