Nini Ola

Cyber Security

January 30 2024

Lab 1: Sniff/ Spoof

1. Task 1.1

- Task 1.1a

- Code snippet

```
#!/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-08b5583a0b6c', filter='icmp', prn=print_pkt)
#pkt = sniff(iface='br-08b5583a0b6c', filter='tcp && src host 10.9.0.6 && dst port 23', prn=print_pkt)
pkt = sniff(iface='br-08b5583a0b6c', filter='net 128.230.0.0/16', prn=print_pkt)
~
~
~
```

- Output1: for running with root privileges/ showing only icmp files

```
seed@instance-1:/home/niniola142002/Labsetup$ sudo docker exec -it hostA-10.9.0.5 /bin/bash
root@8e14c407221e:/# ping hostB-10.9.0.6
PING hostB-10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=1 ttl=64 time=0.396 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=2 ttl=64 time=0.126 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=3 ttl=64 time=0.139 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=4 ttl=64 time=0.112 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=5 ttl=64 time=0.129 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=6 ttl=64 time=0.141 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=7 ttl=64 time=0.134 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=8 ttl=64 time=0.114 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=9 ttl=64 time=0.086 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=10 ttl=64 time=0.122 ms
^C
--- hostB-10.9.0.6 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9146ms
rtt min/avg/max/mdev = 0.086/0.149/0.396/0.083 ms
root@8e14c407221e:/#
```

- Output 2: running without root privileges

```
^Croot@instance-1:/volumes# su seed
seed@instance-1:/volumes$ ./tas11.1.py
Traceback (most recent call last):
  File "./tas11.1.py", line 10, in <module>
    pkt = sniff(iface='br-08b5583a0b6c', 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, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type))  # noqa:
  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@instance-1:/volumes$ 3434
```

- Observation: when you run with root privileges it works and captures packet but when you run without root privileges it shows an error: unable to perform/ operation not permitted

- Explanation: capturing network packets often requires elevated privileges due to security concerns. Root privileges provide the access to the network interface for packet sniffing.

- Task 1.1b

- Code snippet: same as 1.1A with some code commented out
- Output1: only icmp files

```
seed@instance-1:/home/niniola142002/Labsetup$ sudo docker exec -it hostA-10.9.0.5 /bin/bash
root@8e14c407221e:/# ping hostB-10.9.0.6
PING hostB-10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=1 ttl=64 time=0.396 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=2 ttl=64 time=0.126 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=3 ttl=64 time=0.139 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=4 ttl=64 time=0.112 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=5 ttl=64 time=0.129 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=6 ttl=64 time=0.141 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=7 ttl=64 time=0.134 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=8 ttl=64 time=0.114 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=9 ttl=64 time=0.086 ms
64 bytes from hostB-10.9.0.6.net-10.9.0.0 (10.9.0.6): icmp_seq=10 ttl=64 time=0.122 ms
^C
--- hostB-10.9.0.6 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9146ms
rtt min/avg/max/mdev = 0.086/0.149/0.396/0.083 ms
root@8e14c407221e:/#
```

- Output2: only tcp from any ip and port 23

```
###[ 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       = 52
     id        = 1775
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x1fa9
     src       = 10.9.0.6
     dst       = 10.9.0.5
     \options   \
###[ TCP ]###
        sport     = 33324
        dport     = telnet
        seq       = 2834194813
        ack       = 1961650000
        dataofs   = 8
        reserved  = 0
        flags     = A
        window    = 501
        chksum    = 0x1443
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (2731132725, 261069770))]

###[ Ethernet ]###
  dst       = 02:42:0a:09:00:06
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
```

- Output 3:Capture packets comes from or to go to a particular subnet. You can pick any subnet, such as 128.230.0.0/16

```
     \options   \
 ###[ ICMP ]###
        type      = dest-unreach
        code      = host-unreachable
        chksum    = 0xfcfe
        reserved  = 0
        length    = 0
        nexthopmtu= 0
 ###[ IP in ICMP ]###
           version   = 4
           ihl       = 5
           tos       = 0x0
           len       = 84
           id        = 33342
           flags     = DF
           frag      = 0
           ttl       = 53
           proto     = icmp
           chksum    = 0x386b
           src       = 10.9.0.6
           dst       = 128.230.0.11
           \options   \
 ###[ ICMP in ICMP ]###
              type      = echo-request
              code      = 0
              chksum    = 0xa041
              id        = 0x3
              seq       = 0x6
 ###[ Raw ]###
                 load      = '\xaag\xb9e\x00\x00\x00\x00(\x15\r\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16
\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'
```

