Francis Convery, Stephen Hess
Dr. Linh Ngo
CSC302
14 November 2019

Sniffing and Spoofing

Directory:

opt/anaconda3/bin/python

Set Up

```
cp -R /local/repository/codes .
```

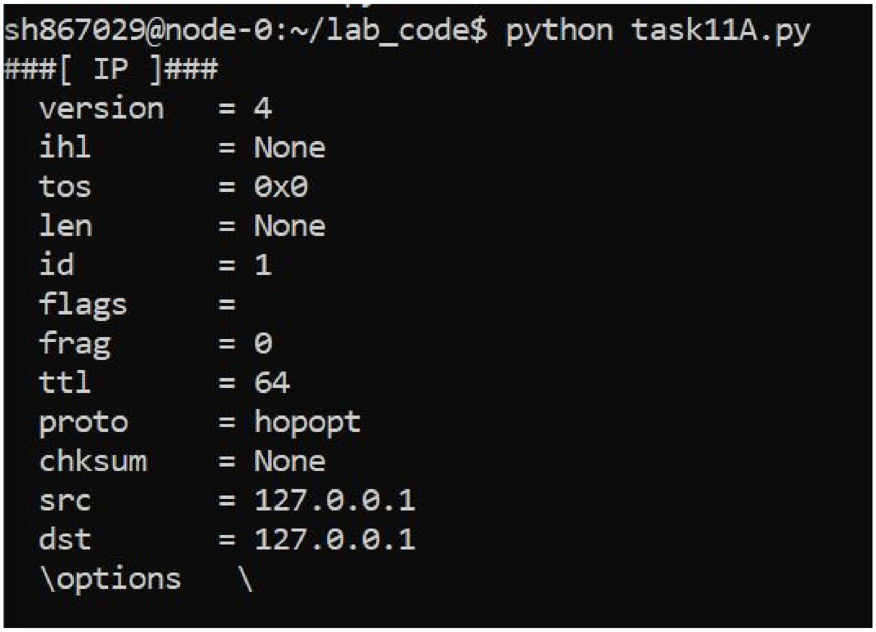
URLS:

ssh -p 22 sh867029@clnodevm014-1.clemson.cloudlab.us

node-1 clnodevm014-2 ready pcvm emulab-ops/UBUNTU16-64-STD ssh -p 22 sh867029@clnodevm014-2.clemson.cloudlab.us

node-2 clnodevm014-3 ready pcvm emulab-ops/UBUNTU16-64-STD ssh -p 22 sh867029@clnodevm014-3.clemson.cloudlab.us

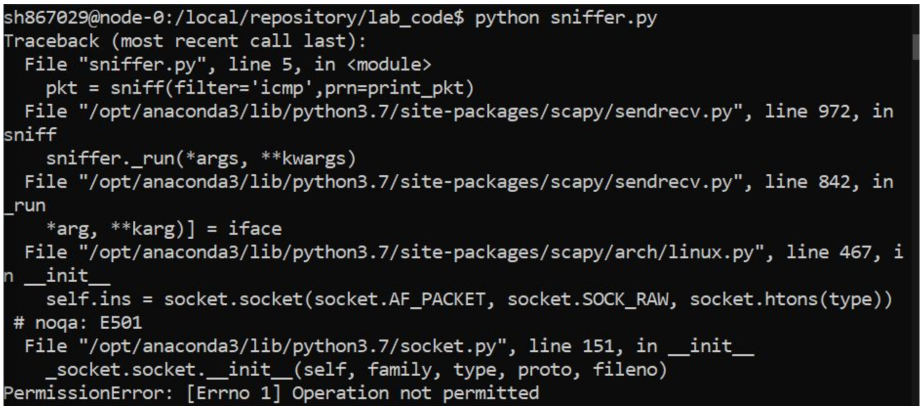
Node-1 IPv6: fe80::34:3cff:fef7:1cdb/64



```
sh867029@node-0:~/lab_code$ python task11A.py
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = hopopt
  chksum    = None
  src       = 127.0.0.1
  dst       = 127.0.0.1
  \options   \
```

