Steven Dellamore
Sdelamo
0278562664
Note: I used MC10 NOT MC02

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.

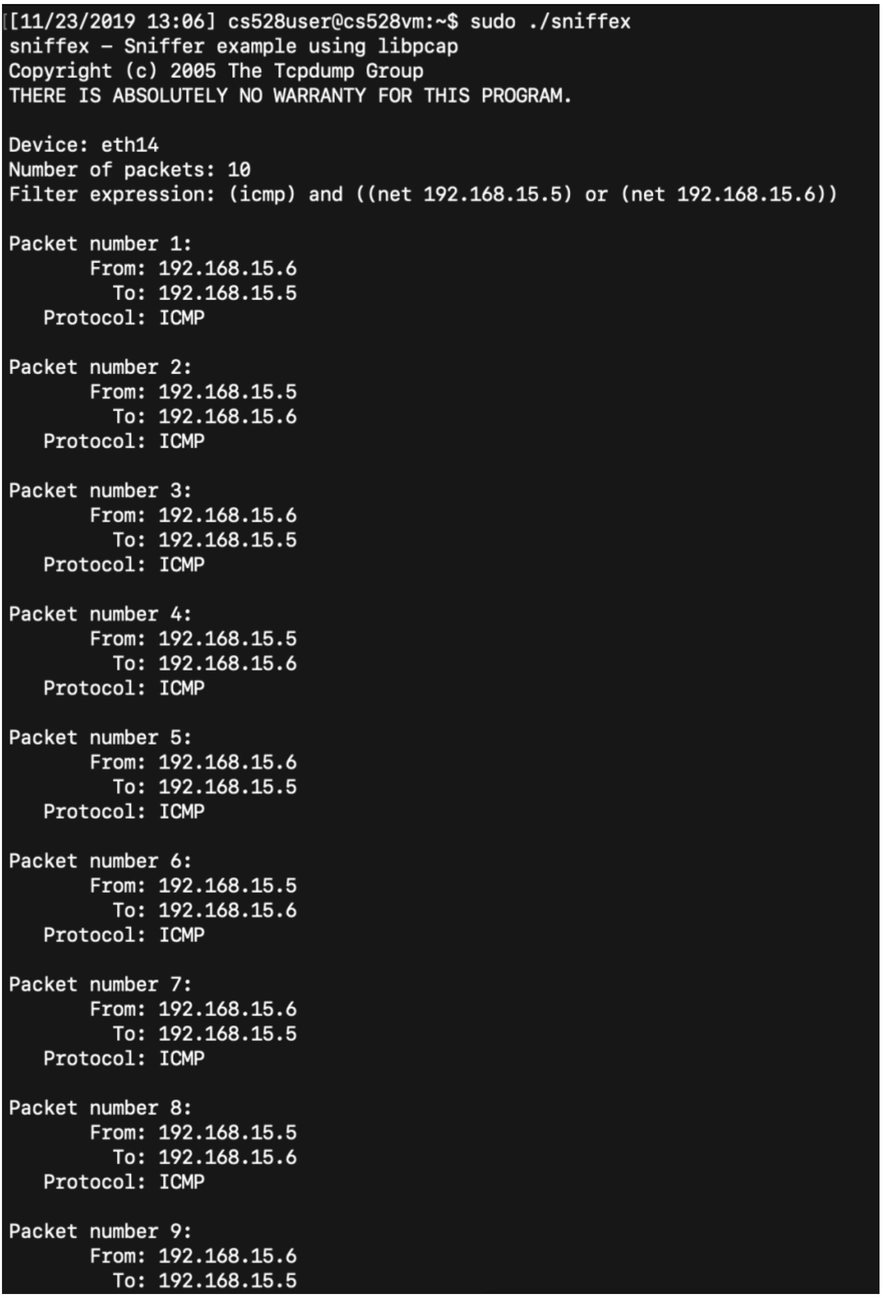
Problem 3:

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.

