Lab Report - SEED Labs – Packet Sniffing and Spoofing Lab

Name: Kostia Kazakov

ID: 321827834

Task 1.1: Sniffing Packets: Task 1.1A:

Capturing a ping with scapy:

```
Terminal
[11/25/18]seed@VM:~/Desktop$ sudo python sniff.py
###[ Ethernet ]###
  dst       = 52:54:00:12:35:02
  src       = 08:00:27:c6:20:f5
  type      = 0x800
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 39570
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x83f8
     src       = 10.0.2.15
     dst       = 8.8.8.8
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x1740
        id        = 0x1042
        seq       = 0x1
###[ Raw ]###
           load      = '\x0c\xef\xfa[\xd4.\n\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14
\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
```

Without root:

```
Terminal
     proto     = icmp
     chksum    = 0x159b
     src       = 8.8.8.8
     dst       = 10.0.2.15
     \options   \
###[ ICMP ]###
        type      = echo-reply
        code      = 0
        chksum    = 0x1f40
        id        = 0x1042
        seq       = 0x1
###[ Raw ]###
           load      = '\x0c\xef\xfa[\xd4.\n\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14
\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

^Z
[7]+  Stopped                 sudo python sniff.py
[11/25/18]seed@VM:~/Desktop$ python sniff.py
Traceback (most recent call last):
  File "sniff.py", line 7, in <module>
    pkt = sniff(filter='icmp',prn=print_pkt)
  File "/home/seed/.local/lib/python2.7/site-packages/scapy/sendrecv.py", line 731, in sniff
    *arg, **karg)] = iface
  File "/home/seed/.local/lib/python2.7/site-packages/scapy/arch/linux.py", line 567, in __ini
t__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type))
  File "/usr/lib/python2.7/socket.py", line 191, in __init__
    _sock = _realsocket(family, type, proto)
socket.error: [Errno 1] Operation not permitted
[11/25/18]seed@VM:~/Desktop$
```

As the screenshot below we can see clearly that if we run the sniff.py without sudo (root privileges) the problem will not execute in our terminal, because we don't have permission to our network adapter(for creating a raw socket).

Task 1.1B:

Capture only the ICMP packet: we actually did it in Task 1.1A, because the code provided already was with filter "icmp".

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

```
Terminal
[11/26/18]seed@VM:~/Desktop$ sudo python sniff.py
###[ Ethernet ]###
  dst       = 52:54:00:12:35:02
  src       = 08:00:27:c6:20:f5
  type      = 0x800
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 60
     id        = 33751
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x9ab6
     src       = 10.0.2.15
     dst       = 8.8.8.8
     \options   \
###[ TCP ]###
        sport     = 40920
        dport     = telnet
        seq       = 1795958954
        ack       = 0
        dataofs   = 10
        reserved  = 0
        flags     = S
        window    = 29200
        chksum    = 0x1c4d
        urgptr    = 0
```