Task 1.1A

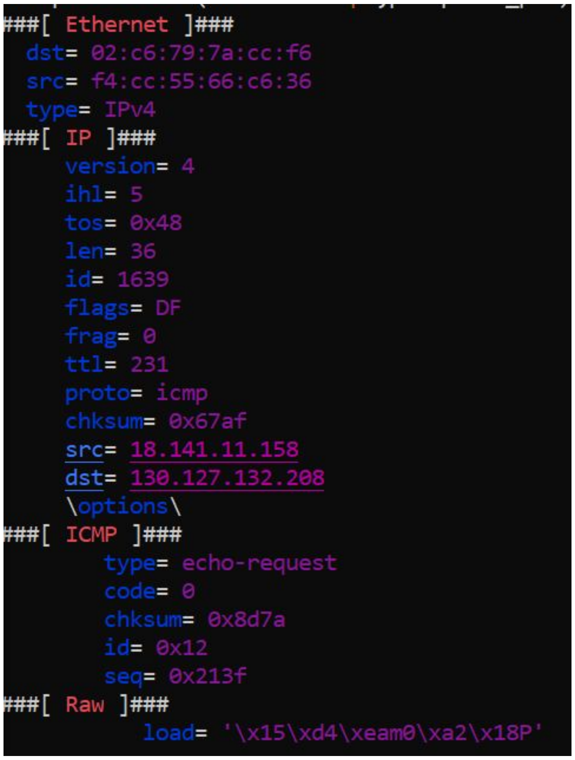
Without Sudo:



```
sh867029@node-0:/local/repository/lab_code$ python sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 5, in <module>
    pkt = sniff(filter='icmp',prn=print_pkt)
  File "/opt/anaconda3/lib/python3.7/site-packages/scapy/sendrecv.py", line 972, in
sniff
    sniffer._run(*args, **kwargs)
  File "/opt/anaconda3/lib/python3.7/site-packages/scapy/sendrecv.py", line 842, in
_run
    *arg, **karg)] = iface
  File "/opt/anaconda3/lib/python3.7/site-packages/scapy/arch/linux.py", line 467, i
n __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type))
 # noqa: E501
  File "/opt/anaconda3/lib/python3.7/socket.py", line 151, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
```

Without using root privileges, sniffer.py does not have permission to capture the packets.

With Sudo:

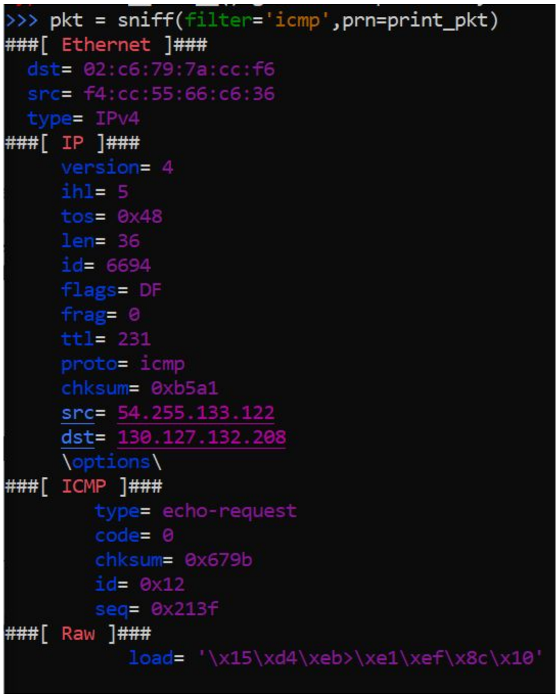


```
###[ Ethernet ]###
  dst= 02:c6:79:7a:cc:f6
  src= f4:cc:55:66:c6:36
  type= IPv4
###[ IP ]###
     version= 4
     ihl= 5
     tos= 0x48
     len= 36
     id= 1639
     flags= DF
     frag= 0
     ttl= 231
     proto= icmp
     chksum= 0x67af
     src= 18.141.11.158
     dst= 130.127.132.208
     \options\
###[ ICMP ]###
        type= echo-request
        code= 0
        chksum= 0x8d7a
        id= 0x12
        seq= 0x213f
###[ Raw ]###
           load= '\x15\xd4\xeam0\xa2\x18P'
```

Once sniffer.py was executed with the sudo command, the program had root privilege and was able to capture packets.