Steven Dellamore
Sdelamo
0278562664
Note: I used MC10 NOT MC02

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))”

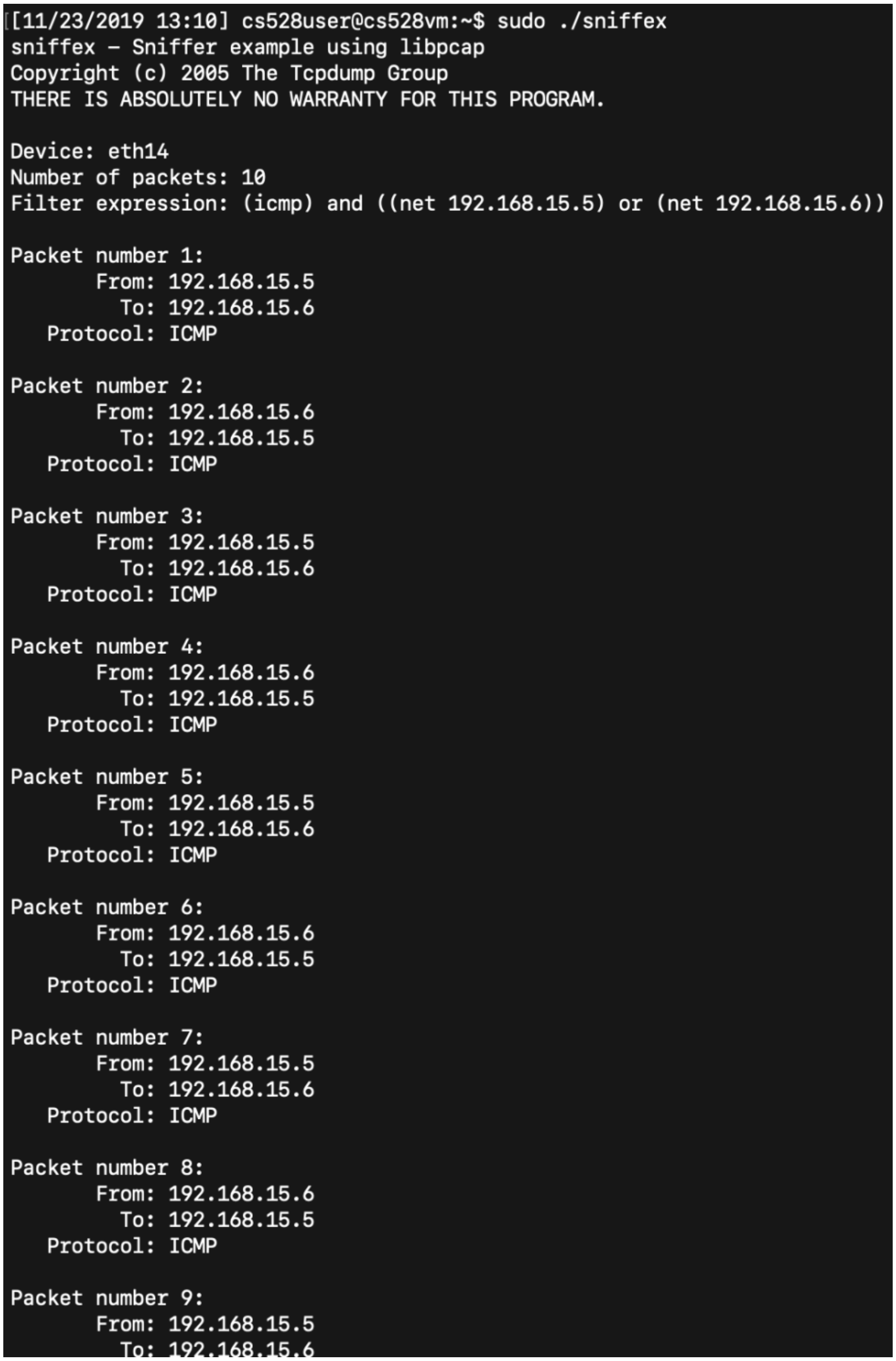


```
[11/23/2019 13:06] cs528user@cs528vm:~$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: (icmp) and ((net 192.168.15.5) or (net 192.168.15.6))

Packet number 1:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: ICMP

Packet number 2:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: ICMP

Packet number 3:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: ICMP

Packet number 4:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: ICMP

Packet number 5:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: ICMP

Packet number 6:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: ICMP

Packet number 7:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: ICMP

Packet number 8:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: ICMP

Packet number 9:
       From: 192.168.15.6
         To: 192.168.15.5
```

