Praarthana Ramakrishnan (praartr)

Advanced Network Security

Project 2

TASK 1: Writing Packet Sniffing Program

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.

Solution:

1. Set the device : To choose which interface to sniff on

   2 ways to do it:

   - User defines the device using string
   - pcap gives name of interface using
     dev = pcap_lookupdev(errbuf); If this fails, error message is saved in errbuf.

2. Open the device for sniffing
   - pcap_t *pcap_open_live(char *device, int snaplen, int promisc, int to_ms, char *ebuf) where
     1. char * device - device that we want to stiff on.
     2. int snaplen - maximum number of bytes captured by pcap
     3. int promisc – 1 for promiscuous mode and 0 for non-promiscuous mode
     4. int to_ms – read time out in milliseconds
     5. char *ebuf – saves error message
3. Filter Traffic
   - Compile the filter using
     Int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int optimize, bpf_u_int32 netmask)
     Where
     1. pcap_t *p - session handle
     2. struct bpf_program *fp - to strore the compiler version of the filter
     3. char * str - the expression
     4. int optimize – 1 means network should be optimized
   - pcap_lookupnet() - returns one of IPV4 network numbers and mask.

- int pcap_setfilter(pcap_t *p, struct bpf_program *fp)
  where
  1. pcap_t *p – session handler
  2. struct bpf)program *fp – reference to compilerd version of expression

4. Sniffing
   - Capture a single packet at a time using:
     u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h);
     where
     1. pcap_t *p – session handler
     2. struct pcap_pkthdr *h – pointer to general information about the packet

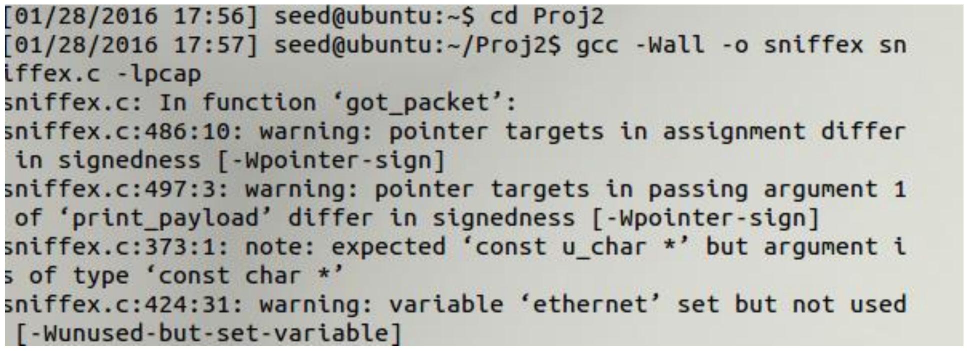
   - Perform a loop that waits for n number of packets to be sniffed using :
     Int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)
     Where
     1. pcap_t *p – session handler
     2. int cnt – how many packets to sniff
     3. pcap_handler callback – the name of callback function

5. Close the sniffer session
   - pcap_close()

<u>2)</u> Why do you need the root privilege to run sniffex? Where does the program fail if executed without the root privilege?

<u>Solution:</u>

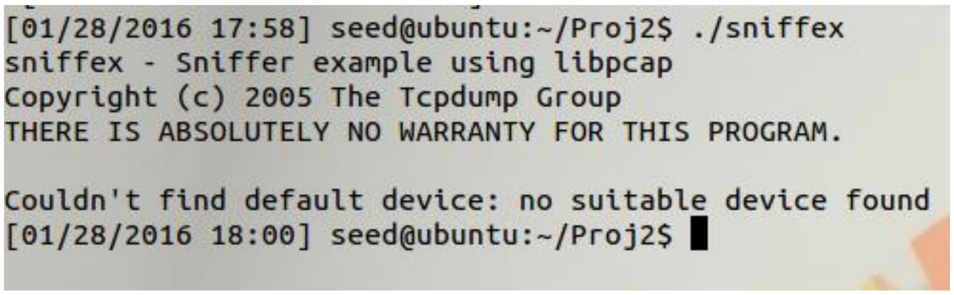
STEP 1: First, we compile sniffex.c like shown in Figure 1 below.



```
[01/28/2016 17:56] seed@ubuntu:~$ cd Proj2
[01/28/2016 17:57] seed@ubuntu:~/Proj2$ gcc -Wall -o sniffex sn
iffex.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 i
s of type ‘const char *’
sniffex.c:424:31: warning: variable ‘ethernet’ set but not used
 [-Wunused-but-set-variable]
```

