CPSC-8580 Security in Emerging Computing and Networking Systems

Lab1a – Packet Sniffing

Nandini Krupa Krishnamurthy
C10811031

Task 1: Writing Packet Sniffing Program

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 library calls that are required for snipper programs are as follows:

1. pcap_lookupdev()
   Used to set up the device that is supposed to be sniffed on.
2. pcap_lookupnet()
   When the device name is present, this function returns its IPV4 network numbers and its corresponding network mask i.e., the network part of the address. This in turn is used at the time of applying filter.
3. pcap_open_live()
   This is called to start a working session of sniffing.
4. pcap_datalink()
   This is used to get the link-layer header of the device which is essentially used when processing packets.
5. pcap_compile()
   Used before setting up pcap_setfilter() in order to compile the filter expression which is usually in a regular string format.
6. pcap_setfilter()
   Sets the filter so that only the network traffic of interest is sniffed on.
7. pcap_next()/pcap_loop()/pcap_dispatch()
   pcap_next() is used to sniff one pack at a time.
   pcap_loop() and pcap_dispatch() have similar functions. Both are used when a an integer tells them how many packets have to be sniffed on. The only difference that pcap_diapatch() processes only the first batch of packets that it receives from the system whereas pcap_loop() keeps process on until all the packets are processed.
8. pcap_geterr()
   returns the error text of the latest pcap library call in case an error occurs
9. pcap_freecode()
   Used to free up allocated memory by a BF program and used by pcap_compile().
10. pcap_close()
    Used to end the sniffing session.

Problem2: Why do you need the root privilege to run sniffex? Where does the program fail if executed without the root privilege?

a. Need of root privilege

   This is required by pcap_lookupdev(). If the device for sniffing is not set up manually then this function is used. This returns a pointer to a string which specifies the network device that is ideal for the current task of capturing packets. Though this is not ideal as the user has no control on the device that it returns, it needs root access because without the privileges, there is no

permission to inspect the devices that are provided by the operating system. Additionally, the program needs access to network interface for which root privileges is essential.

b. Point of execution fail

When not executed using root privilege, the program fails when it reaches this part of the code:

```
dev = pcap_lookupdev(errbuf);
if (dev == NULL) {
        fprintf(stderr, "Couldn't find default device: %s\n", errbuf);
        return(2);
}
```

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.

The whole process and trials are explained along with screenshots.

1. When promiscuous mode is turned on.
   This is achieved by changing the bit – promisc- to 1 in pcap_open_live() call. This is shown below.

```
/* open capture device */
handle = pcap_open_live(dev, SNAP_LEN, 1, 1000, errbuf);
if (handle == NULL) {
        fprintf(stderr, "Couldn't open device %s: %s\n", dev, errbuf);
        exit(EXIT_FAILURE);
}
```

When promisc = 1 i.e., when in Promiscuous mode, the sniffer sniffs all the traffic on the network. This might or might not be related to it. Completely depending on the reason, both advantage and disadvantage lies in the fact that the sniffer can sniff all the packets.

To test this, 3 VMs were created, IPs of which are as shown:

VM1:

```
[09/18/2019 16:04] seed@ubuntu:~/Desktop/security_lab/lab1$ ifconfig
eth14     Link encap:Ethernet  HWaddr 08:00:27:93:96:42
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe93:9642/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10871 errors:0 dropped:2 overruns:0 frame:0
          TX packets:8885 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5014128 (5.0 MB)  TX bytes:6907238 (6.9 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:538 errors:0 dropped:0 overruns:0 frame:0
          TX packets:538 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:76067 (76.0 KB)  TX bytes:76067 (76.0 KB)

[09/18/2019 16:05] seed@ubuntu:~/Desktop/security_lab/lab1$
```