`

.6 pinging .5

Steven Dellamore
Sdelamo
0278562664
Note: I used MC10 NOT MC02



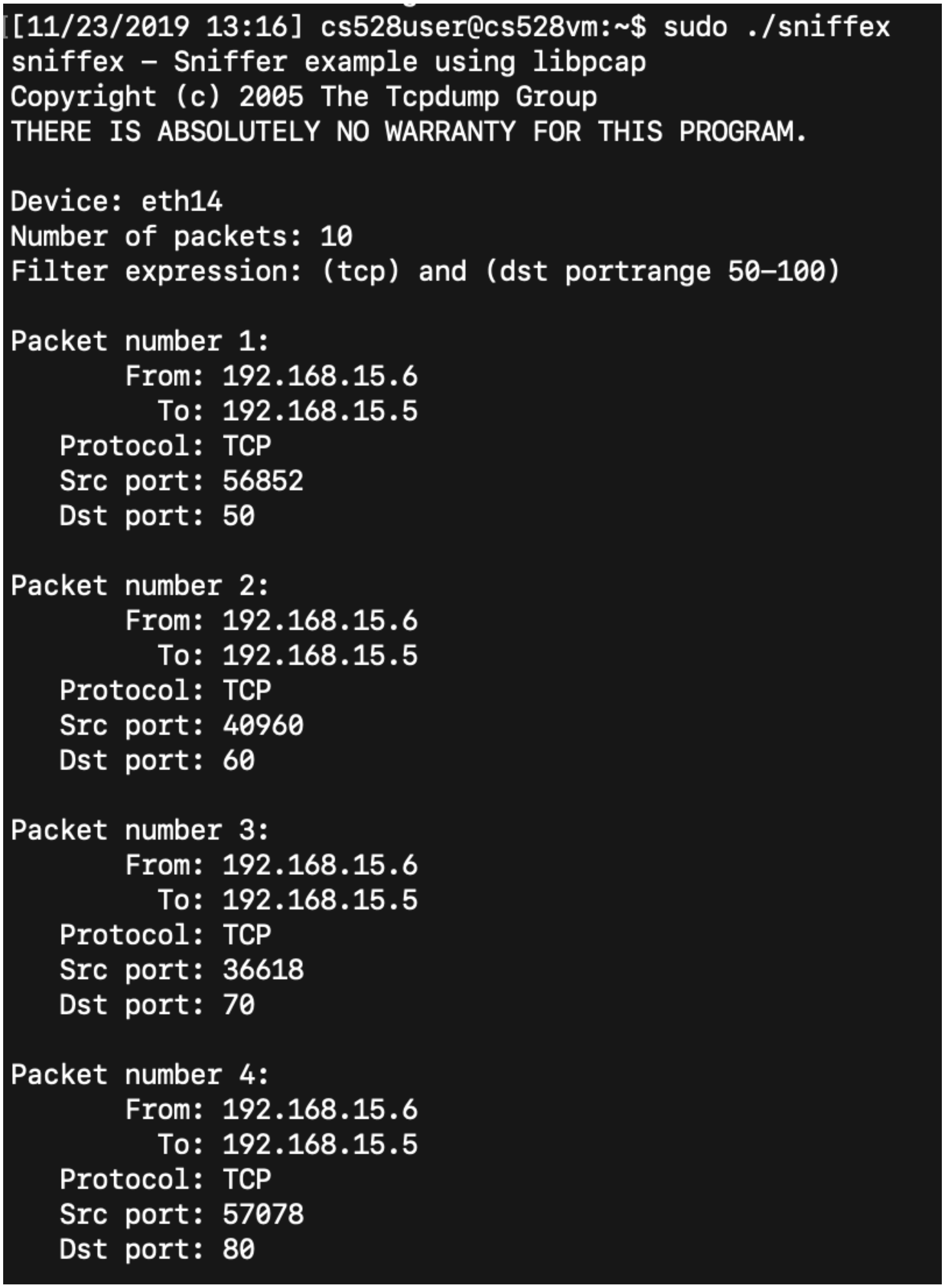
```
[[11/23/2019 13:10] cs528user@cs528vm:~$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 10
Filter expression: (icmp) and ((net 192.168.15.5) or (net 192.168.15.6))

Packet number 1:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: ICMP

Packet number 2:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: ICMP

Packet number 3:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: ICMP

Packet number 4:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: ICMP

Packet number 5:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: ICMP

Packet number 6:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: ICMP

Packet number 7:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: ICMP

Packet number 8:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: ICMP

Packet number 9:
       From: 192.168.15.5
         To: 192.168.15.6
```

.5 pinging .6

Steven Dellamore
Sdelamo
0278562664
Note: I used MC10 NOT MC02

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)”