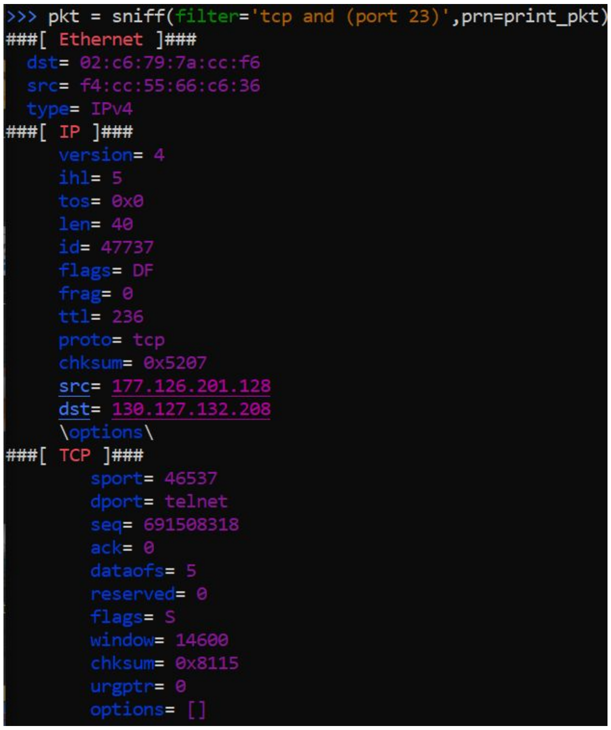
Task 1.1B

ICMP Filter:



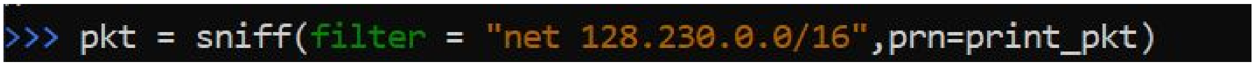
```
>>> pkt = sniff(filter='icmp',prn=print_pkt)
###[ Ethernet ]###
  dst= 02:c6:79:7a:cc:f6
  src= f4:cc:55:66:c6:36
  type= IPv4
###[ IP ]###
     version= 4
     ihl= 5
     tos= 0x48
     len= 36
     id= 6694
     flags= DF
     frag= 0
     ttl= 231
     proto= icmp
     chksum= 0xb5a1
     src= 54.255.133.122
     dst= 130.127.132.208
     \options\
###[ ICMP ]###
        type= echo-request
        code= 0
        chksum= 0x679b
        id= 0x12
        seq= 0x213f
###[ Raw ]###
           load= '\x15\xd4\xeb>\xe1\xef\x8c\x10'
```

TCP Filter:



```
>>> pkt = sniff(filter='tcp and (port 23)',prn=print_pkt)
###[ Ethernet ]###
  dst= 02:c6:79:7a:cc:f6
  src= f4:cc:55:66:c6:36
  type= IPv4
###[ IP ]###
     version= 4
     ihl= 5
     tos= 0x0
     len= 40
     id= 47737
     flags= DF
     frag= 0
     ttl= 236
     proto= tcp
     chksum= 0x5207
     src= 177.126.201.128
     dst= 130.127.132.208
     \options\
###[ TCP ]###
        sport= 46537
        dport= telnet
        seq= 691508318
        ack= 0
        dataofs= 5
        reserved= 0
        flags= S
        window= 14600
        chksum= 0x8115
        urgptr= 0
        options= []
```

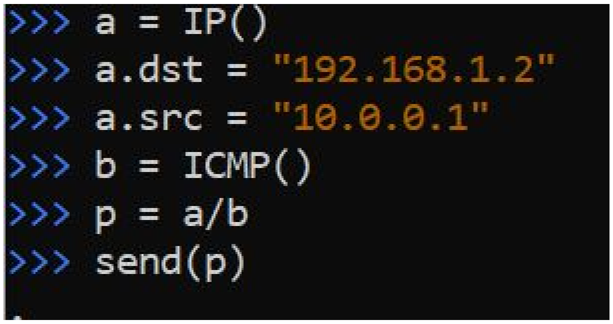
Any Particular Subnet:



```
>>> pkt = sniff(filter = "net 128.230.0.0/16",prn=print_pkt)
```