Figure 1

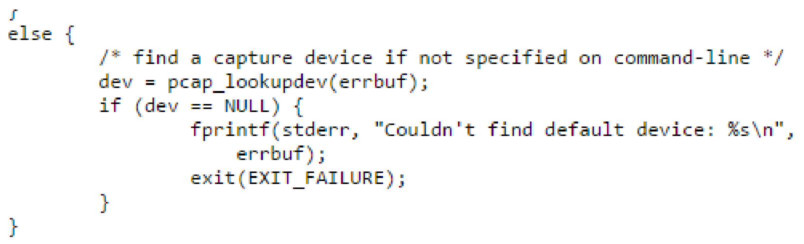
Once it is compiled, we try to run sniffex without root privilege as shown as Figure 2



```
[01/28/2016 17:58] seed@ubuntu:~/Proj2$ ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Couldn't find default device: no suitable device found
[01/28/2016 18:00] seed@ubuntu:~/Proj2$
```

Figure 2

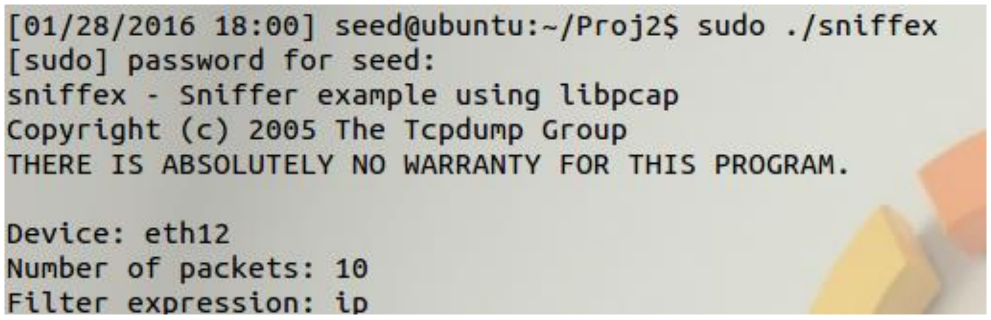
The message is printed "Couldn't find default device: no suitable device found "because there is no root privilege. The error message is saved in the errbuf and the error message is printed.



```
}
else {
        /* find a capture device if not specified on command-line */
        dev = pcap_lookupdev(errbuf);
        if (dev == NULL) {
                fprintf(stderr, "Couldn't find default device: %s\n",
                    errbuf);
                exit(EXIT_FAILURE);
        }
}
```

Figure 3

Successfully runs sniffex with root privilege as shown in Figure 4 below.



```
[01/28/2016 18:00] seed@ubuntu:~/Proj2$ 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: eth12
Number of packets: 10
Filter expression: ip
```

Figure 4

<u>3)</u> 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.