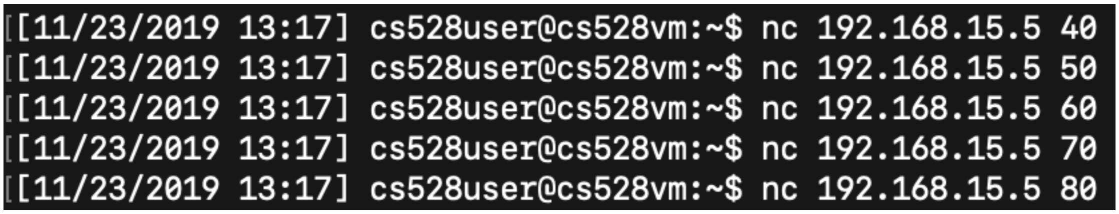
Steven Dellamore
Sdelamo
0278562664
Note: I used MC10 NOT MC02



```
[11/23/2019 13:16] cs528user@cs528vm:~$ 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) and (dst portrange 50-100)

Packet number 1:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: TCP
   Src port: 56852
   Dst port: 50

Packet number 2:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: TCP
   Src port: 40960
   Dst port: 60

Packet number 3:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: TCP
   Src port: 36618
   Dst port: 70

Packet number 4:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: TCP
   Src port: 57078
   Dst port: 80
```

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

Steven Dellamore
Sdelamo
0278562664
Note: I used MC10 NOT MC02



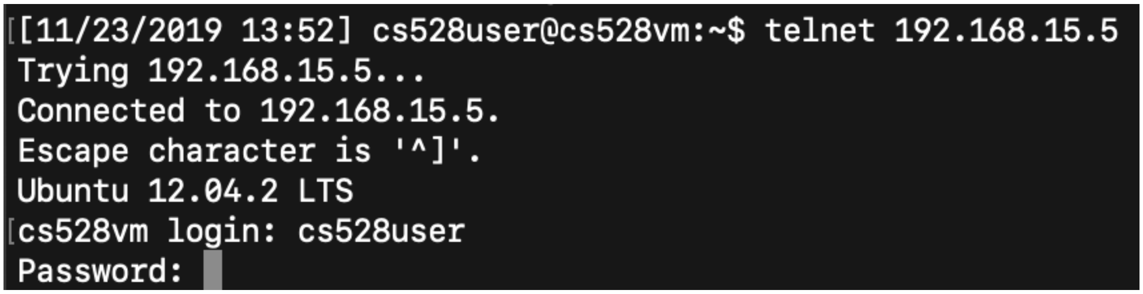
```
[[11/23/2019 13:17] cs528user@cs528vm:~$ nc 192.168.15.5 40
[[11/23/2019 13:17] cs528user@cs528vm:~$ nc 192.168.15.5 50
[[11/23/2019 13:17] cs528user@cs528vm:~$ nc 192.168.15.5 60
[[11/23/2019 13:17] cs528user@cs528vm:~$ nc 192.168.15.5 70
[[11/23/2019 13:17] cs528user@cs528vm:~$ nc 192.168.15.5 80
```

Netcat command used to ping .5

Steven Dellamore
Sdelamo
0278562664
Note: I used MC10 NOT MC02

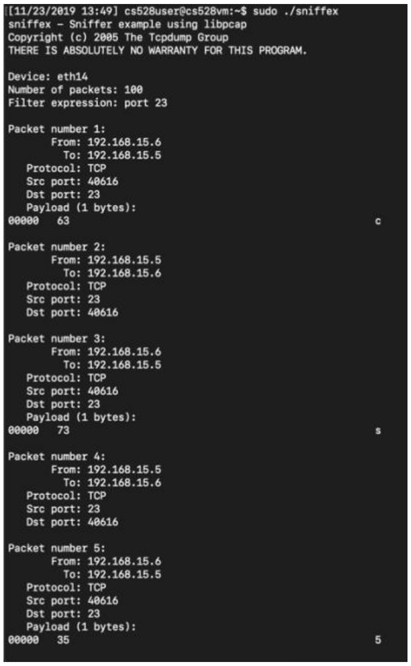
Task 1.c

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)



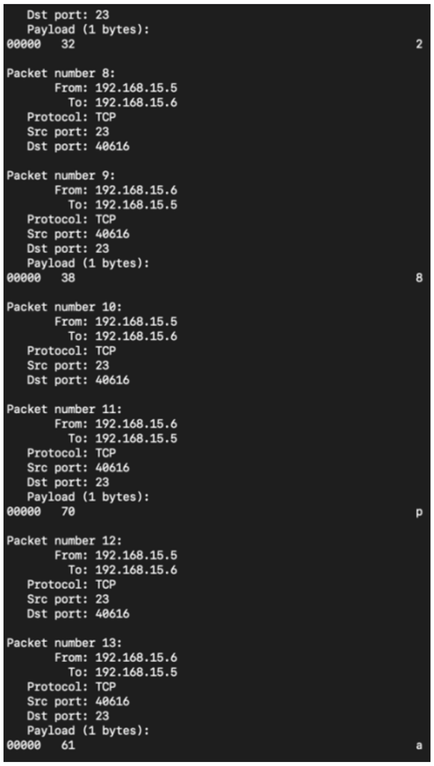
```
[11/23/2019 13:52] cs528user@cs528vm:~$ telnet 192.168.15.5
Trying 192.168.15.5...
Connected to 192.168.15.5.
Escape character is '^]'.
Ubuntu 12.04.2 LTS
cs528vm login: cs528user
Password: 
```

