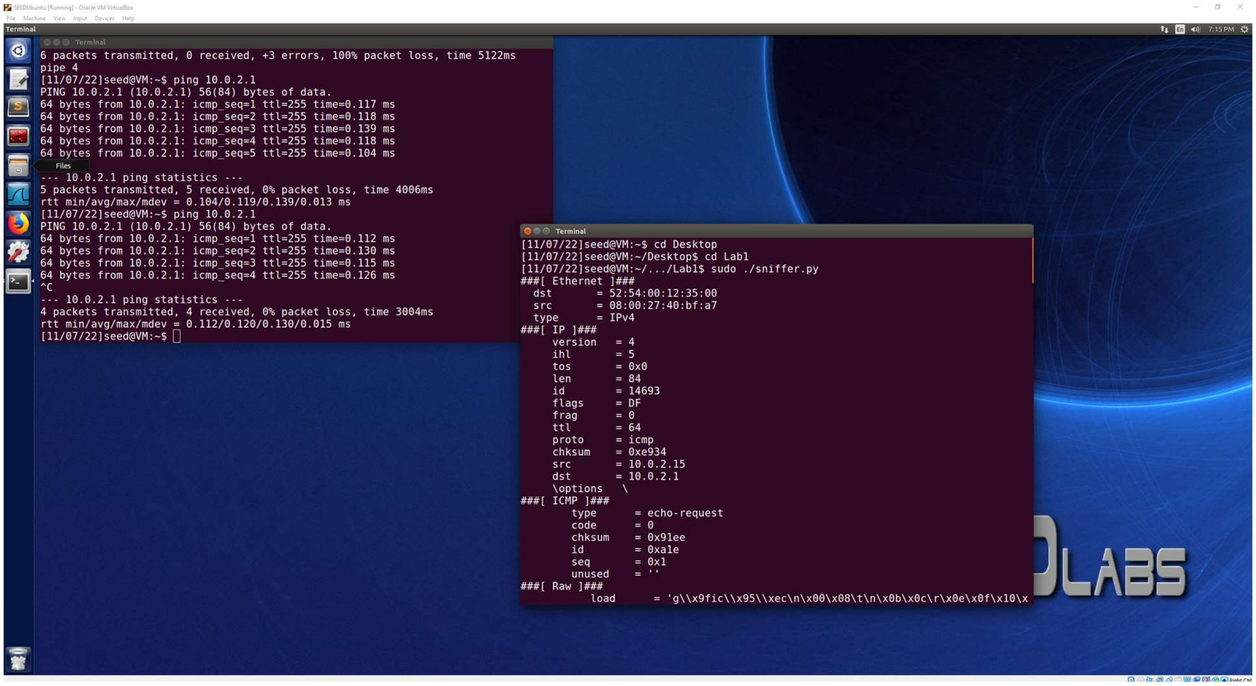
Lab 1
Gunnar Yonker

Task Set 1:

1.1A: Screenshot of packets being captured successfully


SEEDUbuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
Terminal
Files
Terminal
7:15 PM

```
6 packets transmitted, 0 received, +3 errors, 100% packet loss, time 5122ms
pipe 4
[11/07/22]seed@VM:~$ ping 10.0.2.1
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=255 time=0.117 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=255 time=0.118 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=255 time=0.139 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=255 time=0.118 ms
64 bytes from 10.0.2.1: icmp_seq=5 ttl=255 time=0.104 ms

--- 10.0.2.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 0.104/0.119/0.139/0.013 ms
[11/07/22]seed@VM:~$ ping 10.0.2.1
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=255 time=0.112 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=255 time=0.130 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=255 time=0.115 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=255 time=0.126 ms
^C
--- 10.0.2.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.112/0.120/0.130/0.015 ms
[11/07/22]seed@VM:~$
```

```
[11/07/22]seed@VM:~$ cd Desktop
[11/07/22]seed@VM:~/Desktop$ cd Lab1
[11/07/22]seed@VM:~/.../Lab1$ sudo ./sniffer.py
###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
  src       = 08:00:27:40:bf:a7
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 14693
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xe934
     src       = 10.0.2.15
     dst       = 10.0.2.1
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x91ee
        id        = 0xa1e
        seq       = 0x1
        unused    = ''
###[ Raw ]###
           load      = 'g\\x9fic\\x95\\xec\n\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x
```

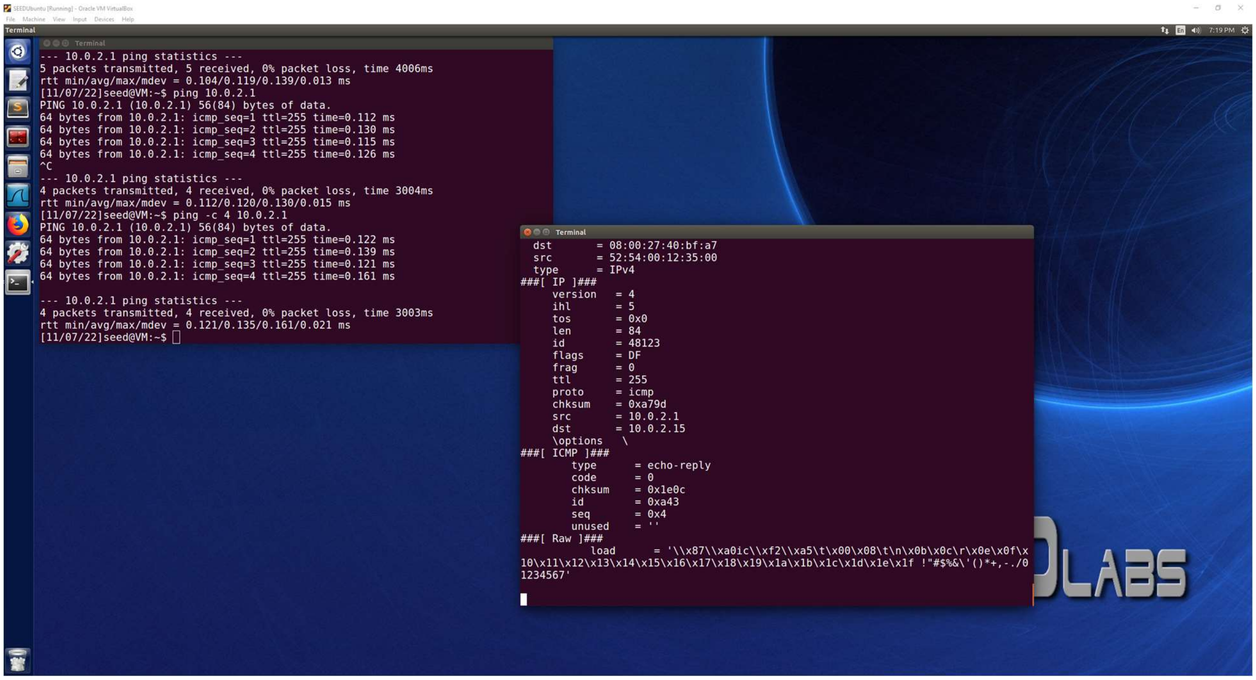
When running the program without the root privilege, the program attempts to run, but then ultimately returns a permission error that is labeled as [Errno 1] Operation not permitted. This simply just means that the action that the program was trying to run, which would be the packet sniffing using scapy was not performed with the right level of permission. This means that the program needs to be run with root privileges.

Lab 1
Gunnar Yonker

1.1B:

Capture only the ICMP packet change:

pkt = sniff(filter='icmp',prn=print_pkt)