<u>Solution:</u>

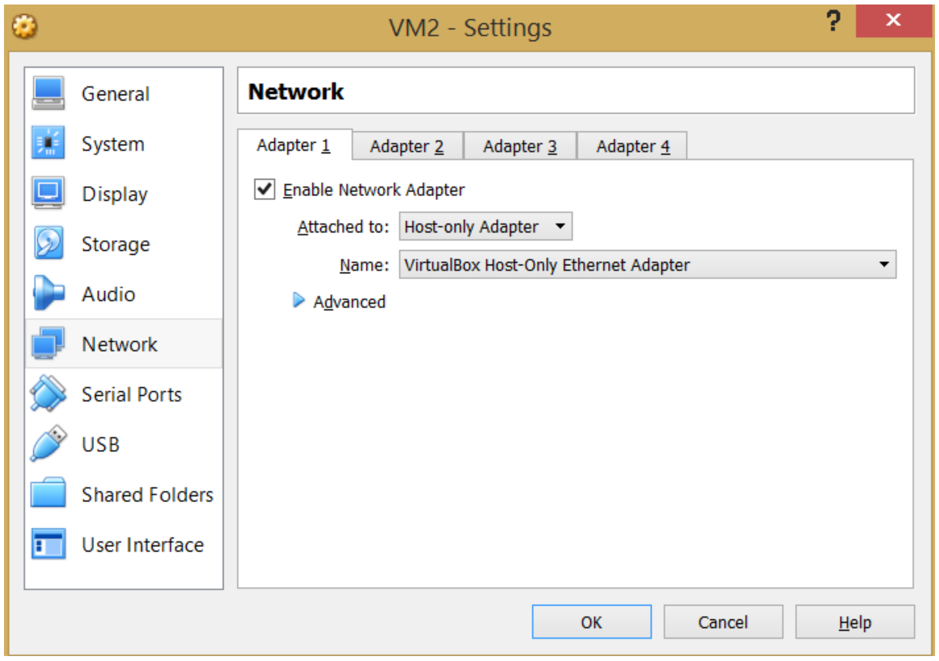
Promiscuous mode and Non-Promiscuous mode:

In non-promiscuous sniffing, a host is sniffing only traffic that is directly related to it. Only traffic to, from, or routed through the host will be picked up by the sniffer.

Promiscuous mode, sniffs all traffic on the wire.

I set up two virtual machines and all in running mode. Then I turned on the promiscuous mode.

Turn on virtual machine's network setting as host-only to be under the same network.



☑ Enable Network Adapter

Attached to: Host-only Adapter

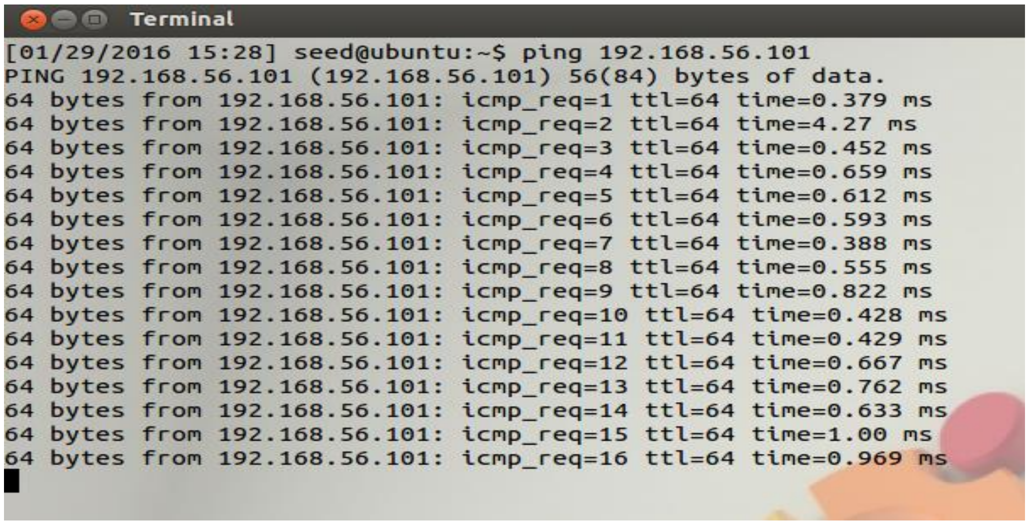
Name: VirtualBox Host-Only Ethernet Adapter

Figure 5

Type "ifconfig" to find the IP addresses of the two virtual machines.