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



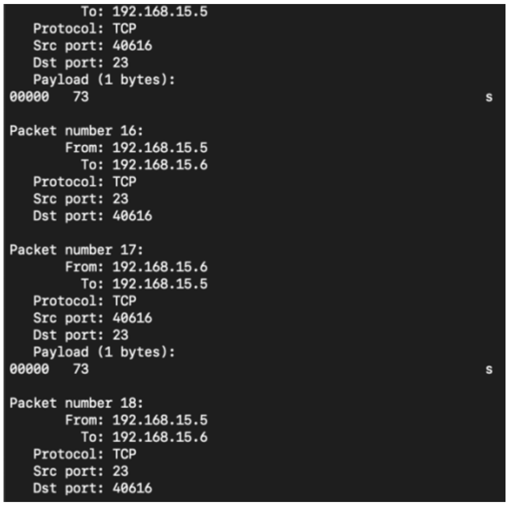
```
[11/23/2019 13:49] cs528user@cs528vm:~$ sudo ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth14
Number of packets: 100
Filter expression: port 23

Packet number 1:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: TCP
   Src port: 40616
   Dst port: 23
   Payload (1 bytes):
00000   63                                                      c

Packet number 2:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: TCP
   Src port: 23
   Dst port: 40616

Packet number 3:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: TCP
   Src port: 40616
   Dst port: 23
   Payload (1 bytes):
00000   73                                                      s

Packet number 4:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: TCP
   Src port: 23
   Dst port: 40616

Packet number 5:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: TCP
   Src port: 40616
   Dst port: 23
   Payload (1 bytes):
00000   35                                                      5
```

Password figure one (cs5)



```
   Dst port: 23
   Payload (1 bytes):
00000   32                                                    2

Packet number 8:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: TCP
   Src port: 23
   Dst port: 40616

Packet number 9:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: TCP
   Src port: 40616
   Dst port: 23
   Payload (1 bytes):
00000   38                                                    8

Packet number 10:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: TCP
   Src port: 23
   Dst port: 40616

Packet number 11:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: TCP
   Src port: 40616
   Dst port: 23
   Payload (1 bytes):
00000   70                                                    p

Packet number 12:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: TCP
   Src port: 23
   Dst port: 40616

Packet number 13:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: TCP
   Src port: 40616
   Dst port: 23
   Payload (1 bytes):
00000   61                                                    a
```

Password figure 2 (28pa)



```
         To: 192.168.15.5
   Protocol: TCP
   Src port: 40616
   Dst port: 23
   Payload (1 bytes):
00000   73                                                    s

Packet number 16:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: TCP
   Src port: 23
   Dst port: 40616

Packet number 17:
       From: 192.168.15.6
         To: 192.168.15.5
   Protocol: TCP
   Src port: 40616
   Dst port: 23
   Payload (1 bytes):
00000   73                                                    s

Packet number 18:
       From: 192.168.15.5
         To: 192.168.15.6
   Protocol: TCP
   Src port: 23
   Dst port: 40616
```

Password figure 3 (ss)

Steven Dellamore
Sdelamo
0278562664
Note: I used MC10 NOT MC02

Task 2.a:

Compile spoof.c : “gcc -o spoof spoof.c”
Run spoof
“sudo ./spoof 0” ICMP
“sudo ./spoof 1” Ethernet

Steven Dellamore
Sdelamo
0278562664
Note: I used MC10 NOT MC02

Task 2.b:

Spoofing machine: 192.168.15.5

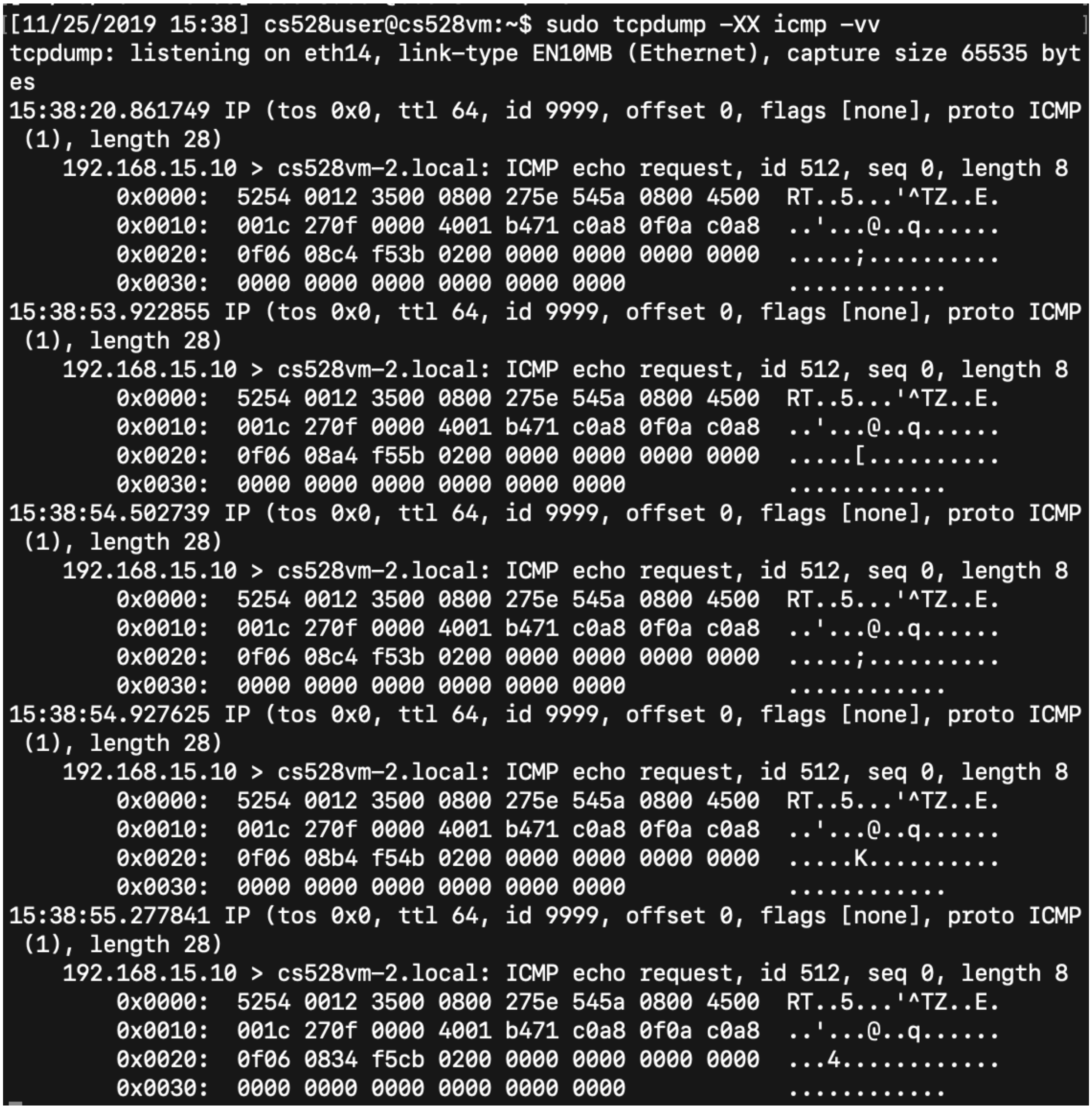
Spoofed machine: 192.168.15.10

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"