Task 1.2

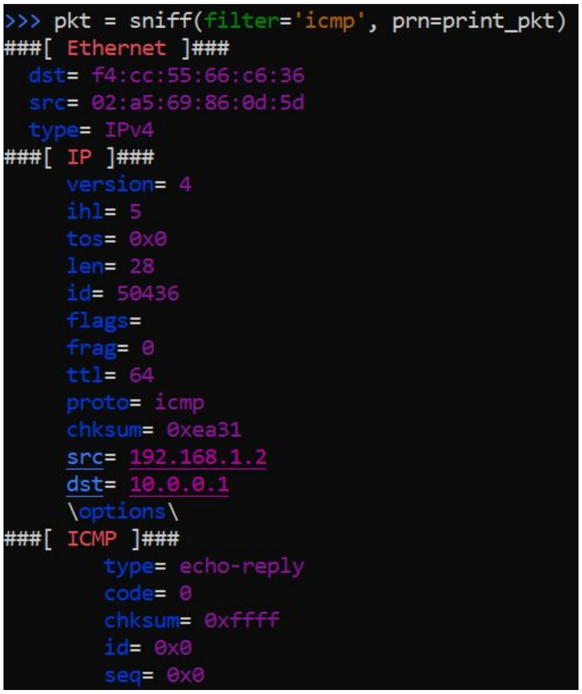
Code of Node-0:



```
>>> a = IP()
>>> a.dst = "192.168.1.2"
>>> a.src = "10.0.0.1"
>>> b = ICMP()
>>> p = a/b
>>> send(p)
```

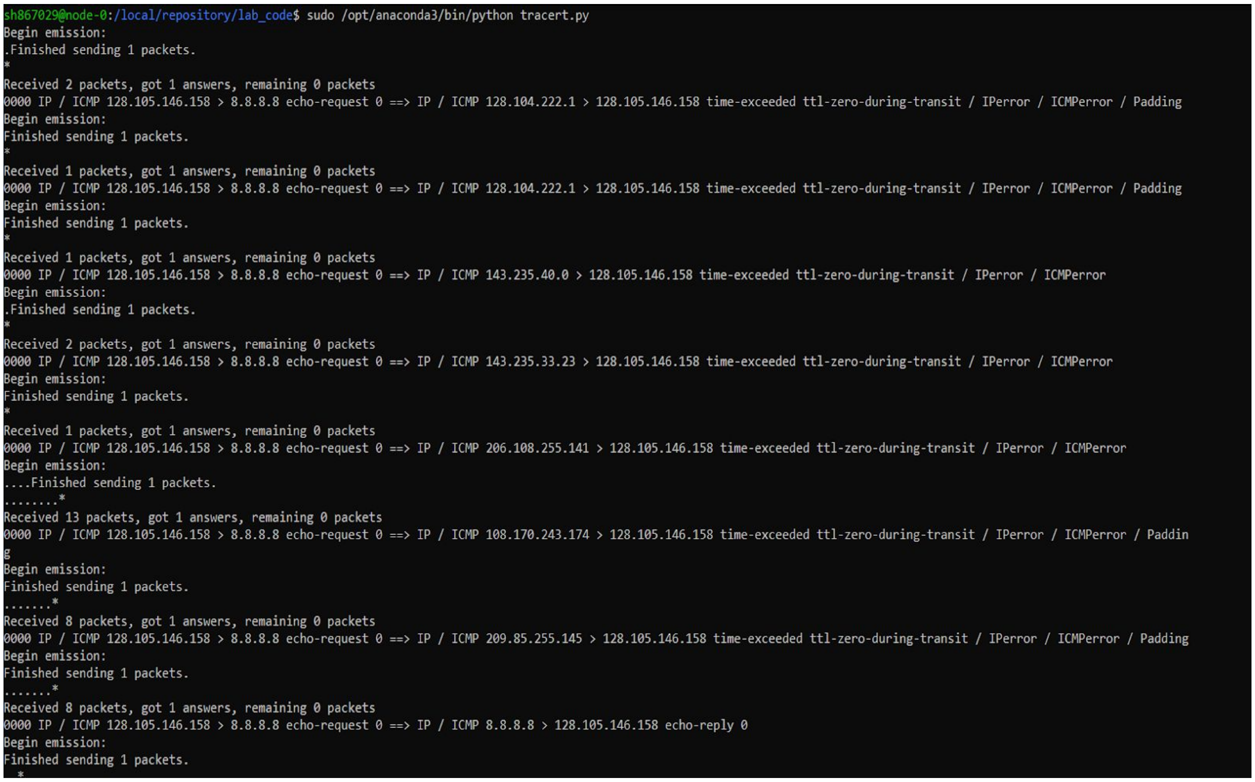
In this screenshot, the source address was spoofed to be “10.0.0.1”. This can be seen in the screenshot below that contains the output of node 1.