VM2:

```
[09/18/2019 16:03] seed@ubuntu:~$ ifconfig
eth14     Link encap:Ethernet  HWaddr 08:00:27:db:78:31
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fedb:7831/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:46 errors:0 dropped:0 overruns:0 frame:0
          TX packets:104 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6289 (6.2 KB)  TX bytes:13228 (13.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:22 errors:0 dropped:0 overruns:0 frame:0
          TX packets:22 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1868 (1.8 KB)  TX bytes:1868 (1.8 KB)

[09/18/2019 16:05] seed@ubuntu:~$
```

VM3:

```
[09/18/2019 16:04] seed@ubuntu:~$ ifconfig
eth15     Link encap:Ethernet  HWaddr 08:00:27:43:f1:c8
          inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe43:f1c8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:123 errors:0 dropped:0 overruns:0 frame:0
          TX packets:130 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:30258 (30.2 KB)  TX bytes:15633 (15.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:26 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2096 (2.0 KB)  TX bytes:2096 (2.0 KB)

[09/18/2019 16:06] seed@ubuntu:~$
```

VM3 was pinged from VM2 along with running the sniffer on VM1 and the results are obtained as shown.

```
[09/18/2019 16:03] seed@ubuntu:~$ ifconfig
eth14     Link encap:Ethernet  HWaddr 08:00:27:db:78:31
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fedb:7831/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:46 errors:0 dropped:0 overruns:0 frame:0
          TX packets:104 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6289 (6.2 KB)  TX bytes:13228 (13.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:22 errors:0 dropped:0 overruns:0 frame:0
          TX packets:22 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1868 (1.8 KB)  TX bytes:1868 (1.8 KB)

[09/18/2019 16:05] seed@ubuntu:~$ ping 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_req=1 ttl=64 time=0.576 ms
64 bytes from 10.0.2.5: icmp_req=2 ttl=64 time=0.756 ms
64 bytes from 10.0.2.5: icmp_req=3 ttl=64 time=0.761 ms
64 bytes from 10.0.2.5: icmp_req=4 ttl=64 time=0.736 ms
64 bytes from 10.0.2.5: icmp_req=5 ttl=64 time=0.830 ms
64 bytes from 10.0.2.5: icmp_req=6 ttl=64 time=0.592 ms
64 bytes from 10.0.2.5: icmp_req=7 ttl=64 time=0.427 ms
64 bytes from 10.0.2.5: icmp_req=8 ttl=64 time=0.829 ms
64 bytes from 10.0.2.5: icmp_req=9 ttl=64 time=0.186 ms
64 bytes from 10.0.2.5: icmp_req=10 ttl=64 time=0.285 ms
```

Sniffer was able to capture all the packets going between VM2 (IP 10.0.2.15) to VM3 (10.0.2.5).

```
[09/18/2019 16:09] seed@ubuntu:~/Desktop/security_lab/lab1$ 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: eth14
Number of packets: 10
Filter expression: ip

Packet number 1:
       From: 10.0.2.15
         To: 10.0.2.5
   Protocol: ICMP

Packet number 2:
       From: 10.0.2.5
         To: 10.0.2.15
   Protocol: ICMP

Packet number 3:
       From: 10.0.2.15
         To: 10.0.2.5
   Protocol: ICMP

Packet number 4:
       From: 10.0.2.5
         To: 10.0.2.15
   Protocol: ICMP

Packet number 5:
       From: 10.0.2.15
         To: 10.0.2.5
   Protocol: ICMP

Packet number 6:
       From: 10.0.2.5
         To: 10.0.2.15
   Protocol: ICMP

Packet number 7:
       From: 10.0.2.15
         To: 10.0.2.5
   Protocol: ICMP

Packet number 8:
       From: 10.0.2.5
         To: 10.0.2.15
   Protocol: ICMP
```