Steven Dellamore
Sdelamo
0278562664
Note: I used MC10 NOT MC02



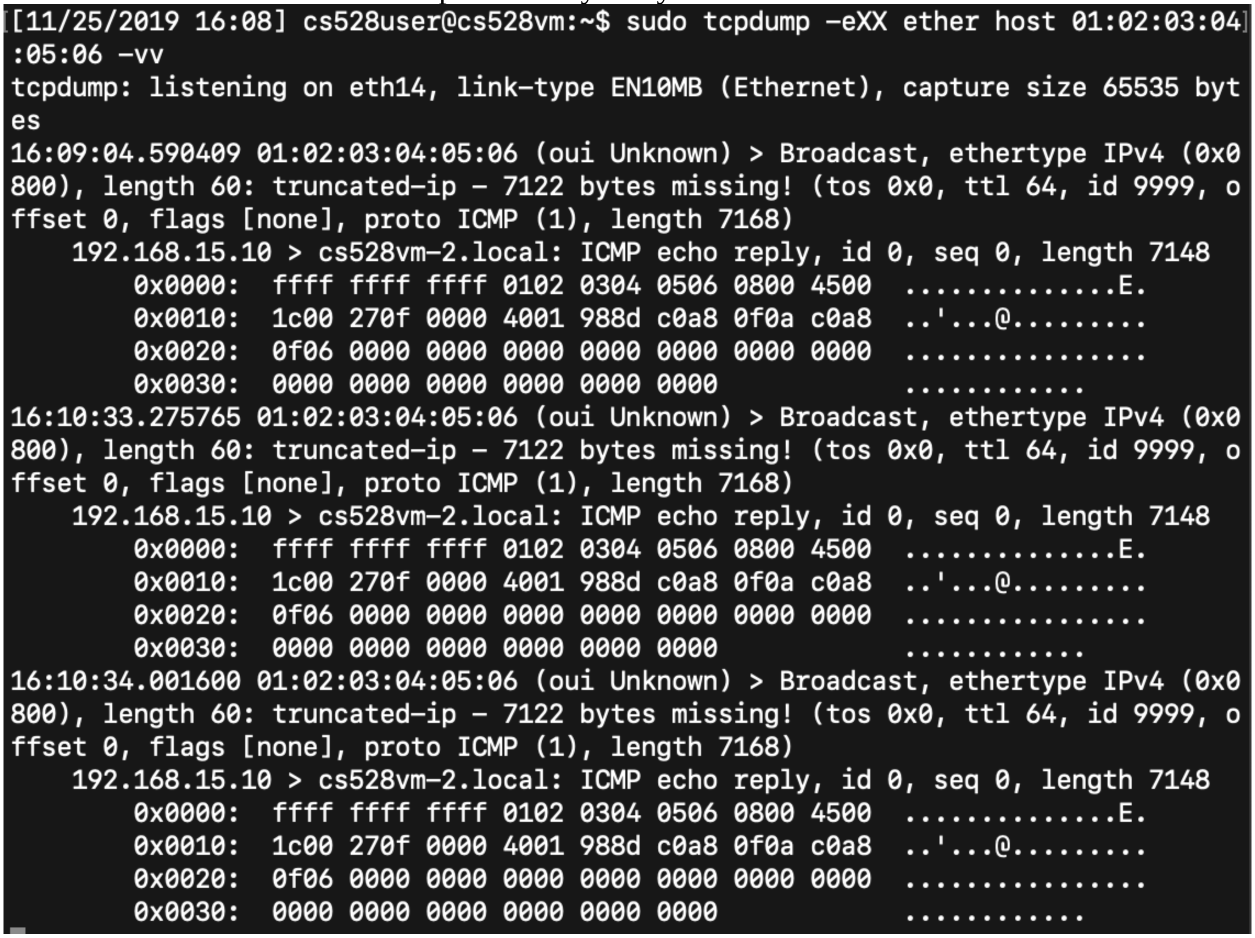
```
[[11/25/2019 15:38] cs528user@cs528vm:~$ sudo tcpdump -XX icmp -vv
tcpdump: listening on eth14, link-type EN10MB (Ethernet), capture size 65535 byt
es
15:38:20.861749 IP (tos 0x0, ttl 64, id 9999, offset 0, flags [none], proto ICMP
 (1), length 28)
    192.168.15.10 > cs528vm-2.local: ICMP echo request, id 512, seq 0, length 8
        0x0000:  5254 0012 3500 0800 275e 545a 0800 4500  RT..5...'^TZ..E.
        0x0010:  001c 270f 0000 4001 b471 c0a8 0f0a c0a8  ..'...@..q......
        0x0020:  0f06 08c4 f53b 0200 0000 0000 0000 0000  .....;..........
        0x0030:  0000 0000 0000 0000 0000 0000            ............
15:38:53.922855 IP (tos 0x0, ttl 64, id 9999, offset 0, flags [none], proto ICMP
 (1), length 28)
    192.168.15.10 > cs528vm-2.local: ICMP echo request, id 512, seq 0, length 8
        0x0000:  5254 0012 3500 0800 275e 545a 0800 4500  RT..5...'^TZ..E.
        0x0010:  001c 270f 0000 4001 b471 c0a8 0f0a c0a8  ..'...@..q......
        0x0020:  0f06 08a4 f55b 0200 0000 0000 0000 0000  .....[..........
        0x0030:  0000 0000 0000 0000 0000 0000            ............
15:38:54.502739 IP (tos 0x0, ttl 64, id 9999, offset 0, flags [none], proto ICMP
 (1), length 28)
    192.168.15.10 > cs528vm-2.local: ICMP echo request, id 512, seq 0, length 8
        0x0000:  5254 0012 3500 0800 275e 545a 0800 4500  RT..5...'^TZ..E.
        0x0010:  001c 270f 0000 4001 b471 c0a8 0f0a c0a8  ..'...@..q......
        0x0020:  0f06 08c4 f53b 0200 0000 0000 0000 0000  .....;..........
        0x0030:  0000 0000 0000 0000 0000 0000            ............
15:38:54.927625 IP (tos 0x0, ttl 64, id 9999, offset 0, flags [none], proto ICMP
 (1), length 28)
    192.168.15.10 > cs528vm-2.local: ICMP echo request, id 512, seq 0, length 8
        0x0000:  5254 0012 3500 0800 275e 545a 0800 4500  RT..5...'^TZ..E.
        0x0010:  001c 270f 0000 4001 b471 c0a8 0f0a c0a8  ..'...@..q......
        0x0020:  0f06 08b4 f54b 0200 0000 0000 0000 0000  .....K..........
        0x0030:  0000 0000 0000 0000 0000 0000            ............
15:38:55.277841 IP (tos 0x0, ttl 64, id 9999, offset 0, flags [none], proto ICMP
 (1), length 28)
    192.168.15.10 > cs528vm-2.local: ICMP echo request, id 512, seq 0, length 8
        0x0000:  5254 0012 3500 0800 275e 545a 0800 4500  RT..5...'^TZ..E.
        0x0010:  001c 270f 0000 4001 b471 c0a8 0f0a c0a8  ..'...@..q......
        0x0020:  0f06 0834 f5cb 0200 0000 0000 0000 0000  ...4............
        0x0030:  0000 0000 0000 0000 0000 0000            ............
```