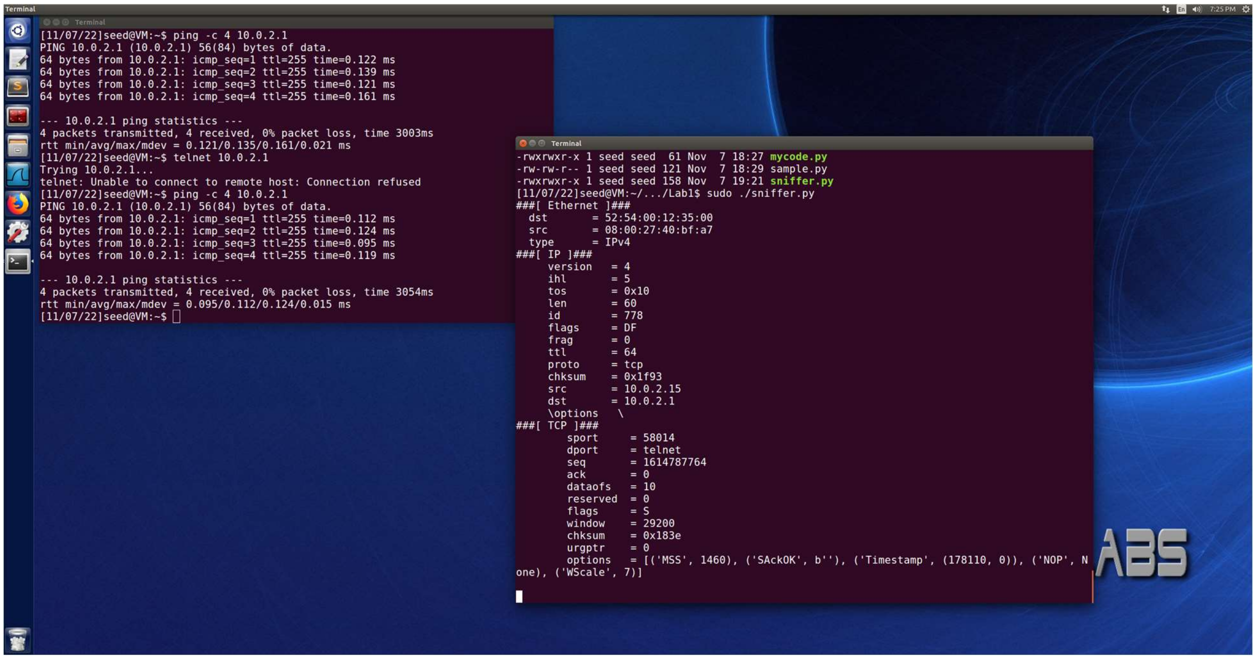
SEEDUbuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
Terminal
Terminal

```
--- 10.0.2.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 0.104/0.119/0.139/0.013 ms
[11/07/22]seed@VM:~$ ping 10.0.2.1
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=255 time=0.112 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=255 time=0.130 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=255 time=0.115 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=255 time=0.126 ms
^C
--- 10.0.2.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.112/0.120/0.130/0.015 ms
[11/07/22]seed@VM:~$ ping -c 4 10.0.2.1
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=255 time=0.122 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=255 time=0.139 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=255 time=0.121 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=255 time=0.161 ms

--- 10.0.2.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.121/0.135/0.161/0.021 ms
[11/07/22]seed@VM:~$ 
```

```
  dst       = 08:00:27:40:bf:a7
  src       = 52:54:00:12:35:00
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 48123
     flags     = DF
     frag      = 0
     ttl       = 255
     proto     = icmp
     chksum    = 0xa79d
     src       = 10.0.2.1
     dst       = 10.0.2.15
     \options   \
###[ ICMP ]###
        type      = echo-reply
        code      = 0
        chksum    = 0x1e0c
        id        = 0xa43
        seq       = 0x4
        unused    = ''
###[ Raw ]###
           load      = '\\x87\\xa0ic\\xf2\\xa5\t\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x
10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./0
1234567'
```

Capture any TCP packet that comes from a particular IP and with a destination port number 23:

pkt = sniff(filter='tcp and dst port 23 and src host 10.0.2.15',prn=print_pkt)



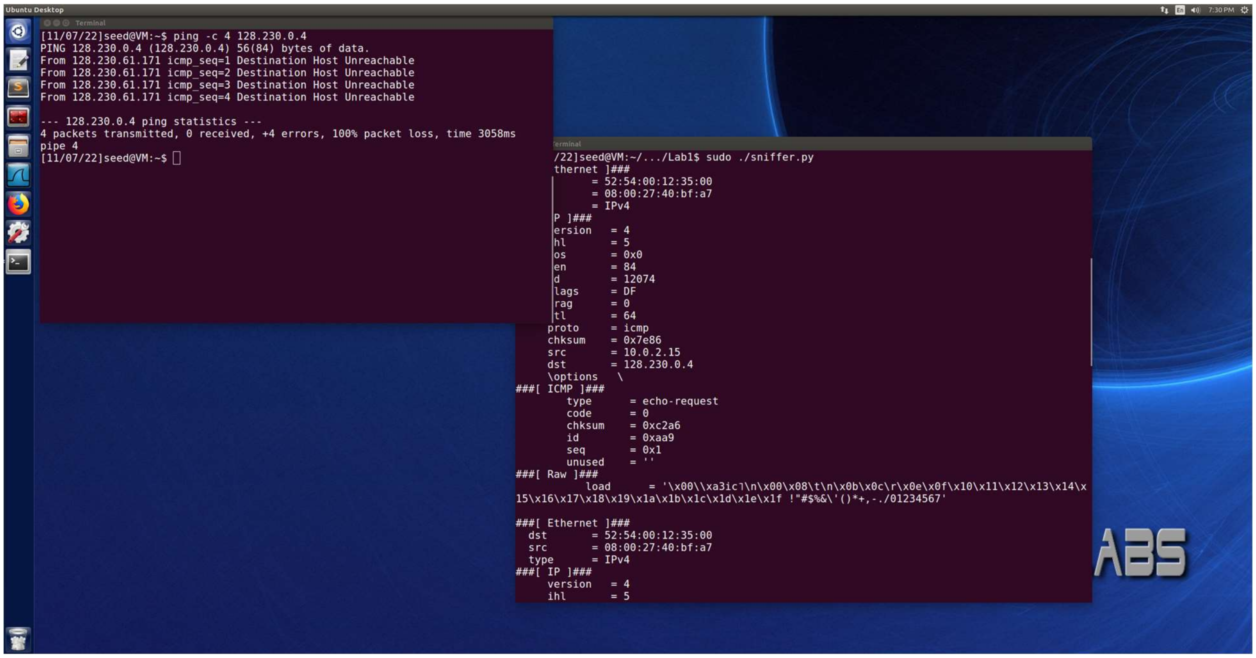
```
[11/07/22]seed@VM:~$ ping -c 4 10.0.2.1
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=255 time=0.122 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=255 time=0.139 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=255 time=0.121 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=255 time=0.161 ms

--- 10.0.2.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.121/0.135/0.161/0.021 ms
[11/07/22]seed@VM:~$ telnet 10.0.2.1
Trying 10.0.2.1...
telnet: Unable to connect to remote host: Connection refused
[11/07/22]seed@VM:~$ ping -c 4 10.0.2.1
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=255 time=0.112 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=255 time=0.124 ms
64 bytes from 10.0.2.1: icmp_seq=3 ttl=255 time=0.095 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=255 time=0.119 ms

--- 10.0.2.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.095/0.112/0.124/0.015 ms
[11/07/22]seed@VM:~$ 
```

```
-rwxrwxr-x 1 seed seed  61 Nov  7 18:27 mycode.py
-rw-rw-r-- 1 seed seed 121 Nov  7 18:29 sample.py
-rwxrwxr-x 1 seed seed 158 Nov  7 19:21 sniffer.py
[11/07/22]seed@VM:~/.../Lab1$ sudo ./sniffer.py
###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
  src       = 08:00:27:40:bf:a7
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 60
     id        = 778
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x1f93
     src       = 10.0.2.15
     dst       = 10.0.2.1
     \options   \
###[ TCP ]###
        sport     = 58014
        dport     = telnet
        seq       = 1614787764
        ack       = 0
        dataofs   = 10
        reserved  = 0
        flags     = S
        window    = 29200
        chksum    = 0x183e
        urgptr    = 0
        options   = [('MSS', 1460), ('SAckOK', b''), ('Timestamp', (178110, 0)), ('NOP', N
one), ('WScale', 7)]
```

Lab 1
Gunnar Yonker

Capture packets come from or to go to a particular subnet. You can pick any subnet, such as 128.230.0.0/16; you should not pick the subnet that your VM is attached to:

pkt = sniff(filter='dst net 128.230.0.0/16',prn=print_pkt)



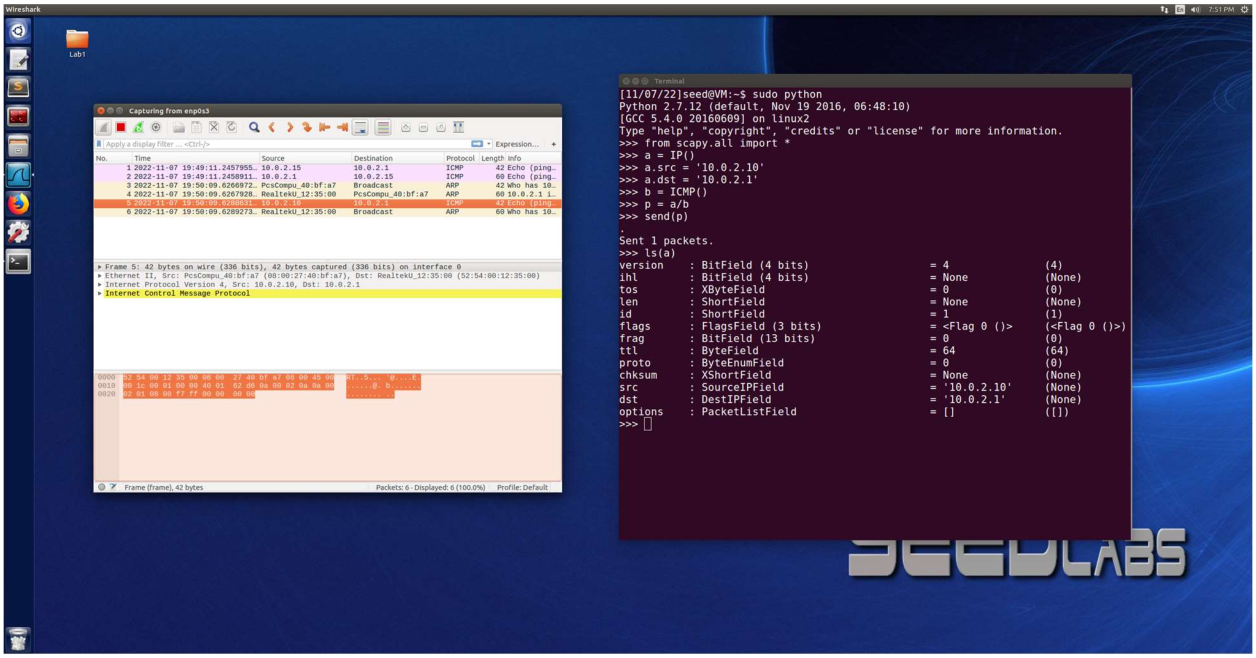
```
[11/07/22]seed@VM:~$ ping -c 4 128.230.0.4
PING 128.230.0.4 (128.230.0.4) 56(84) bytes of data.
From 128.230.61.171 icmp_seq=1 Destination Host Unreachable
From 128.230.61.171 icmp_seq=2 Destination Host Unreachable
From 128.230.61.171 icmp_seq=3 Destination Host Unreachable
From 128.230.61.171 icmp_seq=4 Destination Host Unreachable

--- 128.230.0.4 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3058ms
pipe 4
[11/07/22]seed@VM:~$
```

```
/22]seed@VM:~/.../Lab1$ sudo ./sniffer.py
thernet ]###
        = 52:54:00:12:35:00
        = 08:00:27:40:bf:a7
        = IPv4
P ]###
ersion   = 4
hl       = 5
os       = 0x0
en       = 84
d        = 12074
lags     = DF
rag      = 0
tl       = 64
     proto     = icmp
     chksum    = 0x7e86
     src       = 10.0.2.15
     dst       = 128.230.0.4
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xc2a6
        id        = 0xaa9
        seq       = 0x1
        unused    = ''
###[ Raw ]###
           load      = '\x00\\xa3ic7\n\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x
15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 52:54:00:12:35:00
  src       = 08:00:27:40:bf:a7
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
```