- Observation: the filter only shows files that are applicable to it
- Explanation: the packets captured are based on the specified filter. The filter only captures files that apply to it

  Code explanation: from scapy.all import * imports all functionalities from the Scapy library.
- The print_pkt function is defined to display the details of each captured packet using the show method provided by Scapy.
- The sniff function from Scapy is used to capture packets on the specified network interface (iface).
- It includes a packet filter (filter) to selectively capture packets based on certain criteria. The prn parameter specifies the function to be called for each captured packet, which, in this case, is print_pkt.

2. Task 1.2

a. Code snippet

```
#!/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-08b5583a0b6c' )
```

b. Output 1 : showing packet sent

```
Sent 1 packets.
root@instance-1:/volumes# ./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                     = '10.128.0.2'    (None)
dst        : DestIPField                       = '1.2.3.4'       (None)
options    : PacketListField                   = []              ([])
.
Sent 1 packets.
```

c. Output2: showing packet received on tshark (couldn't get wireshark interface to work on gcp vm)

```
seed@instance-1:/home/niniola142002$ sudo tshark -i br-08b5583a0b6c
Running as user "root" and group "root". This could be dangerous.
Capturing on 'br-08b5583a0b6c'
^C0 packets captured
seed@instance-1:/home/niniola142002$ sudo tshark -i br-08b5583a0b6c -f "host  1.2.3.4"
Running as user "root" and group "root". This could be dangerous.
Capturing on 'br-08b5583a0b6c'
^C0 packets captured
seed@instance-1:/home/niniola142002$ sudo tshark -i ens4 -f "host 1.2.3.4"
Running as user "root" and group "root". This could be dangerous.
Capturing on 'ens4'
    1 0.000000000   10.128.0.2 → 1.2.3.4      ICMP 42 Echo (ping) request  id=0x0000, seq=0/0, ttl=64
```

d. Observation: the packet being sent from the attacker is received on the tshark showing that the request was successfully spoofed as the destination matches the destination on my ip fields.

Explanation: the sent message shows that the packets was generated and transmitted while the tshark output shows that the destination and source match the ones specified in my script

Code explanation:

a = IP(): Create an IP packet object a.

a.dst = '1.2.3.4': Set the destination IP address for the IP packet.

b = ICMP(): Create an ICMP packet object b.

p = a/b: Combine the IP and ICMP packets to create a new packet p.

ls(a): Display the fields and their values of the IP packet a.

send(p, iface='br-08b5583a0b6c'): Send the created packet p on the specified network interface (br-08b5583a0b6c in this case).

3. Task 1.3

a. Code snippet

```
#!/usr/bin/env python3

from scapy.all import *
import sys

target_ip = '8.8.4.4'
ttl_value = int(sys.argv[1])

# Craft IP and ICMP packets
ip_packet = IP(dst=target_ip, ttl=ttl_value)
icmp_packet = ICMP()

# Combine IP and ICMP packets
request_packet = ip_packet / icmp_packet

# Send packet and receive response
response = sr1(request_packet, timeout=2, verbose=1)

if response:
    print("Source:", response.src)
else:
    print("No response received.")
```

b. Output

```
Received 19 packets, got 0 answers, remaining 1 packets
No response received.
root@instance-1:/volumes# ./task1.3.py 90
Begin emission:
Finished sending 1 packets.

Received 5 packets, got 1 answers, remaining 0 packets
Source: 8.8.8.8
root@instance-1:/volumes# ./task1.3.py 1
Begin emission:
Finished sending 1 packets.

Received 3 packets, got 0 answers, remaining 1 packets
No response received.
root@instance-1:/volumes# ./task1.3.py 2
Begin emission:
Finished sending 1 packets.

Received 5 packets, got 0 answers, remaining 1 packets
No response received.
root@instance-1:/volumes# ./task1.3.py 90
Begin emission:
Finished sending 1 packets.

Received 5 packets, got 1 answers, remaining 0 packets
Source: 8.8.8.8
root@instance-1:/volumes# ./task1.3.py 900
Begin emission:

Received 6 packets, got 0 answers, remaining 1 packets
No response received.
root@instance-1:/volumes# ./task1.3.py 80
Begin emission:
Finished sending 1 packets.

Received 4 packets, got 1 answers, remaining 0 packets
Source: 8.8.8.8
root@instance-1:/volumes# ./task1.3.py 80
Begin emission:
Finished sending 1 packets.

Received 5 packets, got 1 answers, remaining 0 packets
Source: 8.8.4.4
root@instance-1:/volumes# ./task1.3.py 10
Begin emission:
Finished sending 1 packets.

Received 5 packets, got 0 answers, remaining 1 packets
No response received.
root@instance-1:/volumes# ./task1.3.py 70
Begin emission:
Finished sending 1 packets.

Received 6 packets, got 1 answers, remaining 0 packets
Source: 8.8.4.4
```