Now we have two virtual machines with IP addresses as: Ubuntu (192.168.56.101) and Ubuntu Clone (192.68.56.102).

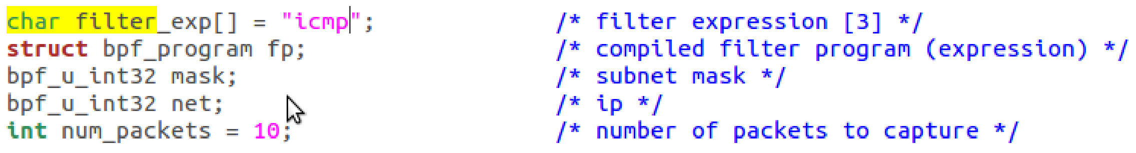
The communication is initiated using ping command as shown as figure 6.



```
[01/29/2016 15:28] seed@ubuntu:~$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_req=1 ttl=64 time=0.379 ms
64 bytes from 192.168.56.101: icmp_req=2 ttl=64 time=4.27 ms
64 bytes from 192.168.56.101: icmp_req=3 ttl=64 time=0.452 ms
64 bytes from 192.168.56.101: icmp_req=4 ttl=64 time=0.659 ms
64 bytes from 192.168.56.101: icmp_req=5 ttl=64 time=0.612 ms
64 bytes from 192.168.56.101: icmp_req=6 ttl=64 time=0.593 ms
64 bytes from 192.168.56.101: icmp_req=7 ttl=64 time=0.388 ms
64 bytes from 192.168.56.101: icmp_req=8 ttl=64 time=0.555 ms
64 bytes from 192.168.56.101: icmp_req=9 ttl=64 time=0.822 ms
64 bytes from 192.168.56.101: icmp_req=10 ttl=64 time=0.428 ms
64 bytes from 192.168.56.101: icmp_req=11 ttl=64 time=0.429 ms
64 bytes from 192.168.56.101: icmp_req=12 ttl=64 time=0.667 ms
64 bytes from 192.168.56.101: icmp_req=13 ttl=64 time=0.762 ms
64 bytes from 192.168.56.101: icmp_req=14 ttl=64 time=0.633 ms
64 bytes from 192.168.56.101: icmp_req=15 ttl=64 time=1.00 ms
64 bytes from 192.168.56.101: icmp_req=16 ttl=64 time=0.969 ms
```

Figure 6

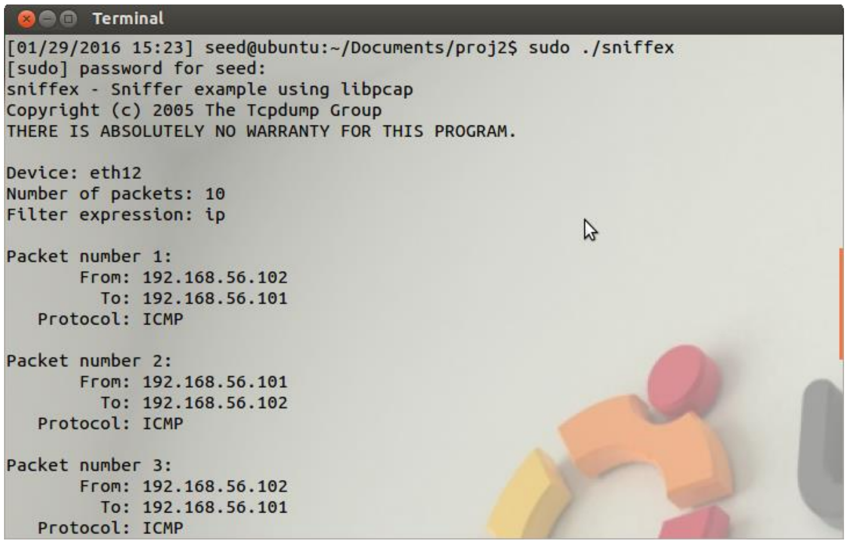
Now, the communication is good. The filter expression is changed to "icmp" since ping uses "icmp".



```
char filter_exp[] = "icmp";                /* filter expression [3] */
struct bpf_program fp;                     /* compiled filter program (expression) */
bpf_u_int32 mask;                          /* subnet mask */
bpf_u_int32 net;                           /* ip */
int num_packets = 10;                      /* number of packets to capture */
```