1.2

Original: 10.0.2.15

Spoofed example: 10.0.2.10



No.	Time	Source	Destination	Protocol	Length	Info
1	2022-11-07 19:49:11.2457955…	10.0.2.15	10.0.2.1	ICMP	42	Echo (ping…
2	2022-11-07 19:49:11.2458911…	10.0.2.1	10.0.2.15	ICMP	60	Echo (ping…
3	2022-11-07 19:50:09.6266972…	PcsCompu_40:bf:a7	Broadcast	ARP	42	Who has 10…
4	2022-11-07 19:50:09.6267928…	RealtekU_12:35:00	PcsCompu_40:bf:a7	ARP	60	10.0.2.1 i…
5	2022-11-07 19:50:09.6288631…	10.0.2.10	10.0.2.1	ICMP	42	Echo (ping…
6	2022-11-07 19:50:09.6289273…	RealtekU_12:35:00	Broadcast	ARP	60	Who has 10…

```
▸ Frame 5: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_40:bf:a7 (08:00:27:40:bf:a7), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
▸ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.2.1
▸ Internet Control Message Protocol

0000  52 54 00 12 35 00 08 00  27 40 bf a7 08 00 45 00   RT..5... '@....E.
0010  00 1c 00 01 00 00 40 01  62 d6 0a 00 02 0a 0a 00   ......@. b.......
0020  02 01 08 00 f7 ff 00 00  00 00                     ........ ..

Frame (frame), 42 bytes          Packets: 6 · Displayed: 6 (100.0%)     Profile: Default
```

```
[11/07/22]seed@VM:~$ sudo python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> a = IP()
>>> a.src = '10.0.2.10'
>>> a.dst = '10.0.2.1'
>>> b = ICMP()
>>> p = a/b
>>> send(p)
.
Sent 1 packets.
>>> ls(a)
version    : BitField (4 bits)                     = 4               (4)
ihl        : BitField (4 bits)                     = None            (None)
tos        : XByteField                            = 0               (0)
len        : ShortField                            = None            (None)
id         : ShortField                            = 1               (1)
flags      : FlagsField (3 bits)                   = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField (13 bits)                    = 0               (0)
ttl        : ByteField                             = 64              (64)
proto      : ByteEnumField                         = 0               (0)
chksum     : XShortField                           = None            (None)
src        : SourceIPField                         = '10.0.2.10'     (None)
dst        : DestIPField                           = '10.0.2.1'      (None)
options    : PacketListField                       = []              ([])
>>>
```

Lab 1
Gunnar Yonker

Task 1.3:

Destination: 142.250.191.132

TTL = 1 – 10.0.2.1

TTL = 2 - 192.169.1.1

TTL = 3 – 142.250.191.132

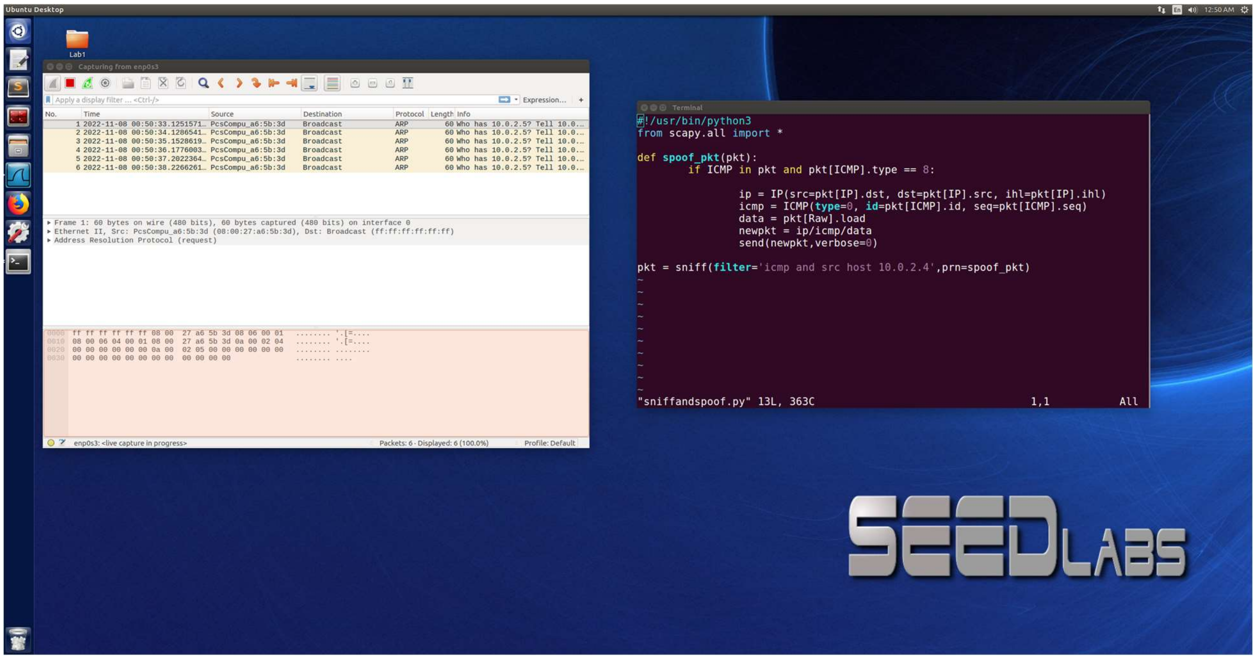
Echo response received

Task 1.4:

VM #1 : Running the program IP: 10.0.2.15

VM #2: IP: 10.0.2.4 (shut off)

VM #3: IP: 10.0.2.5 (trying to ping VM #2, looking for response from VM #1 spoofing as VM #2 responding)



No.	Time	Source	Destination	Protocol	Length	Info
1	2022-11-08 00:50:33.1251571…	PcsCompu_a6:5b:3d	Broadcast	ARP	60	Who has 10.0.2.5? Tell 10.0.…
2	2022-11-08 00:50:34.1286541…	PcsCompu_a6:5b:3d	Broadcast	ARP	60	Who has 10.0.2.5? Tell 10.0.…
3	2022-11-08 00:50:35.1528619…	PcsCompu_a6:5b:3d	Broadcast	ARP	60	Who has 10.0.2.5? Tell 10.0.…
4	2022-11-08 00:50:36.1776003…	PcsCompu_a6:5b:3d	Broadcast	ARP	60	Who has 10.0.2.5? Tell 10.0.…
5	2022-11-08 00:50:37.2022364…	PcsCompu_a6:5b:3d	Broadcast	ARP	60	Who has 10.0.2.5? Tell 10.0.…
6	2022-11-08 00:50:38.2266261…	PcsCompu_a6:5b:3d	Broadcast	ARP	60	Who has 10.0.2.5? Tell 10.0.…

```
▸ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_a6:5b:3d (08:00:27:a6:5b:3d), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff 08 00  27 a6 5b 3d 08 06 00 01   ........ '.[=....
0010  08 00 06 04 00 01 08 00  27 a6 5b 3d 0a 00 02 04   ........ '.[=....
0020  00 00 00 00 00 00 0a 00  02 05 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....
```

enp0s3: <live capture in progress>   Packets: 6 · Displayed: 6 (100.0%)   Profile: Default