c. Observation: here there was a lack of response when the ttl_values was 1,2…60. When the ttl_value was high like 70, 80, or 90. The packets were received and a response was sent. I could also only get responses from the specified ip addresses in my code. I tried 8.8.8.8 and 8.8.8.4, both received responses when the ttl_value was high.

d. Explanation: the reason for not receiving requests from other ip addresses could be due to their personal security configuration blocking icmp packets or stopping the responses to icmp packets. I don't think this is due to my computer settings because I tried adjusting my security configurations on my vm (adjusting firewall network rules and network tags on vm instances). The packets also showed up on tshark showing that it’s being transmitted

e. **Code explanation**:Imports necessary functions from the Scapy library.

Defines the target IP (8.8.4.4) and TTL value based on the command-line argument.

Crafts an ICMP Echo Request packet with the specified destination IP and TTL.

Sends the crafted packet (request_packet) using sr1 (send and receive in one function) and waits for a response within a timeout of 2 seconds.If a response is received, it prints the source IP of the response; otherwise, it prints that no response was received.

This script pings a target IP with a specific TTL value and checks if a response is received.

4. Task 1.4

a. Code snippet

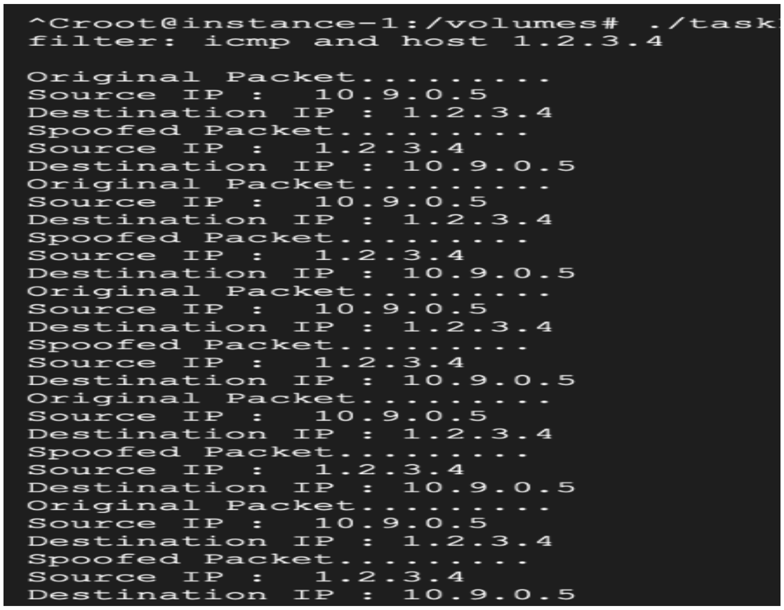
i.