Output of Node-1:

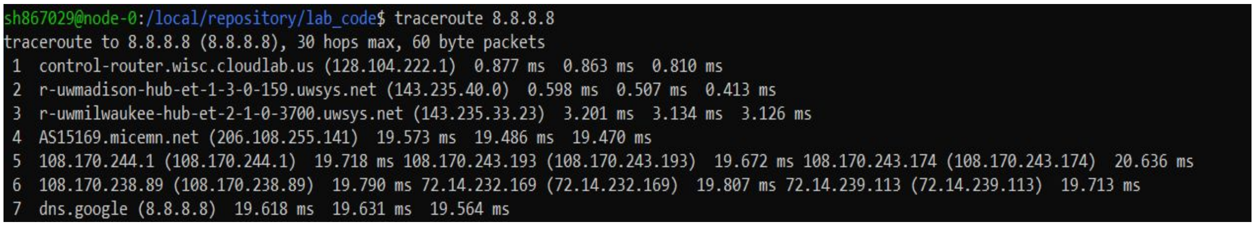


```
>>> pkt = sniff(filter='icmp', prn=print_pkt)
###[ Ethernet ]###
  dst= f4:cc:55:66:c6:36
  src= 02:a5:69:86:0d:5d
  type= IPv4
###[ IP ]###
     version= 4
     ihl= 5
     tos= 0x0
     len= 28
     id= 50436
     flags=
     frag= 0
     ttl= 64
     proto= icmp
     chksum= 0xea31
     src= 192.168.1.2
     dst= 10.0.0.1
     \options\
###[ ICMP ]###
        type= echo-reply
        code= 0
        chksum= 0xffff
        id= 0x0
        seq= 0x0
```

Task 1.3



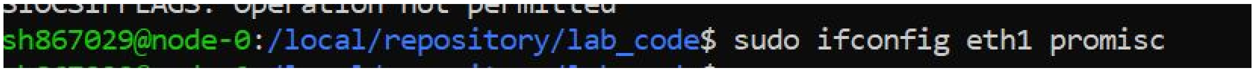
```
sh867029@node-0:/local/repository/lab_code$ sudo /opt/anaconda3/bin/python tracert.py
Begin emission:
.Finished sending 1 packets.
*
Received 2 packets, got 1 answers, remaining 0 packets
0000 IP / ICMP 128.105.146.158 > 8.8.8.8 echo-request 0 ==> IP / ICMP 128.104.222.1 > 128.105.146.158 time-exceeded ttl-zero-during-transit / IPerror / ICMPerror / Padding
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
0000 IP / ICMP 128.105.146.158 > 8.8.8.8 echo-request 0 ==> IP / ICMP 128.104.222.1 > 128.105.146.158 time-exceeded ttl-zero-during-transit / IPerror / ICMPerror / Padding
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
0000 IP / ICMP 128.105.146.158 > 8.8.8.8 echo-request 0 ==> IP / ICMP 143.235.40.0 > 128.105.146.158 time-exceeded ttl-zero-during-transit / IPerror / ICMPerror
Begin emission:
.Finished sending 1 packets.
*
Received 2 packets, got 1 answers, remaining 0 packets
0000 IP / ICMP 128.105.146.158 > 8.8.8.8 echo-request 0 ==> IP / ICMP 143.235.33.23 > 128.105.146.158 time-exceeded ttl-zero-during-transit / IPerror / ICMPerror
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
0000 IP / ICMP 128.105.146.158 > 8.8.8.8 echo-request 0 ==> IP / ICMP 206.108.255.141 > 128.105.146.158 time-exceeded ttl-zero-during-transit / IPerror / ICMPerror
Begin emission:
....Finished sending 1 packets.
........*
Received 13 packets, got 1 answers, remaining 0 packets
0000 IP / ICMP 128.105.146.158 > 8.8.8.8 echo-request 0 ==> IP / ICMP 108.170.243.174 > 128.105.146.158 time-exceeded ttl-zero-during-transit / IPerror / ICMPerror / Paddin
g
Begin emission:
Finished sending 1 packets.
.......*
Received 8 packets, got 1 answers, remaining 0 packets
0000 IP / ICMP 128.105.146.158 > 8.8.8.8 echo-request 0 ==> IP / ICMP 209.85.255.145 > 128.105.146.158 time-exceeded ttl-zero-during-transit / IPerror / ICMPerror / Padding
Begin emission:
Finished sending 1 packets.
.......*
Received 8 packets, got 1 answers, remaining 0 packets
0000 IP / ICMP 128.105.146.158 > 8.8.8.8 echo-request 0 ==> IP / ICMP 8.8.8.8 > 128.105.146.158 echo-reply 0
Begin emission:
Finished sending 1 packets.
  *
```



```
sh867029@node-0:/local/repository/lab_code$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  control-router.wisc.cloudlab.us (128.104.222.1)  0.877 ms  0.863 ms  0.810 ms
 2  r-uwmadison-hub-et-1-3-0-159.uwsys.net (143.235.40.0)  0.598 ms  0.507 ms  0.413 ms
 3  r-uwmilwaukee-hub-et-2-1-0-3700.uwsys.net (143.235.33.23)  3.201 ms  3.134 ms  3.126 ms
 4  AS15169.micemn.net (206.108.255.141)  19.573 ms  19.486 ms  19.470 ms
 5  108.170.244.1 (108.170.244.1)  19.718 ms 108.170.243.193 (108.170.243.193)  19.672 ms 108.170.243.174 (108.170.243.174)  20.636 ms
 6  108.170.238.89 (108.170.238.89)  19.790 ms 72.14.232.169 (72.14.232.169)  19.807 ms 72.14.239.113 (72.14.239.113)  19.713 ms
 7  dns.google (8.8.8.8)  19.618 ms  19.631 ms  19.564 ms
```