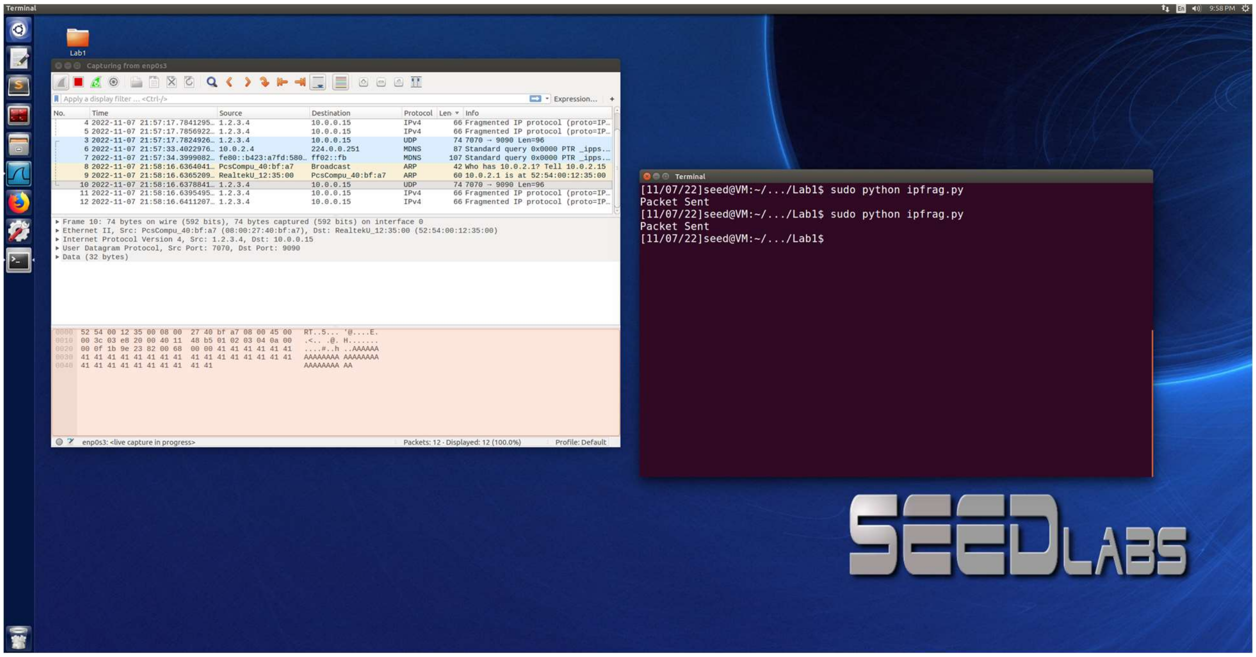
```
#!/usr/bin/python3
from scapy.all import *

def spoof_pkt(pkt):
        if ICMP in pkt and pkt[ICMP].type == 8:

                ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
                icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
                data = pkt[Raw].load
                newpkt = ip/icmp/data
                send(newpkt,verbose=0)

pkt = sniff(filter='icmp and src host 10.0.2.4',prn=spoof_pkt)
~
~
~
~
~
~
~
~
~
~
"sniffandspoof.py" 13L, 363C                              1,1           All
```

I am not entirely sure where I went wrong with trying to sniff and then spoof the IP to respond, I could not find where I was going wrong this problem, and this is the program that I was attempting to use. The pings were being broadcasted and I am disappointed in myself that I cannot figure out where I went wrong on this task.

Lab 1
Gunnar Yonker

Task 2: IP fragmentation and ICMP redirect

1.a: Each fragment contains 32 bytes of data, total length is 96 bytes



```
No. Time                         Source                 Destination         Protocol Len  Info
  4 2022-11-07 21:57:17.7841295… 1.2.3.4                10.0.0.15           IPv4      66 Fragmented IP protocol (proto=IP…
  5 2022-11-07 21:57:17.7856922… 1.2.3.4                10.0.0.15           IPv4      66 Fragmented IP protocol (proto=IP…
  3 2022-11-07 21:57:17.7824926… 1.2.3.4                10.0.0.15           UDP       74 7070 → 9090 Len=96
  6 2022-11-07 21:57:33.4022976… 10.0.2.4               224.0.0.251         MDNS      87 Standard query 0x0000 PTR _ipps.…
  7 2022-11-07 21:57:34.3999082… fe80::b423:a7fd:580…   ff02::fb            MDNS     107 Standard query 0x0000 PTR _ipps.…
  8 2022-11-07 21:58:16.6364041… PcsCompu_40:bf:a7      Broadcast           ARP       42 Who has 10.0.2.1? Tell 10.0.2.15
  9 2022-11-07 21:58:16.6365209… RealtekU_12:35:00      PcsCompu_40:bf:a7   ARP       60 10.0.2.1 is at 52:54:00:12:35:00
 10 2022-11-07 21:58:16.6378841… 1.2.3.4                10.0.0.15           UDP       74 7070 → 9090 Len=96
 11 2022-11-07 21:58:16.6395495… 1.2.3.4                10.0.0.15           IPv4      66 Fragmented IP protocol (proto=IP…
 12 2022-11-07 21:58:16.6411207… 1.2.3.4                10.0.0.15           IPv4      66 Fragmented IP protocol (proto=IP…

▸ Frame 10: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_40:bf:a7 (08:00:27:40:bf:a7), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
▸ Internet Protocol Version 4, Src: 1.2.3.4, Dst: 10.0.0.15
▸ User Datagram Protocol, Src Port: 7070, Dst Port: 9090
▸ Data (32 bytes)

0000  52 54 00 12 35 00 08 00  27 40 bf a7 08 00 45 00   RT..5... '@....E.
0010  00 3c 03 e8 20 00 40 11  48 b5 01 02 03 04 0a 00   .<.. .@. H.......
0020  00 0f 1b 9e 23 82 00 68  00 00 41 41 41 41 41 41   ....#..h ..AAAAAA
0030  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0040  41 41 41 41 41 41 41 41  41 41                     AAAAAAAA AA

enp0s3: <live capture in progress>        Packets: 12 · Displayed: 12 (100.0%)        Profile: Default
```

```
[11/07/22]seed@VM:~/.../Lab1$ sudo python ipfrag.py
Packet Sent
[11/07/22]seed@VM:~/.../Lab1$ sudo python ipfrag.py
Packet Sent
[11/07/22]seed@VM:~/.../Lab1$
```

1.b:

First scenario:

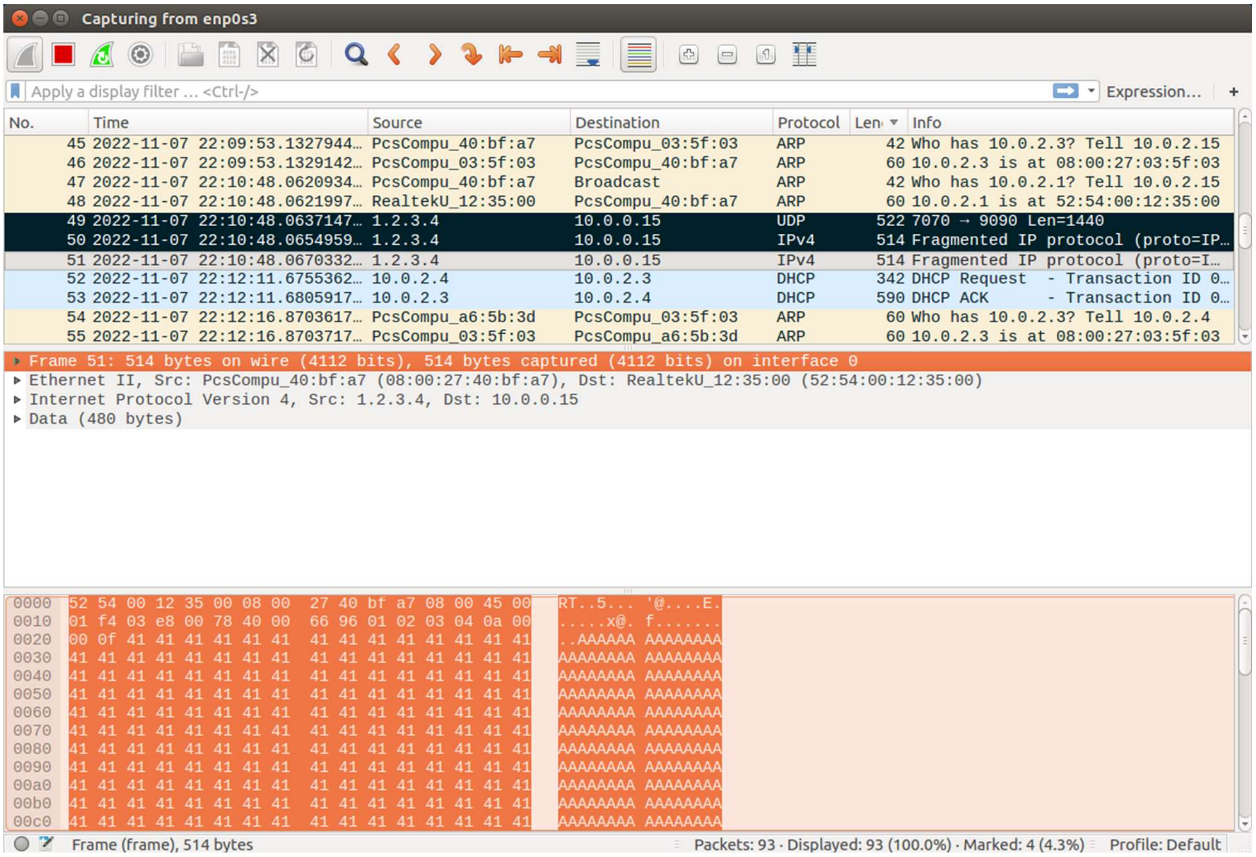
1440 bytes of data in 3 total fragments

Original off-set of the second fragment is 60, new value for the second fragment off-set to cause overlap will be 50. This will cause 80 bytes of data to overlap between fragments 1 and 2.

In WireShark, it still shows the 3 packets coming through containing 480 bytes per packet and a length of 1440 as shown below. However, this packet would now have 80 bytes between fragment 1 and 2 that are overlapped during the re-assembly of the packet so the information that would be in an overlapped fragment would be messed up during the reassembly.