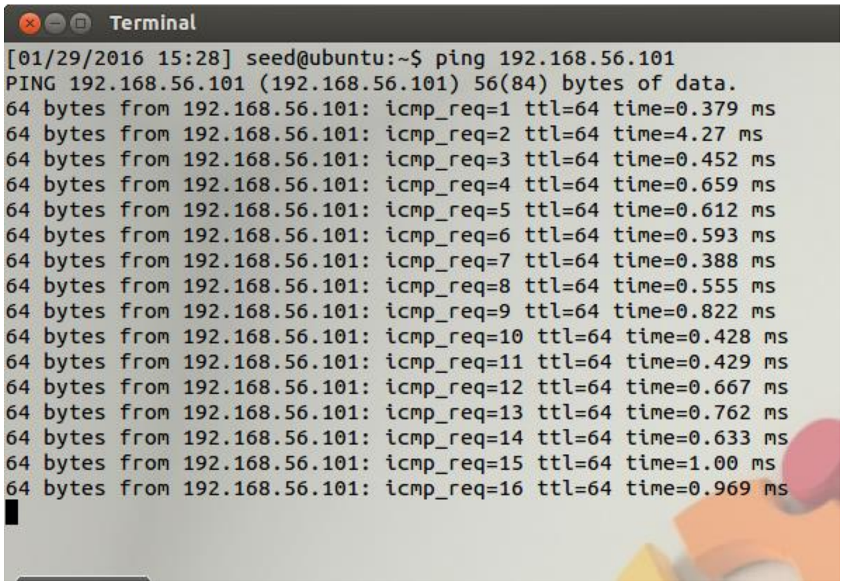
Figure 7

Now the promiscuous mode is ON as shown in figure 8 and figure 9 and run sniffex from Ubuntu with root privilege and capture all traffic from the network.



```
[01/29/2016 15:23] seed@ubuntu:~/Documents/proj2$ 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: eth12
Number of packets: 10
Filter expression: ip

Packet number 1:
       From: 192.168.56.102
         To: 192.168.56.101
   Protocol: ICMP

Packet number 2:
       From: 192.168.56.101
         To: 192.168.56.102
   Protocol: ICMP

Packet number 3:
       From: 192.168.56.102
         To: 192.168.56.101
   Protocol: ICMP
```

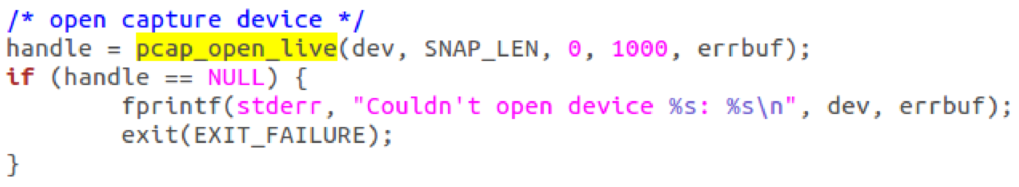
Figure 8



```
[01/29/2016 15:28] seed@ubuntu:~$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_req=1 ttl=64 time=0.379 ms
64 bytes from 192.168.56.101: icmp_req=2 ttl=64 time=4.27 ms
64 bytes from 192.168.56.101: icmp_req=3 ttl=64 time=0.452 ms
64 bytes from 192.168.56.101: icmp_req=4 ttl=64 time=0.659 ms
64 bytes from 192.168.56.101: icmp_req=5 ttl=64 time=0.612 ms
64 bytes from 192.168.56.101: icmp_req=6 ttl=64 time=0.593 ms
64 bytes from 192.168.56.101: icmp_req=7 ttl=64 time=0.388 ms
64 bytes from 192.168.56.101: icmp_req=8 ttl=64 time=0.555 ms
64 bytes from 192.168.56.101: icmp_req=9 ttl=64 time=0.822 ms
64 bytes from 192.168.56.101: icmp_req=10 ttl=64 time=0.428 ms
64 bytes from 192.168.56.101: icmp_req=11 ttl=64 time=0.429 ms
64 bytes from 192.168.56.101: icmp_req=12 ttl=64 time=0.667 ms
64 bytes from 192.168.56.101: icmp_req=13 ttl=64 time=0.762 ms
64 bytes from 192.168.56.101: icmp_req=14 ttl=64 time=0.633 ms
64 bytes from 192.168.56.101: icmp_req=15 ttl=64 time=1.00 ms
64 bytes from 192.168.56.101: icmp_req=16 ttl=64 time=0.969 ms
```