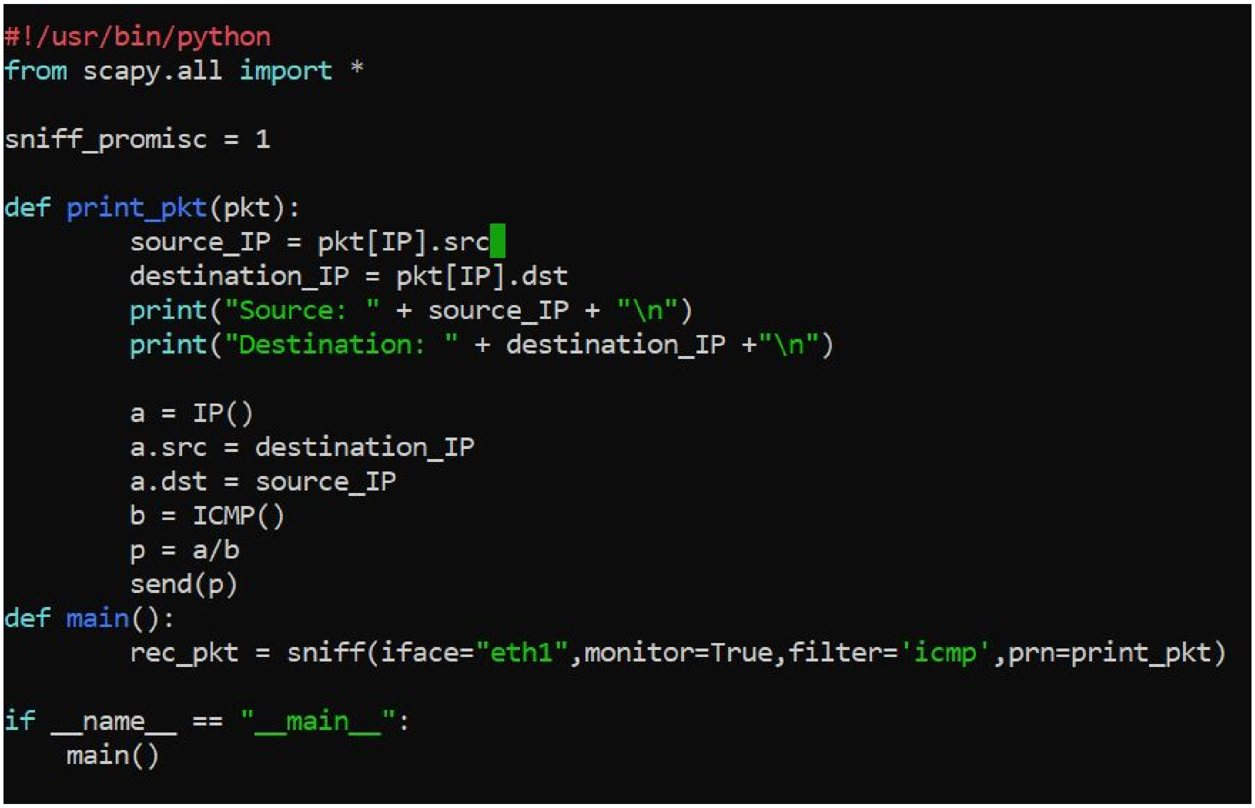
It takes 7 tries before we get an echo-reply. Therefore, we know that there must be 7 routers in between us and 8.8.8.8. This can be verified by using the official traceroute program.