If the second packet were sent over the first packet, the bytes would still be overlapped. However, whichever fragment was sent and received first would have the overlapping 80 bytes overwritten by that next fragment due to the off set being incorrectly placed.

Lab 1
Gunnar Yonker


Capturing from enp0s3

```
No.  Time                          Source              Destination         Protocol  Len  Info
  45 2022-11-07 22:09:53.1327944…  PcsCompu_40:bf:a7   PcsCompu_03:5f:03   ARP        42  Who has 10.0.2.3? Tell 10.0.2.15
  46 2022-11-07 22:09:53.1329142…  PcsCompu_03:5f:03   PcsCompu_40:bf:a7   ARP        60  10.0.2.3 is at 08:00:27:03:5f:03
  47 2022-11-07 22:10:48.0620934…  PcsCompu_40:bf:a7   Broadcast           ARP        42  Who has 10.0.2.1? Tell 10.0.2.15
  48 2022-11-07 22:10:48.0621997…  RealtekU_12:35:00   PcsCompu_40:bf:a7   ARP        60  10.0.2.1 is at 52:54:00:12:35:00
  49 2022-11-07 22:10:48.0637147…  1.2.3.4             10.0.0.15           UDP       522  7070 → 9090 Len=1440
  50 2022-11-07 22:10:48.0654959…  1.2.3.4             10.0.0.15           IPv4      514  Fragmented IP protocol (proto=IP…
  51 2022-11-07 22:10:48.0670332…  1.2.3.4             10.0.0.15           IPv4      514  Fragmented IP protocol (proto=I…
  52 2022-11-07 22:12:11.6755362…  10.0.2.4            10.0.2.3            DHCP      342  DHCP Request  - Transaction ID 0…
  53 2022-11-07 22:12:11.6805917…  10.0.2.3            10.0.2.4            DHCP      590  DHCP ACK      - Transaction ID 0…
  54 2022-11-07 22:12:16.8703617…  PcsCompu_a6:5b:3d   PcsCompu_03:5f:03   ARP        60  Who has 10.0.2.3? Tell 10.0.2.4
  55 2022-11-07 22:12:16.8703717…  PcsCompu_03:5f:03   PcsCompu_a6:5b:3d   ARP        60  10.0.2.3 is at 08:00:27:03:5f:03
```

```
▸ Frame 51: 514 bytes on wire (4112 bits), 514 bytes captured (4112 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_40:bf:a7 (08:00:27:40:bf:a7), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
▸ Internet Protocol Version 4, Src: 1.2.3.4, Dst: 10.0.0.15
▸ Data (480 bytes)
```

```
0000  52 54 00 12 35 00 08 00  27 40 bf a7 08 00 45 00   RT..5... '@....E.
0010  01 f4 03 e8 00 78 40 00  66 96 01 02 03 04 0a 00   .....x@. f.......
0020  00 0f 41 41 41 41 41 41  41 41 41 41 41 41 41 41   ..AAAAAA AAAAAAAA
0030  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0040  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0050  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0060  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0070  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0080  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0090  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
00a0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
00b0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
00c0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
```

Frame (frame), 514 bytes   Packets: 93 · Displayed: 93 (100.0%) · Marked: 4 (4.3%)   Profile: Default

Second scenario:

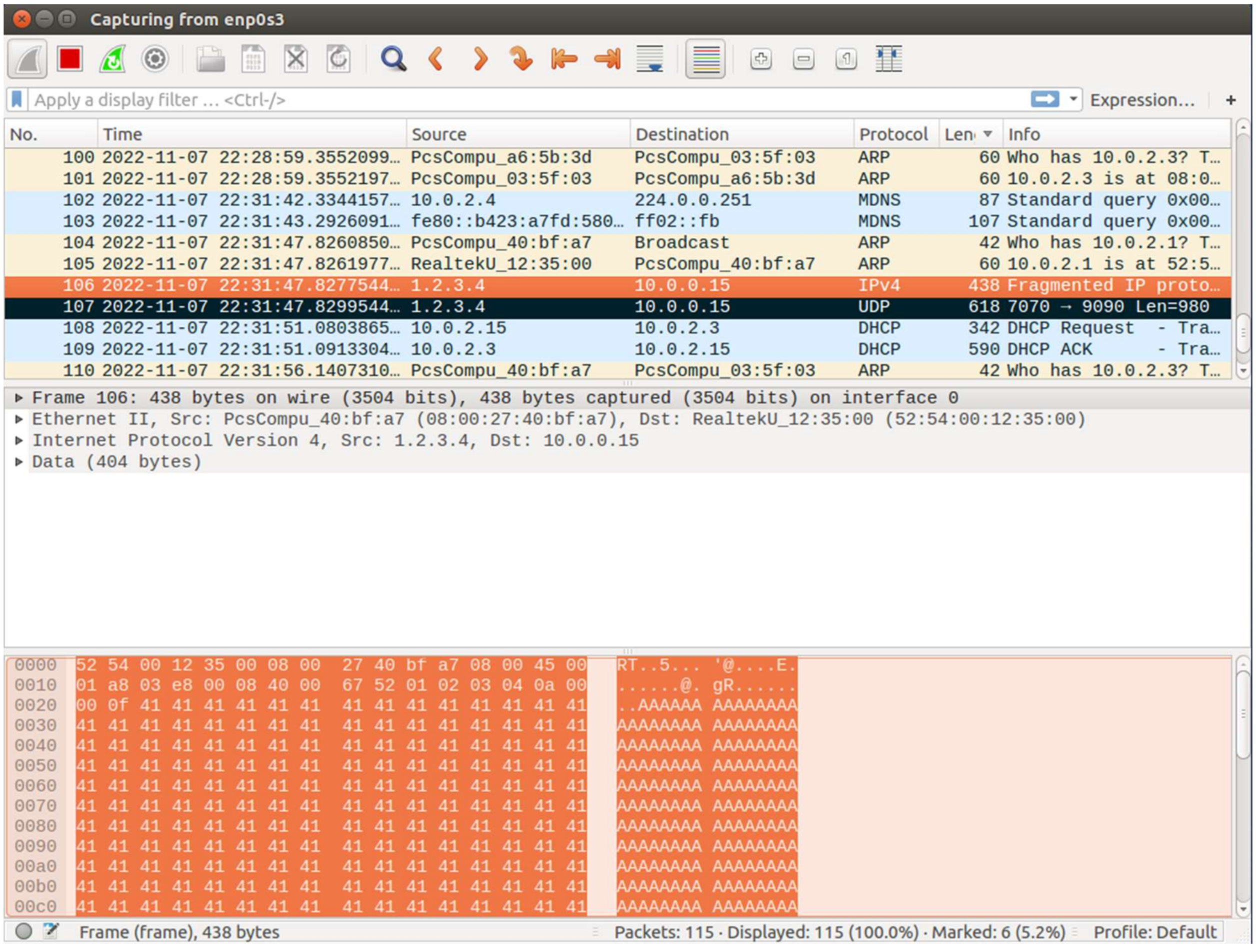
980 bytes of data in 2 total fragments

First fragment is 576, second fragment is 404 bytes with an off set of 72 for correct fragmentation re-assembly. The off-set for the second fragment will be 1 now to cause the second fragment to be completely enclosed in the first fragment.

When the fragments were sent as the first fragment being sent first and the second smaller fragment sent second, both of the fragments can be observed as being sent through WireShark with the correct length and datagram sizes on them. There was no error when being sent. However, when the first fragment arrives it will be set and then the second fragment arrives and with an off-set of 1, the 404 bytes of the second fragment will have an overlap of 404 causing 404 bytes of the first fragment to be overwritten.

When sending the second fragment first, and the first fragment second. The second fragment is received first and the first fragment received second. The same situation as described above will occur with the overlap, but the fragments both have the same ID and the first fragment has a flag of 1 and the second fragment has a flag of 0, so they will still be assembled with each other, but the overlapping will still occur overwriting data that is then lost due to the off-set being incorrect.

Lab 1
Gunnar Yonker



No.	Time	Source	Destination	Protocol	Len	Info
100	2022-11-07 22:28:59.3552099…	PcsCompu_a6:5b:3d	PcsCompu_03:5f:03	ARP	60	Who has 10.0.2.3? T…
101	2022-11-07 22:28:59.3552197…	PcsCompu_03:5f:03	PcsCompu_a6:5b:3d	ARP	60	10.0.2.3 is at 08:0…
102	2022-11-07 22:31:42.3344157…	10.0.2.4	224.0.0.251	MDNS	87	Standard query 0x00…
103	2022-11-07 22:31:43.2926091…	fe80::b423:a7fd:580…	ff02::fb	MDNS	107	Standard query 0x00…
104	2022-11-07 22:31:47.8260850…	PcsCompu_40:bf:a7	Broadcast	ARP	42	Who has 10.0.2.1? T…
105	2022-11-07 22:31:47.8261977…	RealtekU_12:35:00	PcsCompu_40:bf:a7	ARP	60	10.0.2.1 is at 52:5…
106	2022-11-07 22:31:47.8277544…	1.2.3.4	10.0.0.15	IPv4	438	Fragmented IP proto…
107	2022-11-07 22:31:47.8299544…	1.2.3.4	10.0.0.15	UDP	618	7070 → 9090 Len=980
108	2022-11-07 22:31:51.0803865…	10.0.2.15	10.0.2.3	DHCP	342	DHCP Request - Tra…
109	2022-11-07 22:31:51.0913304…	10.0.2.3	10.0.2.15	DHCP	590	DHCP ACK - Tra…
110	2022-11-07 22:31:56.1407310…	PcsCompu_40:bf:a7	PcsCompu_03:5f:03	ARP	42	Who has 10.0.2.3? T…