SSH-in-browser UPLOAD FILE DOWNL

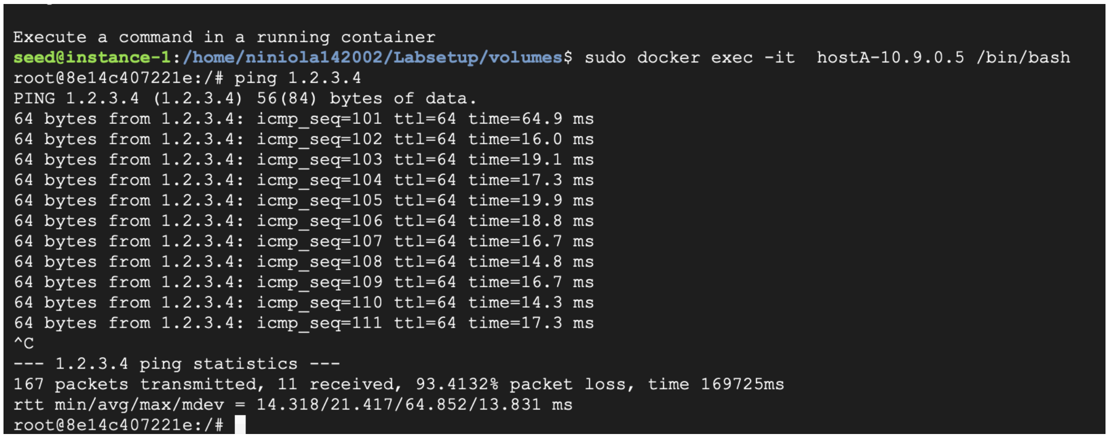
```
import sys
import os
from scapy.all import *


def spoof_pkt(pkt):
    # sniff and print out icmp echo request packet
    if ICMP in pkt and pkt[ICMP].type == 8:
        print("Original Packet.........")
        print("Source IP : ", pkt[IP].src)
        print("Destination IP :", pkt[IP].dst)

        # spoof an icmp echo reply packet
        # swap srcip and dstip
        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

        print("Spoofed Packet.........")
        print("Source IP : ", newpkt[IP].src)
        print("Destination IP :", newpkt[IP].dst)

        send(newpkt, verbose=0)
filter = 'icmp and host 1.2.3.4'
filter = 'icmp and host 8.8.8.8'
print("filter: {}\n".format(filter))
pkt = sniff(iface = 'br-08b5583a0b6c', filter=filter, prn=spoof_pkt)
~
~
```

b. Output

i. ping 1.2.3.4 # a non-existing host on the Internet

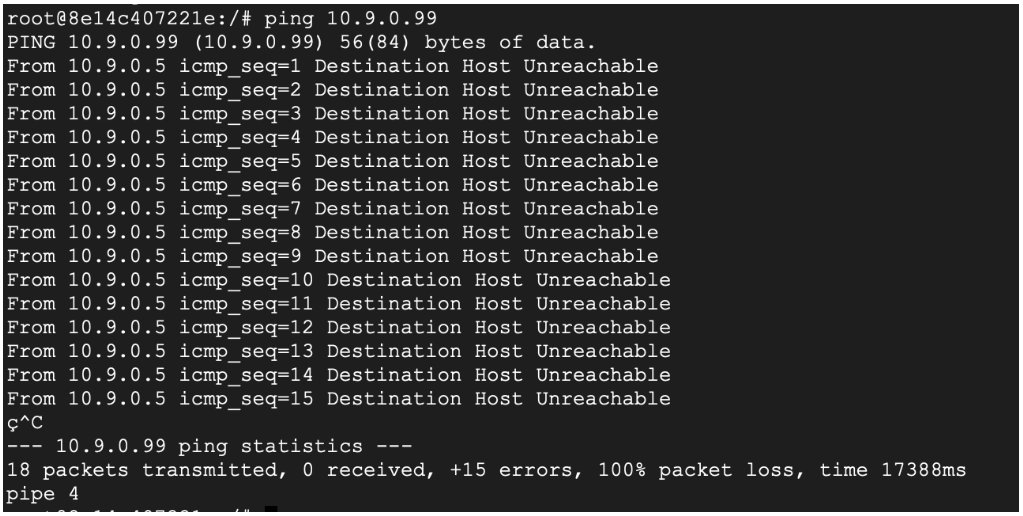


```
^Croot@instance-1:/volumes# ./task
filter: icmp and host 1.2.3.4

Original Packet.........
Source IP :  10.9.0.5
Destination IP : 1.2.3.4
Spoofed Packet.........
Source IP :  1.2.3.4
Destination IP : 10.9.0.5
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 1.2.3.4
Spoofed Packet.........
Source IP :  1.2.3.4
Destination IP : 10.9.0.5
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 1.2.3.4
Spoofed Packet.........
Source IP :  1.2.3.4
Destination IP : 10.9.0.5
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 1.2.3.4
Spoofed Packet.........
Source IP :  1.2.3.4
Destination IP : 10.9.0.5
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 1.2.3.4
Spoofed Packet.........
Source IP :  1.2.3.4
Destination IP : 10.9.0.5
```