Figure 9

Now , the promiscuous mode is turned OFF by changing the argument to 0 as shown below in Figure 10.



```
/* open capture device */
handle = pcap_open_live(dev, SNAP_LEN, 0, 1000, errbuf);
if (handle == NULL) {
        fprintf(stderr, "Couldn't open device %s: %s\n", dev, errbuf);
        exit(EXIT_FAILURE);
}
```

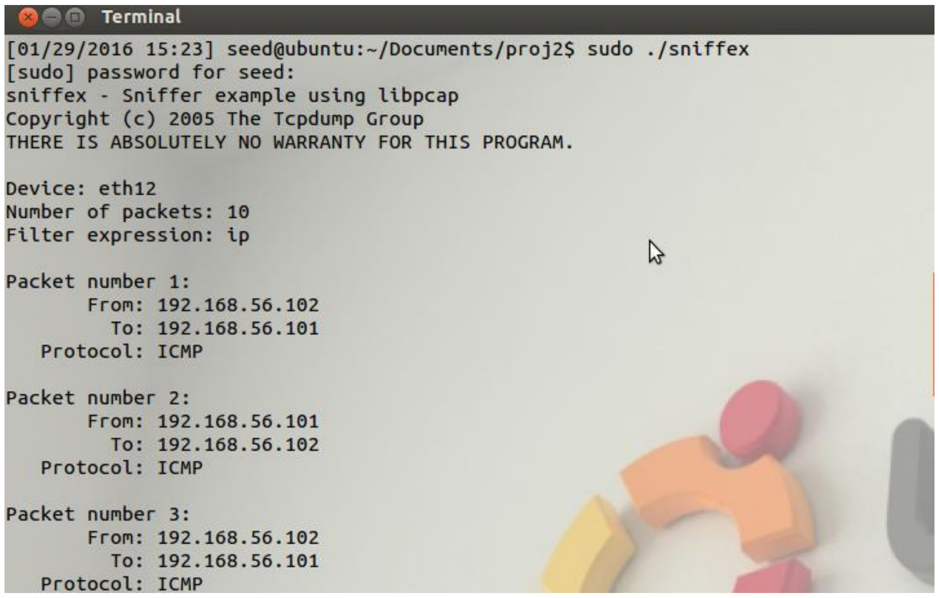
Figure 10

Since the promiscuous mode is turned OFF, it can only capture packets from or to it.