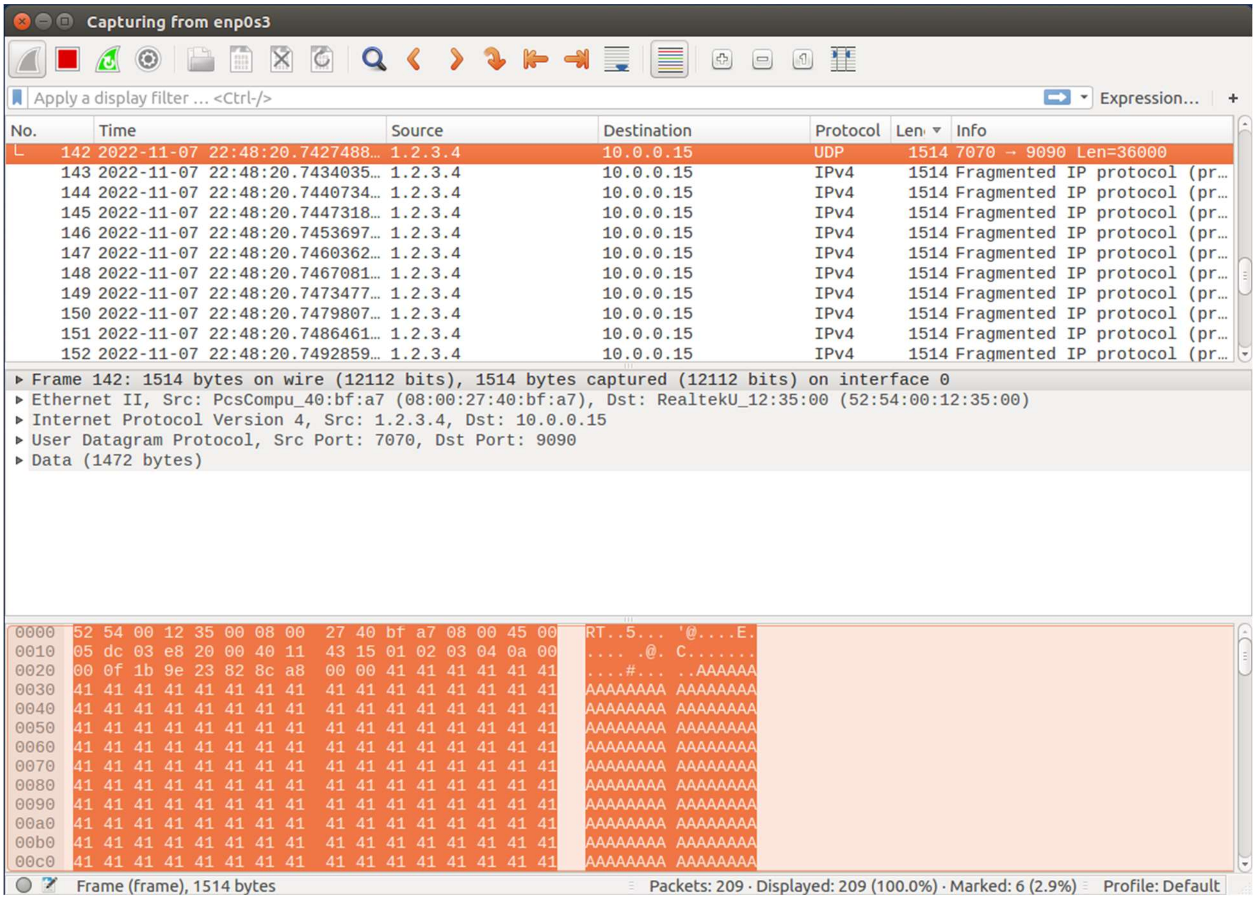
```
▸ Frame 106: 438 bytes on wire (3504 bits), 438 bytes captured (3504 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_40:bf:a7 (08:00:27:40:bf:a7), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
▸ Internet Protocol Version 4, Src: 1.2.3.4, Dst: 10.0.0.15
▸ Data (404 bytes)
```

```
0000  52 54 00 12 35 00 08 00  27 40 bf a7 08 00 45 00   RT..5... '@....E.
0010  01 a8 03 e8 00 08 40 00  67 52 01 02 03 04 0a 00   ......@. gR......
0020  00 0f 41 41 41 41 41 41  41 41 41 41 41 41 41 41   ..AAAAAA AAAAAAAA
0030  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0040  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0050  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0060  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0070  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0080  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0090  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
00a0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
00b0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
00c0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
```

Frame (frame), 438 bytes   Packets: 115 · Displayed: 115 (100.0%) · Marked: 6 (5.2%)   Profile: Default

Task 1.c:

2^16 is 65536, so if we create a packet that was 66000 bytes it would exceed the maximal size of an IP packet. This can be fragmented down to become possible. If we assumed an MTU of 1500 which is typical, there would be 44 fragments that needed to be sent. When attempting to send this package, each fragment would contain a payload of 1480 other than the first fragment and the 44th fragment which would contain 860 bytes. This would allow a packet of size 66000 bytes to be sent and then reassembled.



```
No.  Time                          Source     Destination  Protocol  Len  Info
└ 142 2022-11-07 22:48:20.7427488… 1.2.3.4    10.0.0.15    UDP       1514 7070 → 9090 Len=36000
  143 2022-11-07 22:48:20.7434035… 1.2.3.4    10.0.0.15    IPv4      1514 Fragmented IP protocol (pr…
  144 2022-11-07 22:48:20.7440734… 1.2.3.4    10.0.0.15    IPv4      1514 Fragmented IP protocol (pr…
  145 2022-11-07 22:48:20.7447318… 1.2.3.4    10.0.0.15    IPv4      1514 Fragmented IP protocol (pr…
  146 2022-11-07 22:48:20.7453697… 1.2.3.4    10.0.0.15    IPv4      1514 Fragmented IP protocol (pr…
  147 2022-11-07 22:48:20.7460362… 1.2.3.4    10.0.0.15    IPv4      1514 Fragmented IP protocol (pr…
  148 2022-11-07 22:48:20.7467081… 1.2.3.4    10.0.0.15    IPv4      1514 Fragmented IP protocol (pr…
  149 2022-11-07 22:48:20.7473477… 1.2.3.4    10.0.0.15    IPv4      1514 Fragmented IP protocol (pr…
  150 2022-11-07 22:48:20.7479807… 1.2.3.4    10.0.0.15    IPv4      1514 Fragmented IP protocol (pr…
  151 2022-11-07 22:48:20.7486461… 1.2.3.4    10.0.0.15    IPv4      1514 Fragmented IP protocol (pr…
  152 2022-11-07 22:48:20.7492859… 1.2.3.4    10.0.0.15    IPv4      1514 Fragmented IP protocol (pr…
```

```
▸ Frame 142: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_40:bf:a7 (08:00:27:40:bf:a7), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
▸ Internet Protocol Version 4, Src: 1.2.3.4, Dst: 10.0.0.15
▸ User Datagram Protocol, Src Port: 7070, Dst Port: 9090
▸ Data (1472 bytes)
```

```
0000  52 54 00 12 35 00 08 00  27 40 bf a7 08 00 45 00   RT..5... '@....E.
0010  05 dc 03 e8 20 00 40 11  43 15 01 02 03 04 0a 00   .... .@. C.......
0020  00 0f 1b 9e 23 82 8c a8  00 00 41 41 41 41 41 41   ....#... ..AAAAAA
0030  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0040  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0050  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0060  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0070  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0080  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
0090  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
00a0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
00b0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
00c0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41   AAAAAAAA AAAAAAAA
```

Frame (frame), 1514 bytes · Packets: 209 · Displayed: 209 (100.0%) · Marked: 6 (2.9%) · Profile: Default

Task 1.d:

When the packets were sent as a fragment, they could not be fully reassembled due to the next part of the fragment never showing up and the fragment staying in the kernel memory until they time out. When sending the packeted fragments, it is incredibly easy to send a lot of packets very fast causing the memory of the victim system to fill up before the IP packets time out. This results in a Dos (denial of service) attack. Observing on Wireshark from the attacking vm, all of the fragments were seen going out to the intended victim, and on the victim vm on Wireshark all of the fragments were being received but was not displaying that the fragment reassembly time was exceeded, so the fragments were able to keep being received at the same rate but not timing out at the same rate. If the victim system did not have a packet filter security in place, this would result in a DoS attack causing the system to shut down.