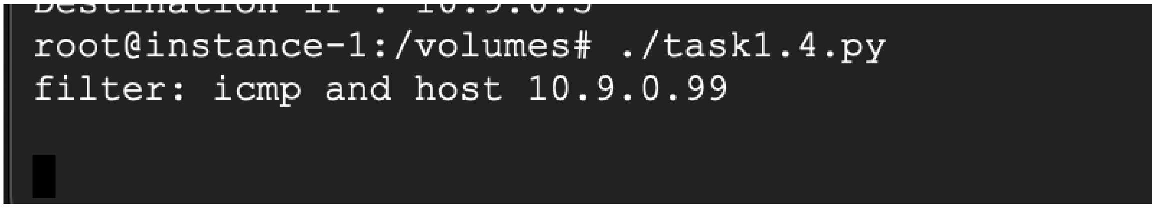


```
Execute a command in a running container
seed@instance-1:/home/niniola142002/Labsetup/volumes$ sudo docker exec -it  hostA-10.9.0.5 /bin/bash
root@8e14c407221e:/# ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
64 bytes from 1.2.3.4: icmp_seq=101 ttl=64 time=64.9 ms
64 bytes from 1.2.3.4: icmp_seq=102 ttl=64 time=16.0 ms
64 bytes from 1.2.3.4: icmp_seq=103 ttl=64 time=19.1 ms
64 bytes from 1.2.3.4: icmp_seq=104 ttl=64 time=17.3 ms
64 bytes from 1.2.3.4: icmp_seq=105 ttl=64 time=19.9 ms
64 bytes from 1.2.3.4: icmp_seq=106 ttl=64 time=18.8 ms
64 bytes from 1.2.3.4: icmp_seq=107 ttl=64 time=16.7 ms
64 bytes from 1.2.3.4: icmp_seq=108 ttl=64 time=14.8 ms
64 bytes from 1.2.3.4: icmp_seq=109 ttl=64 time=16.7 ms
64 bytes from 1.2.3.4: icmp_seq=110 ttl=64 time=14.3 ms
64 bytes from 1.2.3.4: icmp_seq=111 ttl=64 time=17.3 ms
^C
--- 1.2.3.4 ping statistics ---
167 packets transmitted, 11 received, 93.4132% packet loss, time 169725ms
rtt min/avg/max/mdev = 14.318/21.417/64.852/13.831 ms
root@8e14c407221e:/# 
```

ii. ping 10.9.0.99 # a non-existing host on the LAN

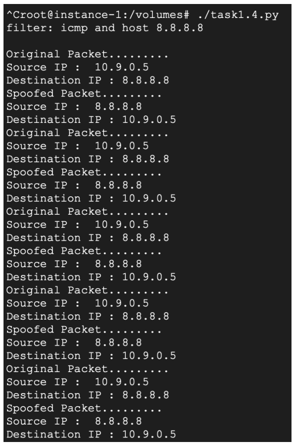


```
root@8e14c407221e:/# ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
From 10.9.0.5 icmp_seq=1 Destination Host Unreachable
From 10.9.0.5 icmp_seq=2 Destination Host Unreachable
From 10.9.0.5 icmp_seq=3 Destination Host Unreachable
From 10.9.0.5 icmp_seq=4 Destination Host Unreachable
From 10.9.0.5 icmp_seq=5 Destination Host Unreachable
From 10.9.0.5 icmp_seq=6 Destination Host Unreachable
From 10.9.0.5 icmp_seq=7 Destination Host Unreachable
From 10.9.0.5 icmp_seq=8 Destination Host Unreachable
From 10.9.0.5 icmp_seq=9 Destination Host Unreachable
From 10.9.0.5 icmp_seq=10 Destination Host Unreachable
From 10.9.0.5 icmp_seq=11 Destination Host Unreachable
From 10.9.0.5 icmp_seq=12 Destination Host Unreachable
From 10.9.0.5 icmp_seq=13 Destination Host Unreachable
From 10.9.0.5 icmp_seq=14 Destination Host Unreachable
From 10.9.0.5 icmp_seq=15 Destination Host Unreachable
ç^C
--- 10.9.0.99 ping statistics ---
18 packets transmitted, 0 received, +15 errors, 100% packet loss, time 17388ms
pipe 4
```