Task 1.4



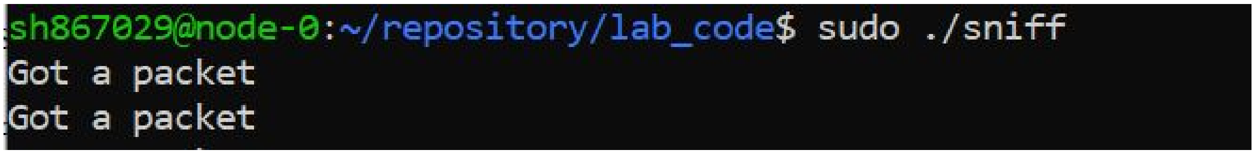
```
SIOCSIFFLAGS: Operation not permitted
sh867029@node-0:/local/repository/lab_code$ sudo ifconfig eth1 promisc
```

This command enables promiscuous mode on the interface.

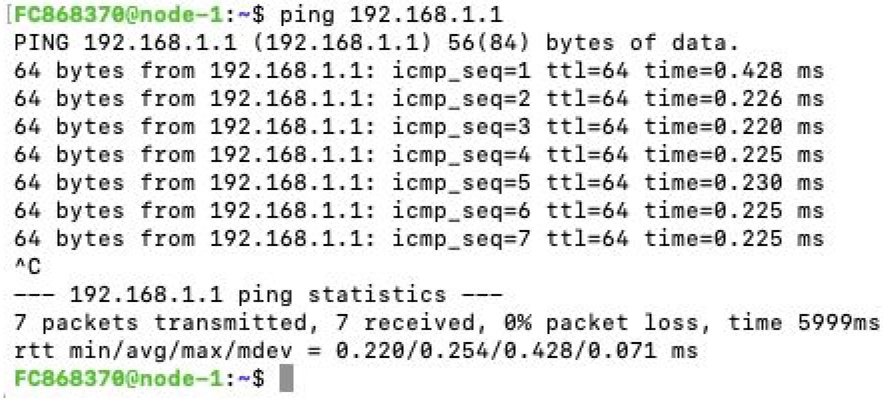


```
#!/usr/bin/python
from scapy.all import *

sniff_promisc = 1

def print_pkt(pkt):
        source_IP = pkt[IP].src
        destination_IP = pkt[IP].dst
        print("Source: " + source_IP + "\n")
        print("Destination: " + destination_IP +"\n")

        a = IP()
        a.src = destination_IP
        a.dst = source_IP
        b = ICMP()
        p = a/b
        send(p)
def main():
        rec_pkt = sniff(iface="eth1",monitor=True,filter='icmp',prn=print_pkt)

if __name__ == "__main__":
    main()
```