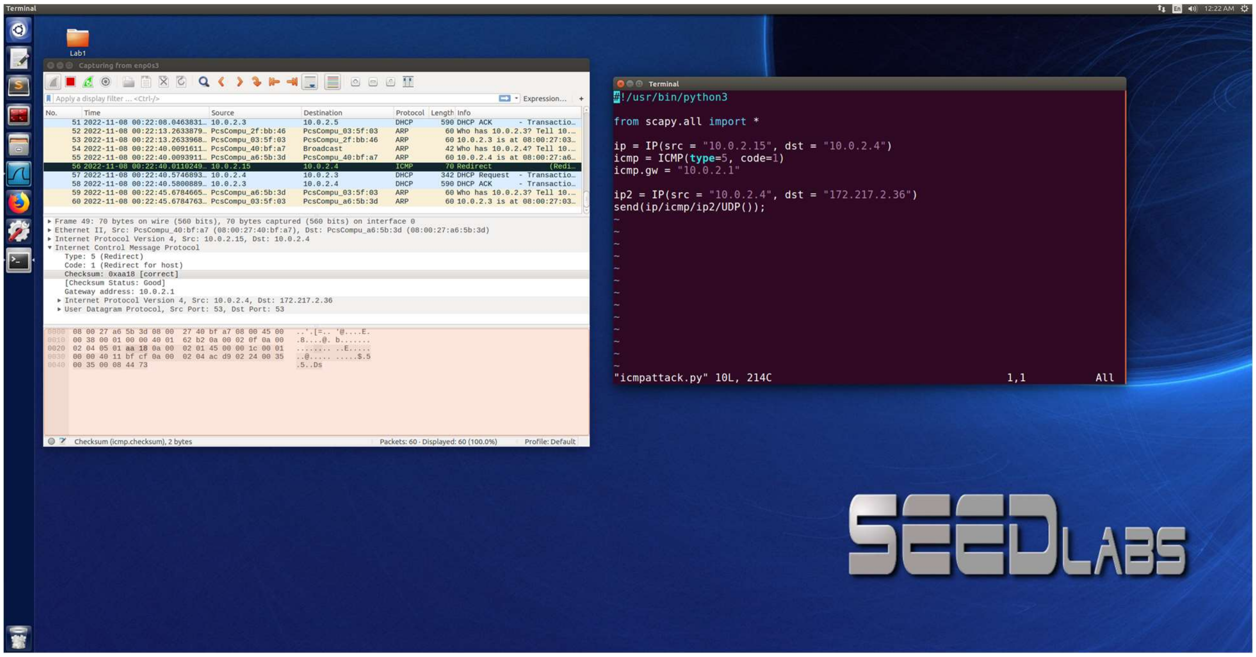
Lab 1
Gunnar Yonker

Task 2: ICMP Redirect Attack

VM(Host A) Victim 10.0.2.4

VM(Host B) Attacker 10.0.2.15

Destination B(outside webserver): 172.217.2.36



```
No.  Time                          Source              Destination         Protocol Length Info
  51 2022-11-08 00:22:08.0463831… 10.0.2.3            10.0.2.5            DHCP      590 DHCP ACK      - Transactio…
  52 2022-11-08 00:22:13.2633879… PcsCompu_2f:bb:46   PcsCompu_03:5f:03   ARP        60 Who has 10.0.2.3? Tell 10.…
  53 2022-11-08 00:22:13.2633968… PcsCompu_03:5f:03   PcsCompu_2f:bb:46   ARP        60 10.0.2.3 is at 08:00:27:03…
  54 2022-11-08 00:22:40.0091611… PcsCompu_40:bf:a7   Broadcast           ARP        42 Who has 10.0.2.4? Tell 10.…
  55 2022-11-08 00:22:40.0093911… PcsCompu_a6:5b:3d   PcsCompu_40:bf:a7   ARP        60 10.0.2.4 is at 08:00:27:a6…
  56 2022-11-08 00:22:40.0110249… 10.0.2.15           10.0.2.4            ICMP       70 Redirect              (Redi…
  57 2022-11-08 00:22:40.5746893… 10.0.2.4            10.0.2.3            DHCP      342 DHCP Request  - Transactio…
  58 2022-11-08 00:22:40.5800889… 10.0.2.3            10.0.2.4            DHCP      590 DHCP ACK      - Transactio…
  59 2022-11-08 00:22:45.6784665… PcsCompu_a6:5b:3d   PcsCompu_03:5f:03   ARP        60 Who has 10.0.2.3? Tell 10.…
  60 2022-11-08 00:22:45.6784763… PcsCompu_03:5f:03   PcsCompu_a6:5b:3d   ARP        60 10.0.2.3 is at 08:00:27:03…

▸ Frame 49: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_40:bf:a7 (08:00:27:40:bf:a7), Dst: PcsCompu_a6:5b:3d (08:00:27:a6:5b:3d)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.4
▾ Internet Control Message Protocol
    Type: 5 (Redirect)
    Code: 1 (Redirect for host)
    Checksum: 0xaa18 [correct]
    [Checksum Status: Good]
    Gateway address: 10.0.2.1
  ▸ Internet Protocol Version 4, Src: 10.0.2.4, Dst: 172.217.2.36
  ▸ User Datagram Protocol, Src Port: 53, Dst Port: 53

0000  08 00 27 a6 5b 3d 08 00  27 40 bf a7 08 00 45 00   ..'.[=.. '@....E.
0010  00 38 00 01 00 00 40 01  62 b2 0a 00 02 0f 0a 00   .8....@. b.......
0020  02 04 05 01 aa 18 0a 00  02 01 45 00 00 1c 00 01   ........ ..E.....
0030  00 00 40 11 bf cf 0a 00  02 04 ac d9 02 24 00 35   ..@..... .....$.5
0040  00 35 00 08 44 73                                  .5..Ds

Checksum (icmp.checksum), 2 bytes          Packets: 60 · Displayed: 60 (100.0%)          Profile: Default
```

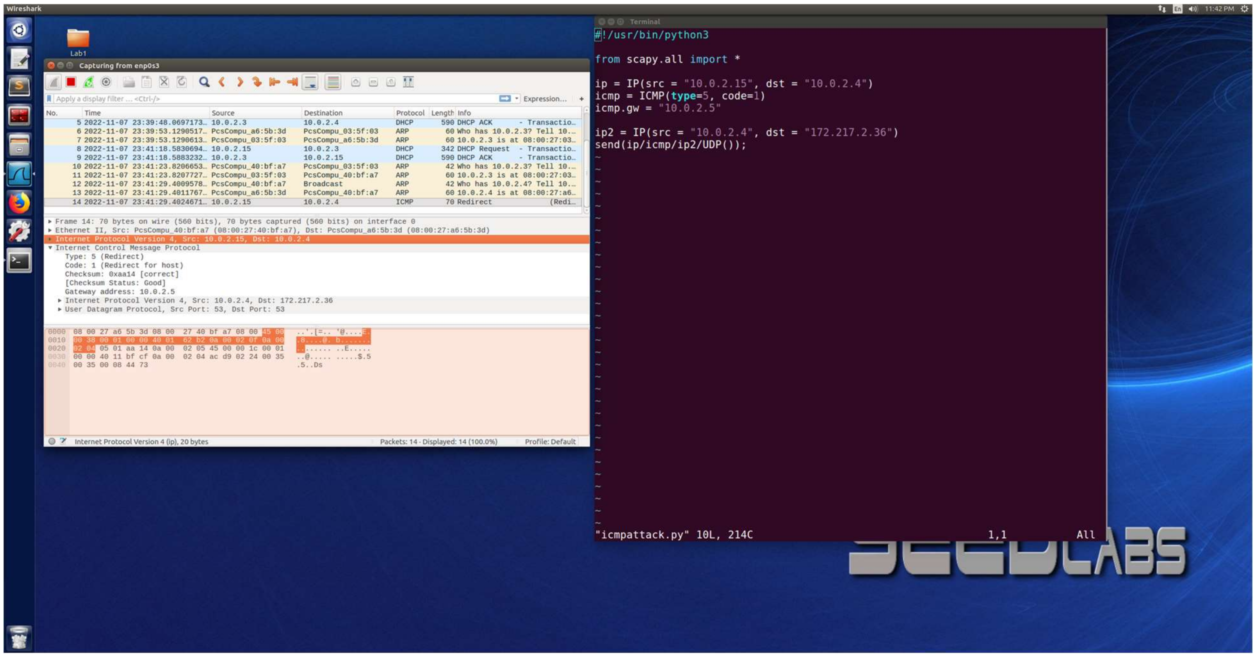
```
#!/usr/bin/python3

from scapy.all import *

ip = IP(src = "10.0.2.15", dst = "10.0.2.4")
icmp = ICMP(type=5, code=1)
icmp.gw = "10.0.2.1"

ip2 = IP(src = "10.0.2.4", dst = "172.217.2.36")
send(ip/icmp/ip2/UDP());
~
~
~
~
~
~
~
~
~
~
~
~
~
"icmpattack.py" 10L, 214C                                    1,1           All
```

This screenshot shows that a successful redirect took place for the packet to go from A to B. I used the skeleton code provided in the textbook and filled in the missing pieces with the information from my vms.