4) **Please write filter expressions to capture each of the followings. In your lab reports, you need to include screen dumps to show the results of applying each of these filters.**

- **Capture the ICMP packets between two specific hosts.**
- **Capture the TCP packets that have a destination port range from to port 10 – 100.**

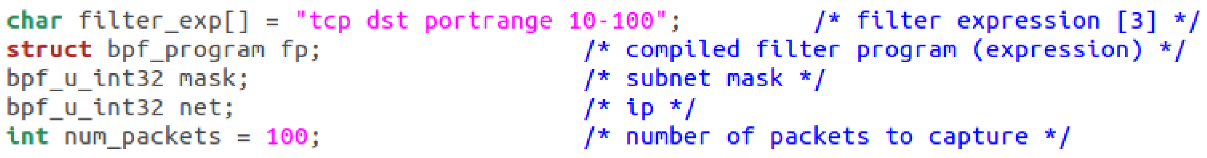
ICMP packets are already captured before as shown in Figure 7 and Figure 11.



```
[01/29/2016 15:23] seed@ubuntu:~/Documents/proj2$ 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: eth12
Number of packets: 10
Filter expression: ip

Packet number 1:
       From: 192.168.56.102
         To: 192.168.56.101
   Protocol: ICMP

Packet number 2:
       From: 192.168.56.101
         To: 192.168.56.102
   Protocol: ICMP

Packet number 3:
       From: 192.168.56.102
         To: 192.168.56.101
   Protocol: ICMP
```

Figure 11

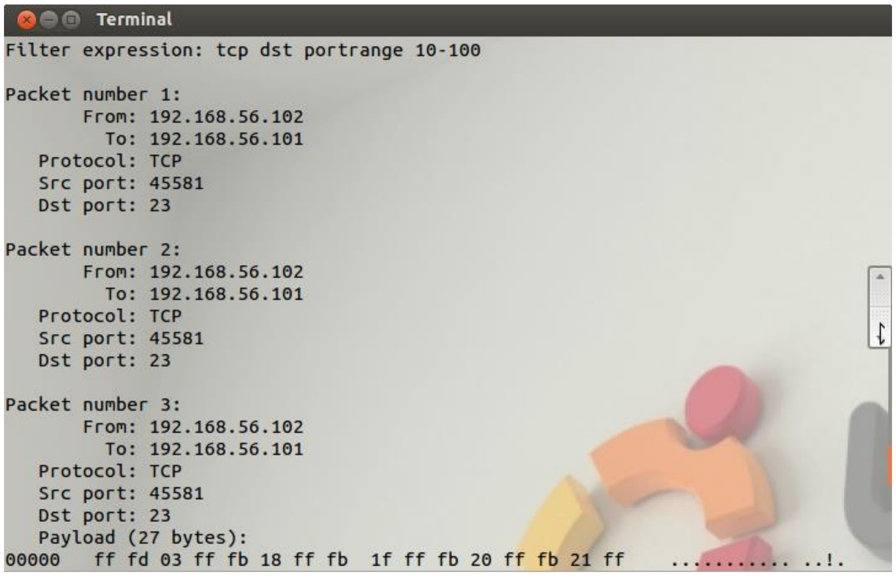
To capture TCP packets that have a destination port range from 10-100, we need to use filter expression as shown in Figure 12.



```
char filter_exp[] = "tcp dst portrange 10-100";         /* filter expression [3] */
struct bpf_program fp;                          /* compiled filter program (expression) */
bpf_u_int32 mask;                               /* subnet mask */
bpf_u_int32 net;                                /* ip */
int num_packets = 100;                          /* number of packets to capture */
```

Figure 12

The TCP packets are successfully being captured as shown in Figure 13.