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

Steven Dellamore
Sdelamo
0278562664
Note: I used MC10 NOT MC02

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.



```
[11/25/2019 16:08] cs528user@cs528vm:~$ sudo tcpdump -eXX ether host 01:02:03:04
:05:06 -vv
tcpdump: listening on eth14, link-type EN10MB (Ethernet), capture size 65535 byt
es
16:09:04.590409 01:02:03:04:05:06 (oui Unknown) > Broadcast, ethertype IPv4 (0x0
800), length 60: truncated-ip - 7122 bytes missing! (tos 0x0, ttl 64, id 9999, o
ffset 0, flags [none], proto ICMP (1), length 7168)
    192.168.15.10 > cs528vm-2.local: ICMP echo reply, id 0, seq 0, length 7148
        0x0000:  ffff ffff ffff 0102 0304 0506 0800 4500  ..............E.
        0x0010:  1c00 270f 0000 4001 988d c0a8 0f0a c0a8  ..'...@.........
        0x0020:  0f06 0000 0000 0000 0000 0000 0000 0000  ................
        0x0030:  0000 0000 0000 0000 0000 0000            ............
16:10:33.275765 01:02:03:04:05:06 (oui Unknown) > Broadcast, ethertype IPv4 (0x0
800), length 60: truncated-ip - 7122 bytes missing! (tos 0x0, ttl 64, id 9999, o
ffset 0, flags [none], proto ICMP (1), length 7168)
    192.168.15.10 > cs528vm-2.local: ICMP echo reply, id 0, seq 0, length 7148
        0x0000:  ffff ffff ffff 0102 0304 0506 0800 4500  ..............E.
        0x0010:  1c00 270f 0000 4001 988d c0a8 0f0a c0a8  ..'...@.........
        0x0020:  0f06 0000 0000 0000 0000 0000 0000 0000  ................
        0x0030:  0000 0000 0000 0000 0000 0000            ............
16:10:34.001600 01:02:03:04:05:06 (oui Unknown) > Broadcast, ethertype IPv4 (0x0
800), length 60: truncated-ip - 7122 bytes missing! (tos 0x0, ttl 64, id 9999, o
ffset 0, flags [none], proto ICMP (1), length 7168)
    192.168.15.10 > cs528vm-2.local: ICMP echo reply, id 0, seq 0, length 7148
        0x0000:  ffff ffff ffff 0102 0304 0506 0800 4500  ..............E.
        0x0010:  1c00 270f 0000 4001 988d c0a8 0f0a c0a8  ..'...@.........
        0x0020:  0f06 0000 0000 0000 0000 0000 0000 0000  ................
        0x0030:  0000 0000 0000 0000 0000 0000            ............
```

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.

Steven Dellamore
Sdelamo
0278562664
Note: I used MC10 NOT MC02

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.