When executed again after stopping the ping, packets were obtained from various other IPs and these packets were completely unrelated to the host on which sniffer was working on (IP 10.0.20.4). The results are shown in the following two screenshots

```
[09/18/2019 16:10] seed@ubuntu:~/Desktop/security_lab/lab1$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: ip

Packet number 1:
       From: 10.0.2.15
         To: 10.0.2.1
   Protocol: UDP

Packet number 2:
       From: 10.0.2.1
         To: 10.0.2.15
   Protocol: UDP

Packet number 3:
       From: 10.0.2.15
         To: 10.0.2.1
   Protocol: UDP

Packet number 4:
       From: 10.0.2.1
         To: 10.0.2.15
   Protocol: UDP

Packet number 5:
       From: 10.0.2.5
         To: 10.0.2.1
   Protocol: UDP

Packet number 6:
       From: 10.0.2.1
         To: 10.0.2.5
   Protocol: UDP

Packet number 7:
       From: 10.0.2.5
         To: 10.0.2.1
   Protocol: UDP

Packet number 8:
       From: 10.0.2.1
         To: 10.0.2.5
   Protocol: UDP
```

```
       From: 10.0.2.15
         To: 10.0.2.1
   Protocol: UDP

Packet number 2:
       From: 10.0.2.1
         To: 10.0.2.15
   Protocol: UDP

Packet number 3:
       From: 10.0.2.15
         To: 10.0.2.1
   Protocol: UDP

Packet number 4:
       From: 10.0.2.1
         To: 10.0.2.15
   Protocol: UDP

Packet number 5:
       From: 10.0.2.5
         To: 10.0.2.1
   Protocol: UDP

Packet number 6:
       From: 10.0.2.1
         To: 10.0.2.5
   Protocol: UDP

Packet number 7:
       From: 10.0.2.5
         To: 10.0.2.1
   Protocol: UDP

Packet number 8:
       From: 10.0.2.1
         To: 10.0.2.5
   Protocol: UDP

Packet number 9:
       From: 10.0.2.5
         To: 224.0.0.251
   Protocol: UDP

Packet number 10:
       From: 10.0.2.4
         To: 10.0.2.1
   Protocol: UDP

Capture complete.
[09/18/2019 16:11] seed@ubuntu:~/Desktop/security_lab/lab1$
```

2. When promiscuous mode is turned off.
   This is achieved by changing the bit - promisc- to 0 in pcap_open_live() call. This is shown below.

```
/* 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);
}
```

   When this mode is turned off (which is the standard case), the sniffer is only able to sniff the packets related to the host its working on. This is the case as in the screenshot shown below where its seen that the sniffer captures packers related to IP 10.0.2.4 (host IP).

```
[09/18/2019 16:15] seed@ubuntu:~/Desktop/security_lab/lab1$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: ip

Packet number 1:
       From: 10.0.2.4
         To: 10.0.2.1
   Protocol: UDP

Packet number 2:
       From: 10.0.2.1
         To: 10.0.2.4
   Protocol: UDP

Packet number 3:
       From: 10.0.2.4
         To: 10.0.2.1
   Protocol: UDP

Packet number 4:
       From: 10.0.2.1
         To: 10.0.2.4
   Protocol: UDP

Packet number 5:
       From: 10.0.2.4
         To: 10.0.2.1
   Protocol: UDP

Packet number 6:
       From: 10.0.2.1
         To: 10.0.2.4
   Protocol: UDP

Packet number 7:
       From: 10.0.2.4
         To: 10.0.2.1
   Protocol: UDP

Packet number 8:
       From: 10.0.2.1
         To: 10.0.2.4
   Protocol: UDP
```

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.

a. Capture the ICMP packets between two specific hosts.

The expression used is:

"icmp and (src host = 10.0.2.15 and dst host = 10.0.2.5) or (src host 10.0.2.5 = and dst host = 10.0.2.15)"

Different IPs (other than host's) were considered to further test Promiscuous modes.
Again, this was carried out in two cases – one when Promiscuous mode was turned on and Promiscuous mode was turned off.