Lab 1
Gunnar Yonker

Questions:

1:



```
No. Time                        Source              Destination         Protocol Length Info
  5 2022-11-07 23:39:48.0697173… 10.0.2.3            10.0.2.4            DHCP     590 DHCP ACK      - Transactio…
  6 2022-11-07 23:39:53.1290517… PcsCompu_a6:5b:3d   PcsCompu_03:5f:03   ARP       60 Who has 10.0.2.3? Tell 10.…
  7 2022-11-07 23:39:53.1290613… PcsCompu_03:5f:03   PcsCompu_a6:5b:3d   ARP       60 10.0.2.3 is at 08:00:27:03…
  8 2022-11-07 23:41:18.5830694… 10.0.2.15           10.0.2.3            DHCP     342 DHCP Request  - Transactio…
  9 2022-11-07 23:41:18.5883232… 10.0.2.3            10.0.2.15           DHCP     590 DHCP ACK      - Transactio…
 10 2022-11-07 23:41:23.8206653… PcsCompu_40:bf:a7   PcsCompu_03:5f:03   ARP       42 Who has 10.0.2.3? Tell 10.…
 11 2022-11-07 23:41:23.8207727… PcsCompu_03:5f:03   PcsCompu_40:bf:a7   ARP       60 10.0.2.3 is at 08:00:27:03…
 12 2022-11-07 23:41:29.4009578… PcsCompu_40:bf:a7   Broadcast           ARP       42 Who has 10.0.2.4? Tell 10.…
 13 2022-11-07 23:41:29.4011767… PcsCompu_a6:5b:3d   PcsCompu_40:bf:a7   ARP       60 10.0.2.4 is at 08:00:27:a6…
 14 2022-11-07 23:41:29.4024671… 10.0.2.15           10.0.2.4            ICMP      70 Redirect            (Redi…
```

```
▸ Frame 14: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_40:bf:a7 (08:00:27:40:bf:a7), Dst: PcsCompu_a6:5b:3d (08:00:27:a6:5b:3d)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.4
▾ Internet Control Message Protocol
    Type: 5 (Redirect)
    Code: 1 (Redirect for host)
    Checksum: 0xaa14 [correct]
    [Checksum Status: Good]
    Gateway address: 10.0.2.5
  ▸ Internet Protocol Version 4, Src: 10.0.2.4, Dst: 172.217.2.36
  ▸ User Datagram Protocol, Src Port: 53, Dst Port: 53
```

```
0000  08 00 27 a6 5b 3d 08 00  27 40 bf a7 08 00 45 00   ..'.[=.. '@....E.
0010  00 38 00 01 00 00 40 01  62 b2 0a 00 02 0f 0a 00   .8....@. b.......
0020  02 04 05 01 aa 14 0a 00  02 05 45 00 00 1c 00 01   ........ ..E.....
0030  00 00 40 11 bf cf 0a 00  02 04 ac d9 02 24 00 35   ..@..... .....$.5
0040  00 35 00 08 44 73                                  .5..Ds
```

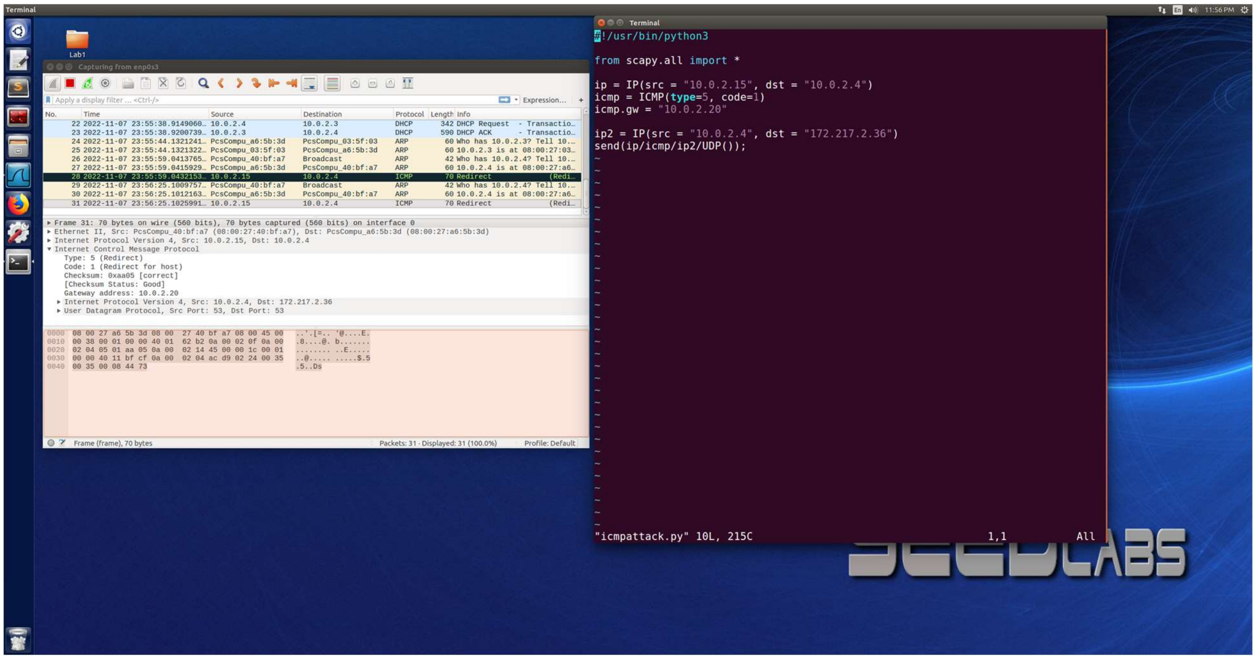
```
Internet Protocol Version 4 (ip), 20 bytes          Packets: 14 · Displayed: 14 (100.0%)          Profile: Default
```

```
#!/usr/bin/python3

from scapy.all import *

ip = IP(src = "10.0.2.15", dst = "10.0.2.4")
icmp = ICMP(type=5, code=1)
icmp.gw = "10.0.2.5"

ip2 = IP(src = "10.0.2.4", dst = "172.217.2.36")
send(ip/icmp/ip2/UDP());
~
"icmpattack.py" 10L, 214C                                    1,1           All
```

For this question we wanted to experiment and see if an ICMP redirect attack could be used on a remote machine. I used 10.0.2.5 which is one of my vms. As seen in the Wireshark screenshot above, a successful redirect did take place where the ICMP redirect attack was directed to a remote machine. This shows that you can use ICMP redirect attack to redirect to a remote machine because the packet was successfully redirected.

Lab 1
Gunnar Yonker

2.



Capturing from enp0s3

```
No.  Time                           Source              Destination         Protocol Length Info
  22 2022-11-07 23:55:38.9149060… 10.0.2.4            10.0.2.3            DHCP     342 DHCP Request  - Transactio…
  23 2022-11-07 23:55:38.9200739… 10.0.2.3            10.0.2.4            DHCP     590 DHCP ACK      - Transactio…
  24 2022-11-07 23:55:44.1321241… PcsCompu_a6:5b:3d   PcsCompu_03:5f:03   ARP       60 Who has 10.0.2.3? Tell 10.…
  25 2022-11-07 23:55:44.1321322… PcsCompu_03:5f:03   PcsCompu_a6:5b:3d   ARP       60 10.0.2.3 is at 08:00:27:03…
  26 2022-11-07 23:55:59.0413765… PcsCompu_40:bf:a7   Broadcast           ARP       42 Who has 10.0.2.4? Tell 10.…
  27 2022-11-07 23:55:59.0415929… PcsCompu_a6:5b:3d   PcsCompu_40:bf:a7   ARP       60 10.0.2.4 is at 08:00:27:a6…
  28 2022-11-07 23:55:59.0432153… 10.0.2.15           10.0.2.4            ICMP      70 Redirect            (Redi…
  29 2022-11-07 23:56:25.1009757… PcsCompu_40:bf:a7   Broadcast           ARP       42 Who has 10.0.2.4? Tell 10.…
  30 2022-11-07 23:56:25.1012163… PcsCompu_a6:5b:3d   PcsCompu_40:bf:a7   ARP       60 10.0.2.4 is at 08:00:27:a6…
  31 2022-11-07 23:56:25.1025991… 10.0.2.15           10.0.2.4            ICMP      70 Redirect            (Redi…
```

```
▸ Frame 31: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_40:bf:a7 (08:00:27:40:bf:a7), Dst: PcsCompu_a6:5b:3d (08:00:27:a6:5b:3d)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.4
▾ Internet Control Message Protocol
    Type: 5 (Redirect)
    Code: 1 (Redirect for host)
    Checksum: 0xaa05 [correct]
    [Checksum Status: Good]
    Gateway address: 10.0.2.20
  ▸ Internet Protocol Version 4, Src: 10.0.2.4, Dst: 172.217.2.36
  ▸ User Datagram Protocol, Src Port: 53, Dst Port: 53
```

```
0000  08 00 27 a6 5b 3d 08 00  27 40 bf a7 08 00 45 00   ..'.[=.. '@....E.
0010  00 38 00 01 00 00 40 01  62 b2 0a 00 02 0f 0a 00   .8....@. b.......
0020  02 04 05 01 aa 05 0a 00  02 14 45 00 00 1c 00 01   ........ ..E.....
0030  00 00 40 11 bf cf 0a 00  02 04 ac d9 02 24 00 35   ..@..... .....$.5
0040  00 35 00 08 44 73                                  .5..Ds
```

Frame (frame), 70 bytes   Packets: 31 · Displayed: 31 (100.0%)   Profile: Default

```
#!/usr/bin/python3

from scapy.all import *

ip = IP(src = "10.0.2.15", dst = "10.0.2.4")
icmp = ICMP(type=5, code=1)
icmp.gw = "10.0.2.20"

ip2 = IP(src = "10.0.2.4", dst = "172.217.2.36")
send(ip/icmp/ip2/UDP());
~
~
~
"icmpattack.py" 10L, 215C                              1,1           All
```

For this question we wanted to experiment and see if an ICMP redirect attack could be redirected to a non-existing machine that is either offline or non-existing. When the packet was redirected using a machine that didn’t exist, the ICMP packed was successfully redirected, but no other packets were received. This means that you would be able to carry out an ICMP redirect attack on a non-existing machine that is on the same network, but the packet will not be received by that machine.