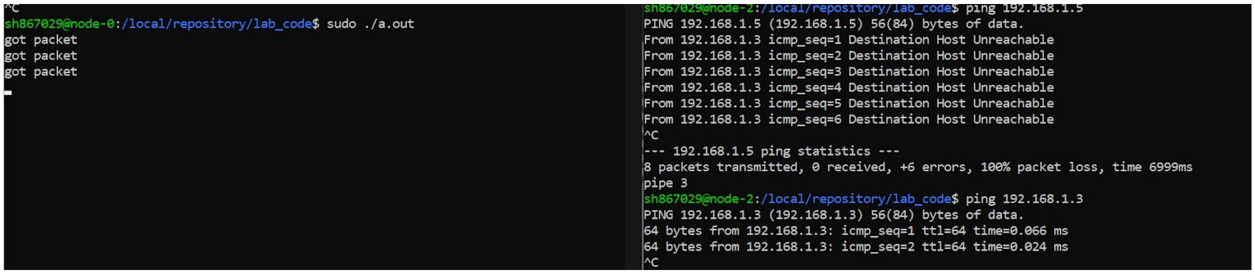
Task 2.1



```
sh867029@node-0:~/repository/lab_code$ sudo ./sniff
Got a packet
Got a packet
```



```
FC868370@node-1:~$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.428 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.226 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.220 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=0.225 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=0.230 ms
64 bytes from 192.168.1.1: icmp_seq=6 ttl=64 time=0.225 ms
64 bytes from 192.168.1.1: icmp_seq=7 ttl=64 time=0.225 ms
^C
--- 192.168.1.1 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 5999ms
rtt min/avg/max/mdev = 0.220/0.254/0.428/0.071 ms
FC868370@node-1:~$
```



```
^C
sh867029@node-0:/local/repository/lab_code$ sudo ./a.out
got packet
got packet
got packet
▬
```

```
sh867029@node-2:/local/repository/lab_code$ ping 192.168.1.5
PING 192.168.1.5 (192.168.1.5) 56(84) bytes of data.
From 192.168.1.3 icmp_seq=1 Destination Host Unreachable
From 192.168.1.3 icmp_seq=2 Destination Host Unreachable
From 192.168.1.3 icmp_seq=3 Destination Host Unreachable
From 192.168.1.3 icmp_seq=4 Destination Host Unreachable
From 192.168.1.3 icmp_seq=5 Destination Host Unreachable
From 192.168.1.3 icmp_seq=6 Destination Host Unreachable
^C
--- 192.168.1.5 ping statistics ---
8 packets transmitted, 0 received, +6 errors, 100% packet loss, time 6999ms
pipe 3
sh867029@node-2:/local/repository/lab_code$ ping 192.168.1.3
PING 192.168.1.3 (192.168.1.3) 56(84) bytes of data.
64 bytes from 192.168.1.3: icmp_seq=1 ttl=64 time=0.066 ms
64 bytes from 192.168.1.3: icmp_seq=2 ttl=64 time=0.024 ms
^C
```

When we ping an address that does not exist(192.168.1.5), we are able to capture the ICMP packet. When we ping a real machine (192.168.1.3) we are unable to capture the packet.

Task 2.1A

Question 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 or book.*

1. pcap_open_live: Opens a live pcap session and sniffs the capture device
2. pcap_compile: Compiles the filter expression stored in a regular string-in order to set the filter
3. pcap_setfilter: Sets the filter that was compiled by the previous call.
4. pcap_loop: Captures packets
5. pcap_close: Closes the sniffing session which is handle in this case

Question 2: *Why do you need the root privilege to run a sniffer program? Where does the program fail if it is executed without the root privilege?*

We needed to utilize the root privilege to run a sniffer program because we wanted to get the interface. Otherwise, a segmentation fault would occur.

Question 3: *Please turn on and turn off the promiscuous mode in your sniffer program. Can you demonstrate the difference when this mode is on and off? Please describe how you can demonstrate this.*

Promiscuous mode allows a malicious user to sniff and pass any packets from a network controller. This even includes traffic that is not addressed to their device.

Task 2.1B

char filter_exp[] = "icmp and (src host 10.219.219.126 and dst host 8.8.8.8) or (src host 8.8.8.8 and dst host 10.219.219.126)";   // get icmp packets between two specific hosts

Task 2.1C

Task 2.2

Task 2.2A

Task 2.2B

Question 4: *Can you set the IP packet length field to an arbitrary value, regardless of how big the actual packet is?*

Yes, a user can set the IP packet length to an arbitrary value. The extra space will be represented as 0’s.

Question 5: *Using the raw socket programming, do you have to calculate the checksum for the IP header?*

The checksum is calculated as soon as the packet reaches the router. The computed checksum is compared to the checksum field of the header. Whether or not the values match determine if the router will drop the packet. The TTL is decremented by one and the packet is passed on.

Question 6: *Why do you need the root privilege to run the programs that use raw sockets? Where does the program fail if executed without the root privilege?*

Raw sockets would allow malicious users to interfere with inbound traffic by spoofing custom packets.

Task 2.2C

Task 2.3