1. When Promiscuous mode was turned off (promisc = 0)
We see that it captures no packet even when VM2 has pinged VM3.

```
[09/18/2019 16:25] seed@ubuntu:~/Desktop/security_lab/lab1$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: icmp and (src host 10.0.2.15 and dst host 10.0.2.5) or (src host 10.0.2.5 and dst host 10.0.2.15)
^C[09/18/2019 16:27] seed@ubuntu:~/Desktop/security_lab/lab1$
```

```
[09/18/2019 16:23] seed@ubuntu:~$ ping 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_req=1 ttl=64 time=0.171 ms
64 bytes from 10.0.2.5: icmp_req=2 ttl=64 time=0.154 ms
64 bytes from 10.0.2.5: icmp_req=3 ttl=64 time=0.171 ms
64 bytes from 10.0.2.5: icmp_req=4 ttl=64 time=0.172 ms
64 bytes from 10.0.2.5: icmp_req=5 ttl=64 time=0.148 ms
64 bytes from 10.0.2.5: icmp_req=6 ttl=64 time=0.169 ms
64 bytes from 10.0.2.5: icmp_req=7 ttl=64 time=0.155 ms
64 bytes from 10.0.2.5: icmp_req=8 ttl=64 time=0.176 ms
64 bytes from 10.0.2.5: icmp_req=9 ttl=64 time=0.239 ms
64 bytes from 10.0.2.5: icmp_req=10 ttl=64 time=0.148 ms
64 bytes from 10.0.2.5: icmp_req=11 ttl=64 time=0.158 ms
64 bytes from 10.0.2.5: icmp_req=12 ttl=64 time=0.180 ms
64 bytes from 10.0.2.5: icmp_req=13 ttl=64 time=0.194 ms
64 bytes from 10.0.2.5: icmp_req=14 ttl=64 time=0.170 ms
64 bytes from 10.0.2.5: icmp_req=15 ttl=64 time=0.179 ms
64 bytes from 10.0.2.5: icmp_req=16 ttl=64 time=0.158 ms
64 bytes from 10.0.2.5: icmp_req=17 ttl=64 time=0.157 ms
64 bytes from 10.0.2.5: icmp_req=18 ttl=64 time=0.196 ms
64 bytes from 10.0.2.5: icmp_req=19 ttl=64 time=0.159 ms
64 bytes from 10.0.2.5: icmp_req=20 ttl=64 time=0.157 ms
64 bytes from 10.0.2.5: icmp_req=21 ttl=64 time=0.163 ms
64 bytes from 10.0.2.5: icmp_req=22 ttl=64 time=0.173 ms
64 bytes from 10.0.2.5: icmp_req=23 ttl=64 time=0.239 ms
```

2. When Promiscuous mode was turned on (promisc = 1)
When promisc =1 and filter expression was still "icmp and (src host = 10.0.2.15 and dst host = 10.0.2.5) or (src host = 10.0.2.5 and dst host = 10.0.2.15)" , it was seen that the sniffer (which was running on a different VM) was able to capture all ICMP packets between these two hosts as seen in the screenshot below.

```
[09/18/2019 16:27] seed@ubuntu:~/Desktop/security_lab/lab1$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: icmp and (src host 10.0.2.15 and dst host 10.0.2.5) or (src host 10.0.2.5 and dst host 10.0.2.15)

Packet number 1:
       From: 10.0.2.15
         To: 10.0.2.5
   Protocol: ICMP

Packet number 2:
       From: 10.0.2.5
         To: 10.0.2.15
   Protocol: ICMP

Packet number 3:
       From: 10.0.2.15
         To: 10.0.2.5
   Protocol: ICMP

Packet number 4:
       From: 10.0.2.5
         To: 10.0.2.15
   Protocol: ICMP

Packet number 5:
       From: 10.0.2.15
         To: 10.0.2.5
   Protocol: ICMP

Packet number 6:
       From: 10.0.2.5
         To: 10.0.2.15
   Protocol: ICMP

Packet number 7:
       From: 10.0.2.15
         To: 10.0.2.5
   Protocol: ICMP

Packet number 8:
       From: 10.0.2.5
         To: 10.0.2.15
   Protocol: ICMP
```

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

Expression used – " tcp dst portrange 10-100"

This was again tested in two cases – one when Promiscuous mode was turned off and two when it was turned on.