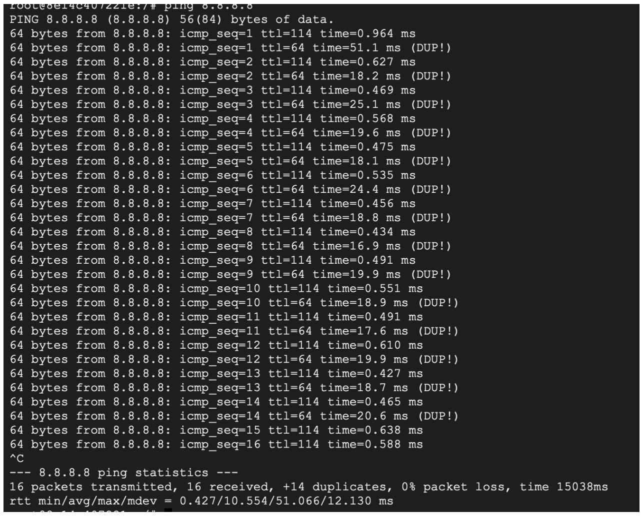


```
Destination IP : 10.9.0.5
root@instance-1:/volumes# ./task1.4.py
filter: icmp and host 10.9.0.99
```

iii. ping 8.8.8.8 # an existing host on the Internet



```
^Croot@instance-1:/volumes# ./task1.4.py
filter: icmp and host 8.8.8.8

Original Packet.........
Source IP :  10.9.0.5
Destination IP : 8.8.8.8
Spoofed Packet.........
Source IP :  8.8.8.8
Destination IP : 10.9.0.5
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 8.8.8.8
Spoofed Packet.........
Source IP :  8.8.8.8
Destination IP : 10.9.0.5
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 8.8.8.8
Spoofed Packet.........
Source IP :  8.8.8.8
Destination IP : 10.9.0.5
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 8.8.8.8
Spoofed Packet.........
Source IP :  8.8.8.8
Destination IP : 10.9.0.5
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 8.8.8.8
Spoofed Packet.........
Source IP :  8.8.8.8
Destination IP : 10.9.0.5
```



```
root@8e14c407221e:/# 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=114 time=0.964 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=51.1 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=0.627 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=18.2 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=0.469 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=25.1 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=114 time=0.568 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=19.6 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=5 ttl=114 time=0.475 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=18.1 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=6 ttl=114 time=0.535 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=64 time=24.4 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=7 ttl=114 time=0.456 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=64 time=18.8 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=8 ttl=114 time=0.434 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=64 time=16.9 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=9 ttl=114 time=0.491 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=64 time=19.9 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=10 ttl=114 time=0.551 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=64 time=18.9 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=11 ttl=114 time=0.491 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=64 time=17.6 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=12 ttl=114 time=0.610 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=64 time=19.9 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=13 ttl=114 time=0.427 ms
64 bytes from 8.8.8.8: icmp_seq=13 ttl=64 time=18.7 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=14 ttl=114 time=0.465 ms
64 bytes from 8.8.8.8: icmp_seq=14 ttl=64 time=20.6 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=15 ttl=114 time=0.638 ms
64 bytes from 8.8.8.8: icmp_seq=16 ttl=114 time=0.588 ms
^C
--- 8.8.8.8 ping statistics ---
16 packets transmitted, 16 received, +14 duplicates, 0% packet loss, time 15038ms
rtt min/avg/max/mdev = 0.427/10.554/51.066/12.130 ms
```

c. Observation : the pings to 1.2.3.4 and 8.8.8.8 received replies indicating that they are alive and their packets were spoofed. while the ping to 10.9.0.5 received no replies and the host was unreachable, therefore, its packets couldn’t be spoofed

d. Explanation: for 1.2.3.4, the program immediately sends a spoofed icmp echo reply whenever it detects an icmp request regardless of whether it exists or not. For the 8.8.8.8 ping, it exists so icmp echo requests send a response mimicking the actual target. Lastly for ping 10.9.0.5, this host is unreachable because it doesn't exist on the network, therefore it has no mac address, which means the packet cannot be created without the mapping of the mac address.

e. **Code explanation:** Defines a function spoof_pkt(pkt) to be called when sniffing ICMP packets.Checks if the packet is an ICMP Echo Request (type == 8).

If it is, prints information about the original packet. Crafts a new ICMP Echo Reply packet with swapped source and destination IP addresses.Prints information about the spoofed packet. Sends the spoofed packet using send() with verbose=0 to suppress output. Sets a filter to sniff ICMP packets targeting either 1.2.3.4 or 8.8.8.8.

Invokes the sniff() function to capture packets, applying the filter and calling spoof_pkt for each matching packet. This script essentially spoofs ICMP Echo Replies in response to ICMP Echo Requests for a specified target IP address