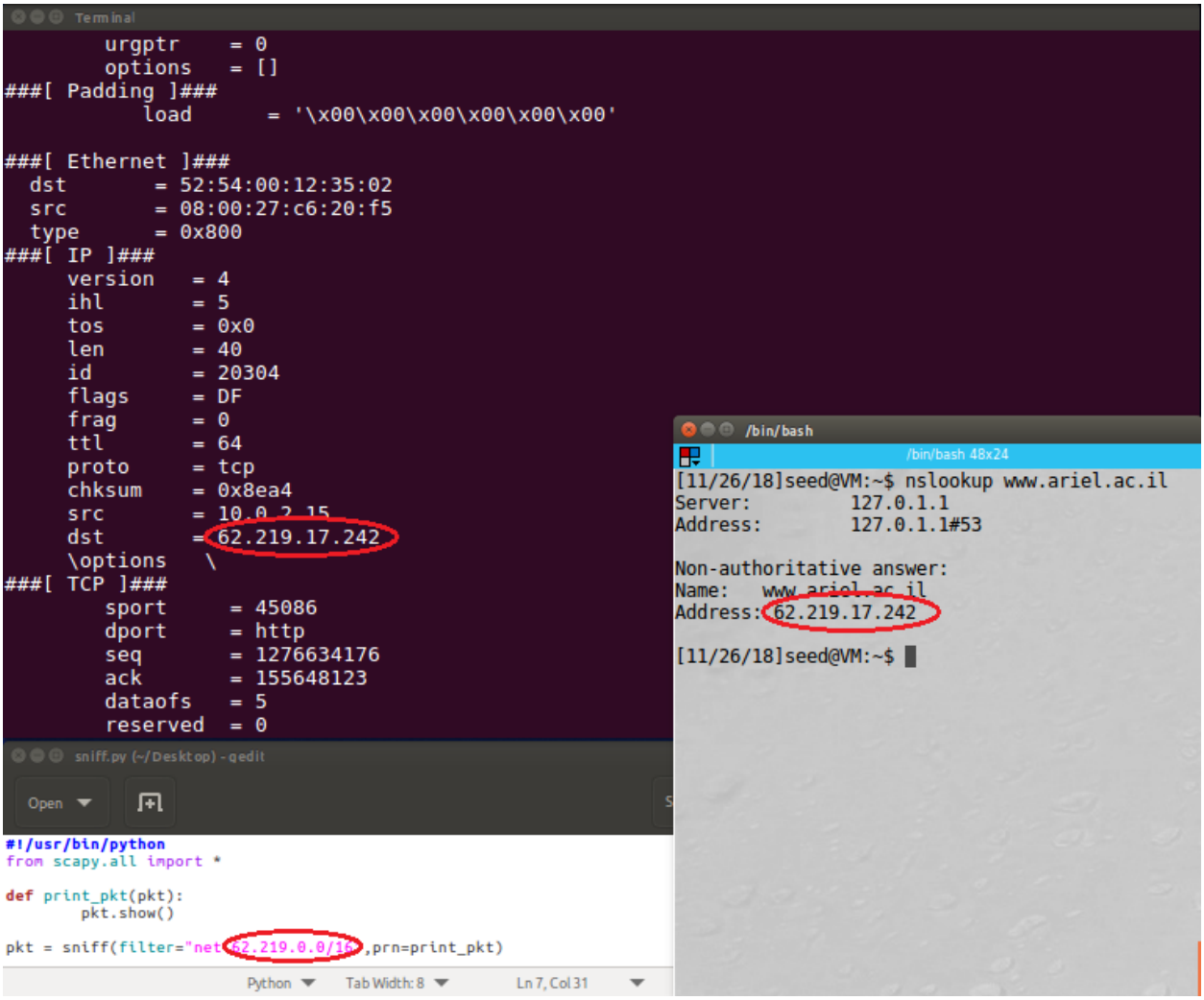
```
sniff.py (~/Desktop) - gedit
Open ▾   Save
#!/usr/bin/python
from scapy.all import *

def print_pkt(pkt):
        pkt.show()

pkt = sniff(filter="host 10.0.2.15 and tcp dst port 23",prn=print_pkt)
Python ▾   Tab Width: 8 ▾   Ln 5, Col 19 ▾   INS
```

(telnet is port 23)

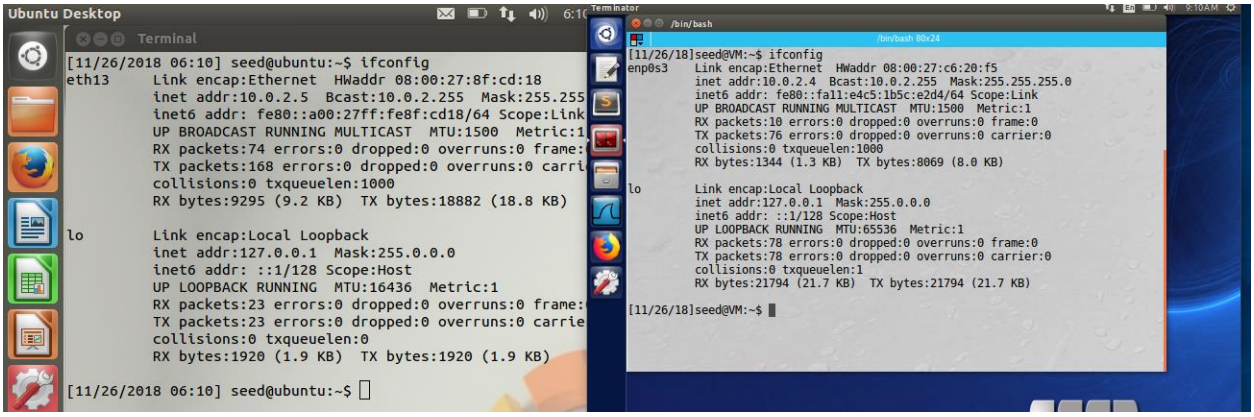
Capture packets comes from or to go to a particular subnet:



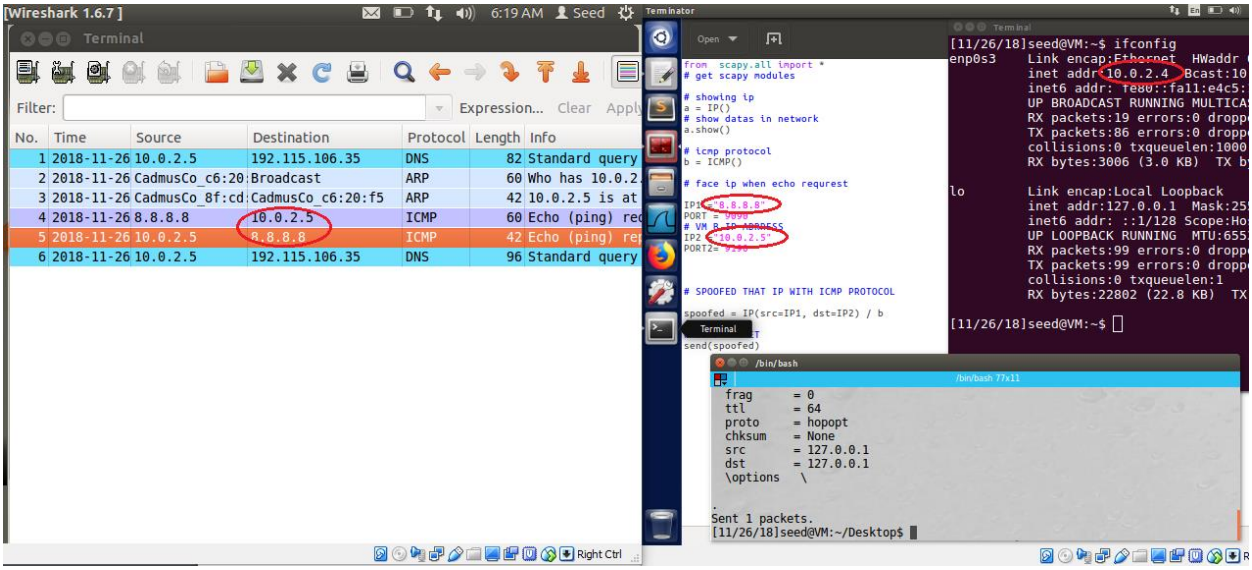
Task 1.2: **Spoofing ICMP Packets:**

In this task, we first need to configure two VM's with different IP's via Nat Network.

The first machines IP is 10.0.2.5 and the second 10.0.2.4:

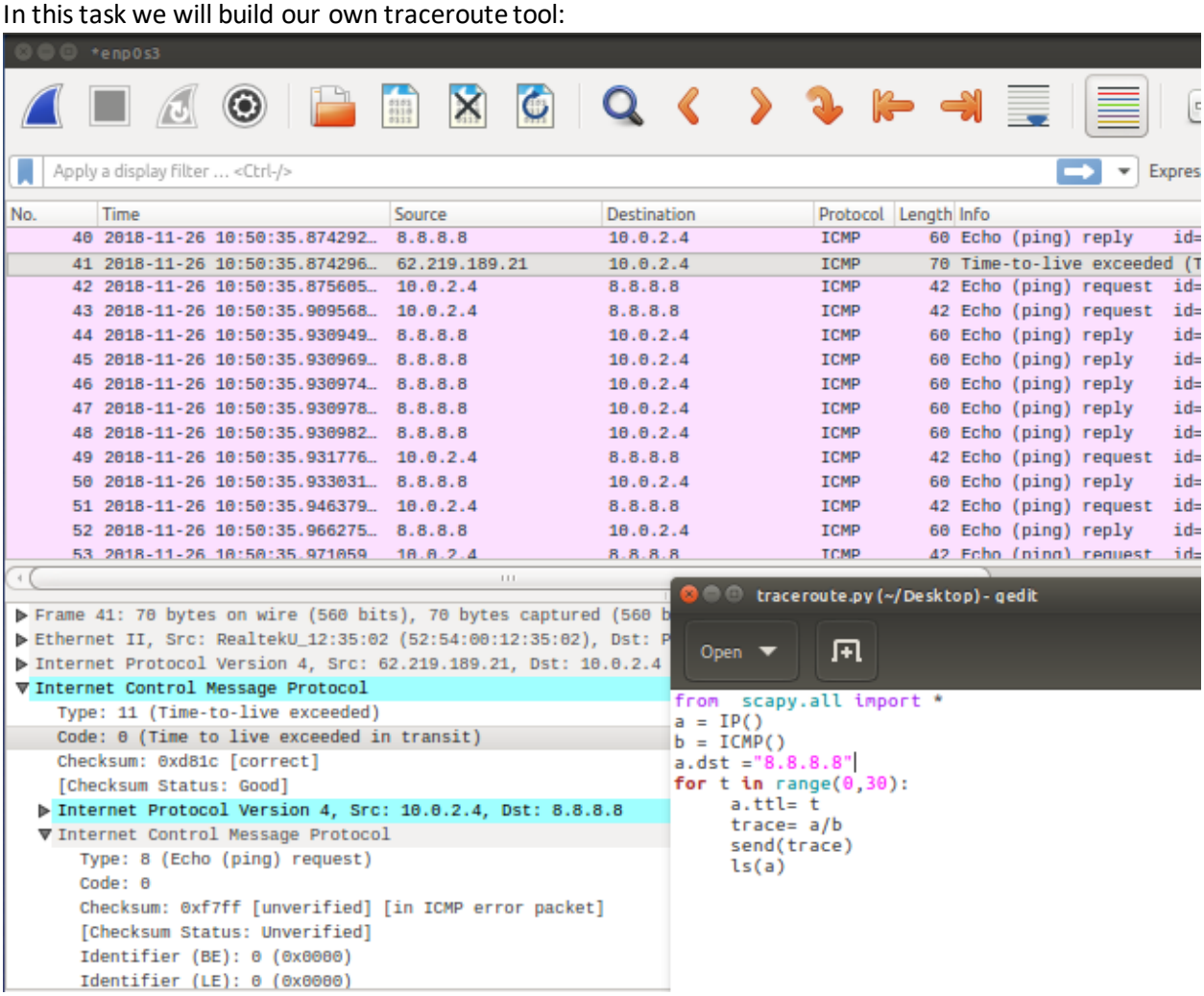


After that, we need to send from machine “4” to machine “5” spoofed ping, we can do it with scapy by adding b as the payload field of a and modifying the fields of a accordingly:



Task 1.3: **Traceroute:**

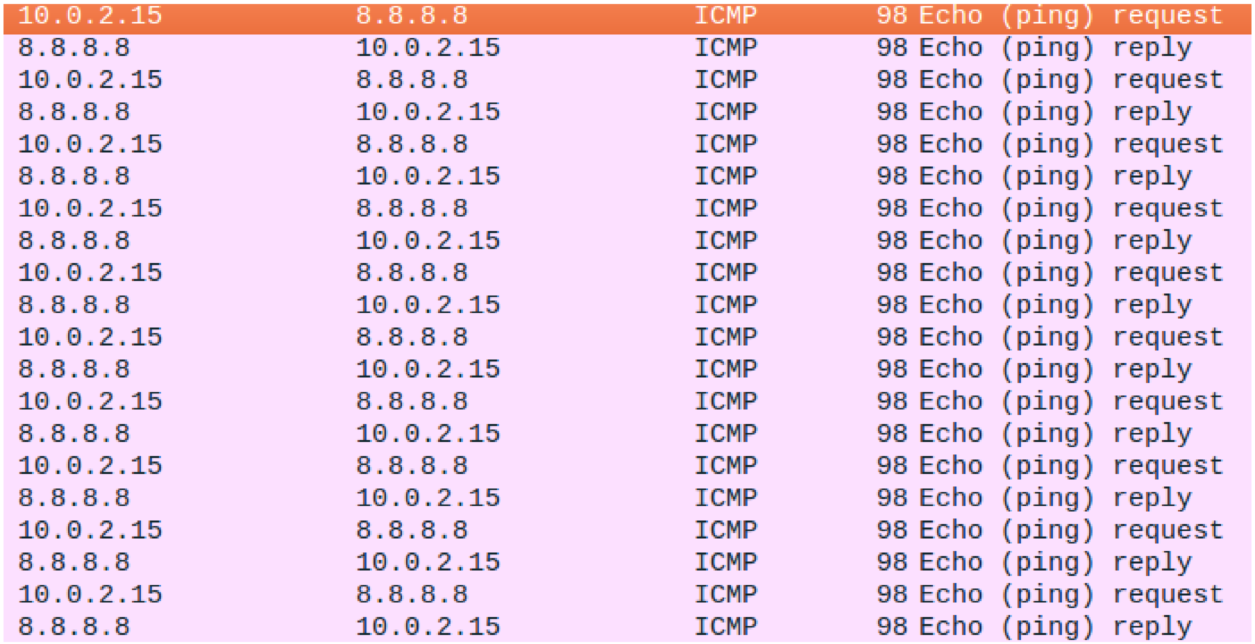
In this task we will build our own traceroute tool:



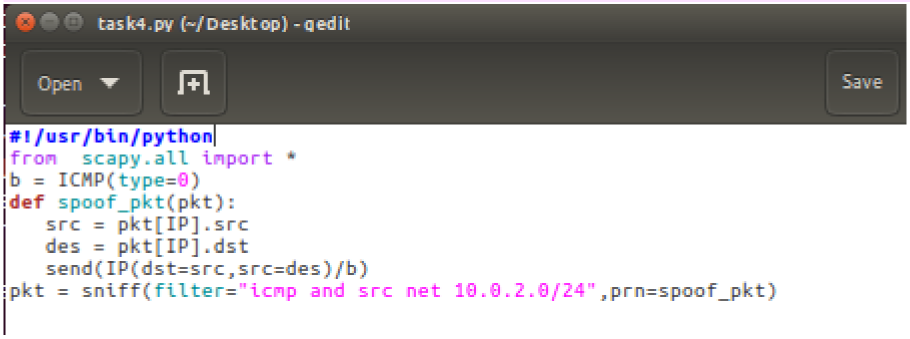
Task 1.4: Sniffing and-then Spoofing:

Combine the sniffing and spoofing techniques to implement the following sniff-and then-spoof program.

- From VM A, you ping an IP X, it should receive an echo-reply.
- From VM B, you run the sniff and spoof which monitors the LAN through packet sniffing.
    - Regardless of what the target IP address is, should send out an echo reply indicating that X is alive.



```
10.0.2.15       8.8.8.8         ICMP      98 Echo (ping) request
8.8.8.8         10.0.2.15       ICMP      98 Echo (ping) reply
10.0.2.15       8.8.8.8         ICMP      98 Echo (ping) request
8.8.8.8         10.0.2.15       ICMP      98 Echo (ping) reply
10.0.2.15       8.8.8.8         ICMP      98 Echo (ping) request
8.8.8.8         10.0.2.15       ICMP      98 Echo (ping) reply
10.0.2.15       8.8.8.8         ICMP      98 Echo (ping) request
8.8.8.8         10.0.2.15       ICMP      98 Echo (ping) reply
10.0.2.15       8.8.8.8         ICMP      98 Echo (ping) request
8.8.8.8         10.0.2.15       ICMP      98 Echo (ping) reply
10.0.2.15       8.8.8.8         ICMP      98 Echo (ping) request
8.8.8.8         10.0.2.15       ICMP      98 Echo (ping) reply
10.0.2.15       8.8.8.8         ICMP      98 Echo (ping) request
8.8.8.8         10.0.2.15       ICMP      98 Echo (ping) reply
10.0.2.15       8.8.8.8         ICMP      98 Echo (ping) request
8.8.8.8         10.0.2.15       ICMP      98 Echo (ping) reply
10.0.2.15       8.8.8.8         ICMP      98 Echo (ping) request
8.8.8.8         10.0.2.15       ICMP      98 Echo (ping) reply
10.0.2.15       8.8.8.8         ICMP      98 Echo (ping) request
8.8.8.8         10.0.2.15       ICMP      98 Echo (ping) reply
```



```
#!/usr/bin/python
from  scapy.all import *
b = ICMP(type=0)
def spoof_pkt(pkt):
    src = pkt[IP].src
    des = pkt[IP].dst
    send(IP(dst=src,src=des)/b)
pkt = sniff(filter="icmp and src net 10.0.2.0/24",prn=spoof_pkt)
```