1. When Promiscuous mode = 0

When a webpage was open in a different VM, there was no sniffing done as shown below.

```
[09/18/2019 16:32] seed@ubuntu:~/Desktop/security_lab/lab1$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: tcp dst portrange 10-100
```

When a webpage (port = 80) was opened in the same VM (IP 10.0.2.4) as the sniffer, it was noticed that that only packets following the filter expression were sniffed as seen in the screenshot below.

```
[09/18/2019 16:32] seed@ubuntu:~/Desktop/security_lab/lab1$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: tcp dst portrange 10-100

Packet number 1:
       From: 10.0.2.4
         To: 64.233.177.19
   Protocol: TCP
   Src port: 46305
   Dst port: 80

Packet number 2:
       From: 10.0.2.4
         To: 64.233.177.19
   Protocol: TCP
   Src port: 46305
   Dst port: 80

Packet number 3:
       From: 10.0.2.4
         To: 64.233.177.19
   Protocol: TCP
   Src port: 46305
   Dst port: 80
   Payload (285 bytes):
00000   47 45 54 20 2f 20 48 54  54 50 2f 31 2e 31 0d 0a    GET / HTTP/1.1..
00016   48 6f 73 74 3a 20 67 6d  61 69 6c 2e 63 6f 6d 0d    Host: gmail.com.
00032   0a 55 73 65 72 2d 41 67  65 6e 74 3a 20 4d 6f 7a    .User-Agent: Moz
00048   69 6c 6c 61 2f 35 2e 30  20 28 58 31 31 3b 20 55    illa/5.0 (X11; U
00064   62 75 6e 74 75 3b 20 4c  69 6e 75 78 20 69 36 38    buntu; Linux i68
00080   36 3b 20 72 76 3a 32 33  2e 30 29 20 47 65 63 6b    6; rv:23.0) Geck
00096   6f 2f 32 30 31 30 30 31  30 31 20 46 69 72 65 66    o/20100101 Firef
00112   6f 78 2f 32 33 2e 30 0d  0a 41 63 63 65 70 74 3a    ox/23.0..Accept:
00128   20 74 65 78 74 2f 68 74  6d 6c 2c 61 70 70 6c 69     text/html,appli
00144   63 61 74 69 6f 6e 2f 78  68 74 6d 6c 2b 78 6d 6c    cation/xhtml+xml
00160   2c 61 70 70 6c 69 63 61  74 69 6f 6e 2f 78 6d 6c    ,application/xml
00176   3b 71 3d 30 2e 39 2c 2a  2f 2a 3b 71 3d 30 2e 38    ;q=0.9,*/*;q=0.8
00192   0d 0a 41 63 63 65 70 74  2d 4c 61 6e 67 75 61 67    ..Accept-Languag
00208   65 3a 20 65 6e 2d 55 53  2c 65 6e 3b 71 3d 30 2e    e: en-US,en;q=0.
00224   35 0d 0a 41 63 63 65 70  74 2d 45 6e 63 6f 64 69    5..Accept-Encodi
00240   6e 67 3a 20 67 7a 69 70  2c 20 64 65 66 6c 61 74    ng: gzip, deflat
00256   65 0d 0a 43 6f 6e 6e 65  63 74 69 6f 6e 3a 20 6b    e..Connection: k
00272   65 65 70 2d 61 6c 69 76  65 0d 0a 0d 0a             eep-alive....

Packet number 4:
```