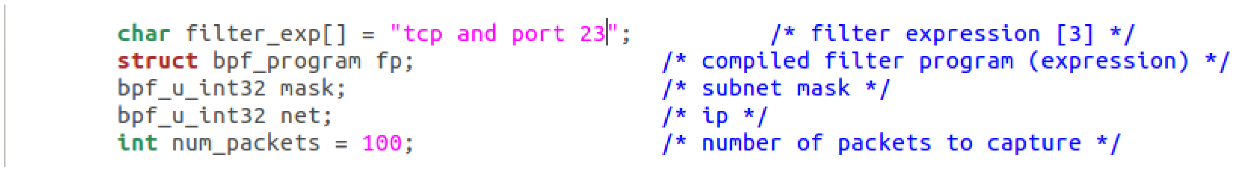
```
Filter expression: tcp dst portrange 10-100

Packet number 1:
       From: 192.168.56.102
         To: 192.168.56.101
   Protocol: TCP
   Src port: 45581
   Dst port: 23

Packet number 2:
       From: 192.168.56.102
         To: 192.168.56.101
   Protocol: TCP
   Src port: 45581
   Dst port: 23

Packet number 3:
       From: 192.168.56.102
         To: 192.168.56.101
   Protocol: TCP
   Src port: 45581
   Dst port: 23
   Payload (27 bytes):
00000   ff fd 03 ff fb 18 ff fb  1f ff fb 20 ff fb 21 ff    ........... ..!.
```

Figure 13

5) Please show how you can use sniffex to capture the password when somebody is using telnet on the network that you are monitoring. You may need to modify the sniffex.c a little bit if needed. You also need to start the telnetd server on your VM. If you are using our pre-built VM, the telnetd server is already installed; just type the following command to start it.

% sudo service openbsd-inetd start

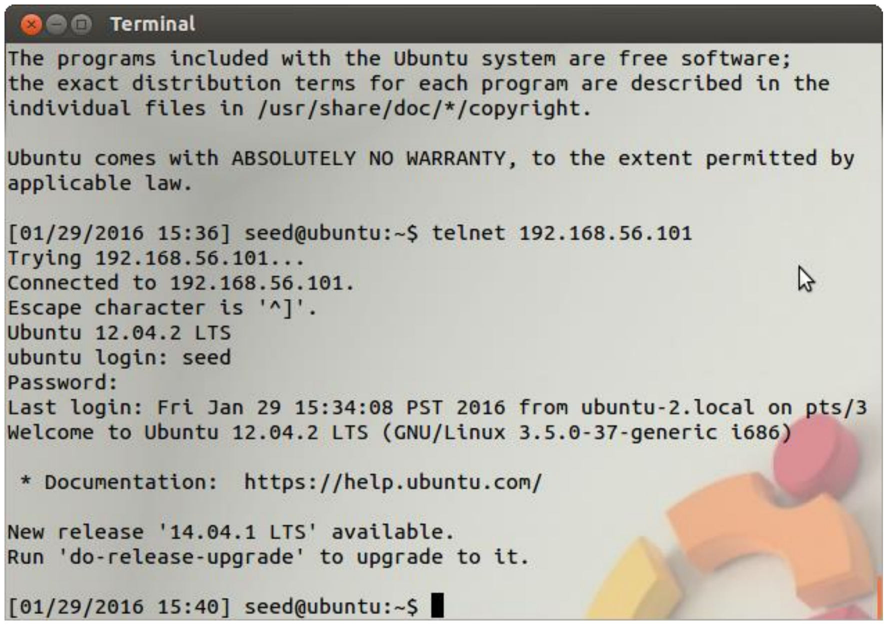
The filter expression is changed to “tcp and port 23” to capture telnet packets as shown in Figure 14.



```
	char filter_exp[] = "tcp and port 23";		/* filter expression [3] */
	struct bpf_program fp;			/* compiled filter program (expression) */
	bpf_u_int32 mask;			/* subnet mask */
	bpf_u_int32 net;			/* ip */
	int num_packets = 100;			/* number of packets to capture */
```

Figure 14

Then we start telnet server on VM as shown in Figure 15. This is how we use sniffex to capture password when somebody is using telnet on the network.



```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

[01/29/2016 15:36] seed@ubuntu:~$ telnet 192.168.56.101
Trying 192.168.56.101...
Connected to 192.168.56.101.
Escape character is '^]'.
Ubuntu 12.04.2 LTS
ubuntu login: seed
Password:
Last login: Fri Jan 29 15:34:08 PST 2016 from ubuntu-2.local 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.

[01/29/2016 15:40] seed@ubuntu:~$ 
```

Figure 15