2. When Promiscuous mode = 1

The image below shows the result when webpages (port = 80) were opened in different VM (VM2 with IP: 10.0.2.15) other than the one on host machine. It is seen that packets were sniffed but with filter rules followed as shown below.

```
[09/18/2019 16:35] seed@ubuntu:~/Desktop/security_lab/lab1$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: tcp dst portrange 10-100

Packet number 1:
       From: 10.0.2.15
         To: 128.230.247.70
   Protocol: TCP
   Src port: 49588
   Dst port: 80

Packet number 2:
       From: 10.0.2.15
         To: 128.230.247.70
   Protocol: TCP
   Src port: 49588
   Dst port: 80

Packet number 3:
       From: 10.0.2.15
         To: 128.230.247.70
   Protocol: TCP
   Src port: 49588
   Dst port: 80
   Payload (406 bytes):
00000   47 45 54 20 2f 7e 77 65  64 75 2f 73 65 65 64 2f    GET /~wedu/seed/
00016   69 6e 64 65 78 2e 68 74  6d 6c 20 48 54 54 50 2f    index.html HTTP/
00032   31 2e 31 0d 0a 48 6f 73  74 3a 20 77 77 77 2e 63    1.1..Host: www.c
00048   69 73 2e 73 79 72 2e 65  64 75 0d 0a 55 73 65 72    is.syr.edu..User
00064   2d 41 67 65 6e 74 3a 20  4d 6f 7a 69 6c 6c 61 2f    -Agent: Mozilla/
00080   35 2e 30 20 28 58 31 31  3b 20 55 62 75 6e 74 75    5.0 (X11; Ubuntu
00096   3b 20 4c 69 6e 75 78 20  69 36 38 36 3b 20 72 76    ; Linux i686; rv
00112   3a 32 33 2e 30 29 20 47  65 63 6b 6f 2f 32 30 31    :23.0) Gecko/201
00128   30 30 31 30 31 20 46 69  72 65 66 6f 78 2f 32 33    00101 Firefox/23
00144   2e 30 0d 0a 41 63 63 65  70 74 3a 20 74 65 78 74    .0..Accept: text
00160   2f 68 74 6d 6c 2c 61 70  70 6c 69 63 61 74 69 6f    /html,applicatio
00176   6e 2f 78 68 74 6d 6c 2b  78 6d 6c 2c 61 70 70 6c    n/xhtml+xml,appl
00192   69 63 61 74 69 6f 6e 2f  78 6d 6c 3b 71 3d 30 2e    ication/xml;q=0.
00208   39 2c 2a 2f 2a 3b 71 3d  30 2e 38 0d 0a 41 63 63    9,*/*;q=0.8..Acc
00224   65 70 74 2d 4c 61 6e 67  75 61 67 65 3a 20 65 6e    ept-Language: en
00240   2d 55 53 2c 65 6e 3b 71  3d 30 2e 35 0d 0a 41 63    -US,en;q=0.5..Ac
00256   63 65 70 74 2d 45 6e 63  6f 64 69 6e 67 3a 20 67    cept-Encoding: g
00272   7a 69 70 2c 20 64 65 66  6c 61 74 65 0d 0a 43 6f    zip, deflate..Co
00288   6e 6e 65 63 74 69 6f 6e  3a 20 6b 65 65 70 2d 61    nnection: keep-a
00304   6c 69 76 65 0d 0a 49 66  2d 4d 6f 64 69 66 69 65    live..If-Modifie
00320   64 2d 53 69 6e 63 65 3a  20 4d 6f 6e 2c 20 30 32    d-Since: Mon, 02
```

When Telnet (port = 23) was used, the results as shown below was obtained as seen below.

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

Packet number 1:
       From: 10.0.2.5
         To: 72.21.91.29
   Protocol: TCP
   Src port: 38318
   Dst port: 80

Packet number 2:
       From: 10.0.2.5
         To: 72.21.91.29
   Protocol: TCP
   Src port: 38318
   Dst port: 80

Packet number 3:
       From: 10.0.2.15
         To: 74.125.21.105
   Protocol: TCP
   Src port: 58933
   Dst port: 23

Packet number 4:
       From: 10.0.2.15
         To: 74.125.21.105
   Protocol: TCP
   Src port: 58933
   Dst port: 23

Packet number 5:
       From: 10.0.2.15
         To: 74.125.21.105
   Protocol: TCP
   Src port: 58933
   Dst port: 23

Packet number 6:
       From: 10.0.2.15
         To: 74.125.21.105
   Protocol: TCP
   Src port: 58933
   Dst port: 23

Packet number 7:
       From: 10.0.2.15
         To: 74.125.21.105
   Protocol: TCP
   Src port: 58933
   Dst port: 23
```