H M Mythreya
PES2UG20CS130
CNS-Lab Week-1

Setting up the lab

First thing I did was to setup the docker file given. This sets up a simulated environment with 3 machines, i.e, Host A, Host B and the Attacker.

Task 1.1A: Sniffing Packets

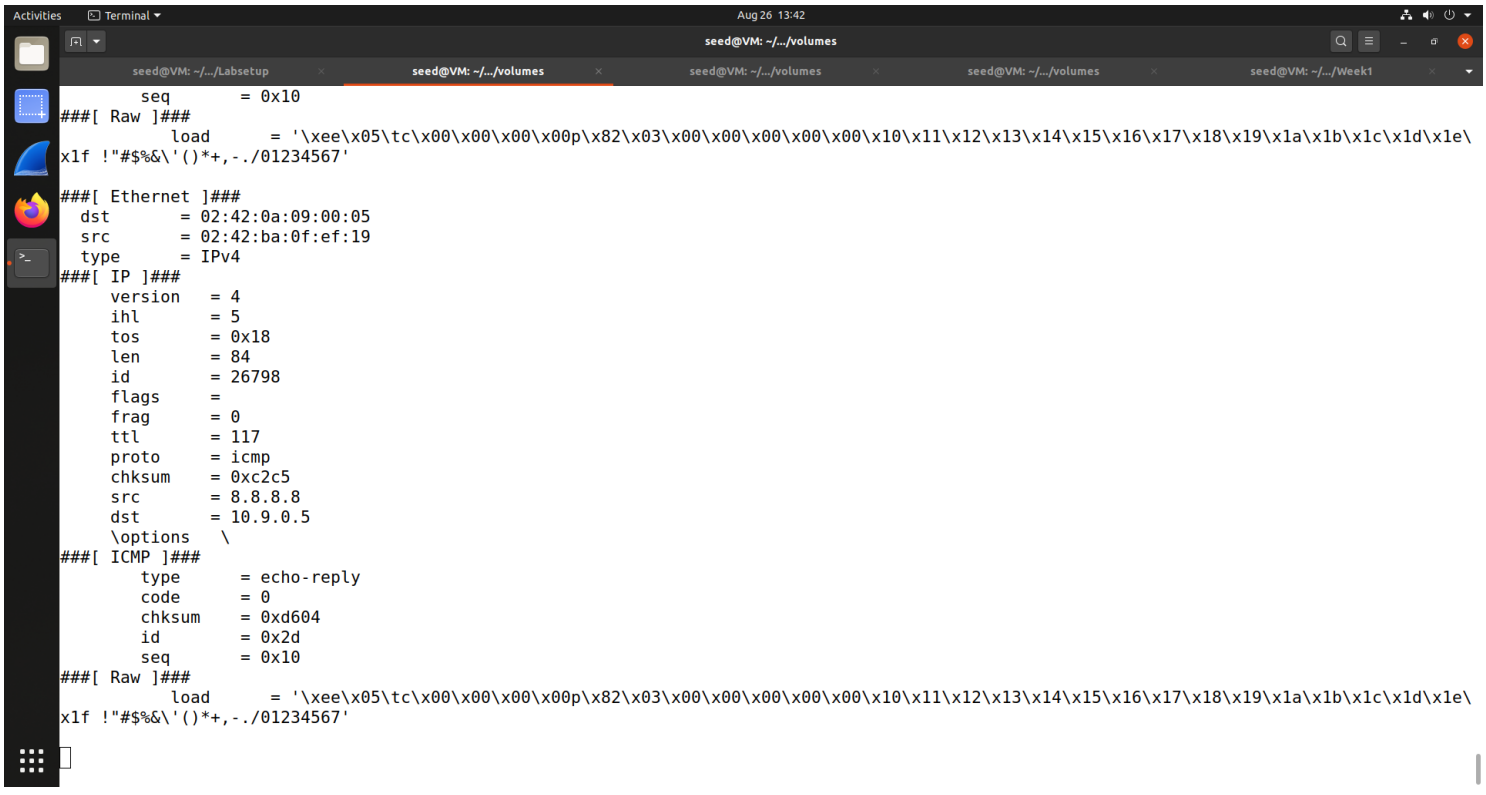
For this task, scapy was used. Initially, a ping command is run from host A, which pings host B. A ping is simply an ICMP packet used to check the presence of an I.P. At this point, on the attacker machine, the sniffing code is run.

The most important code is:

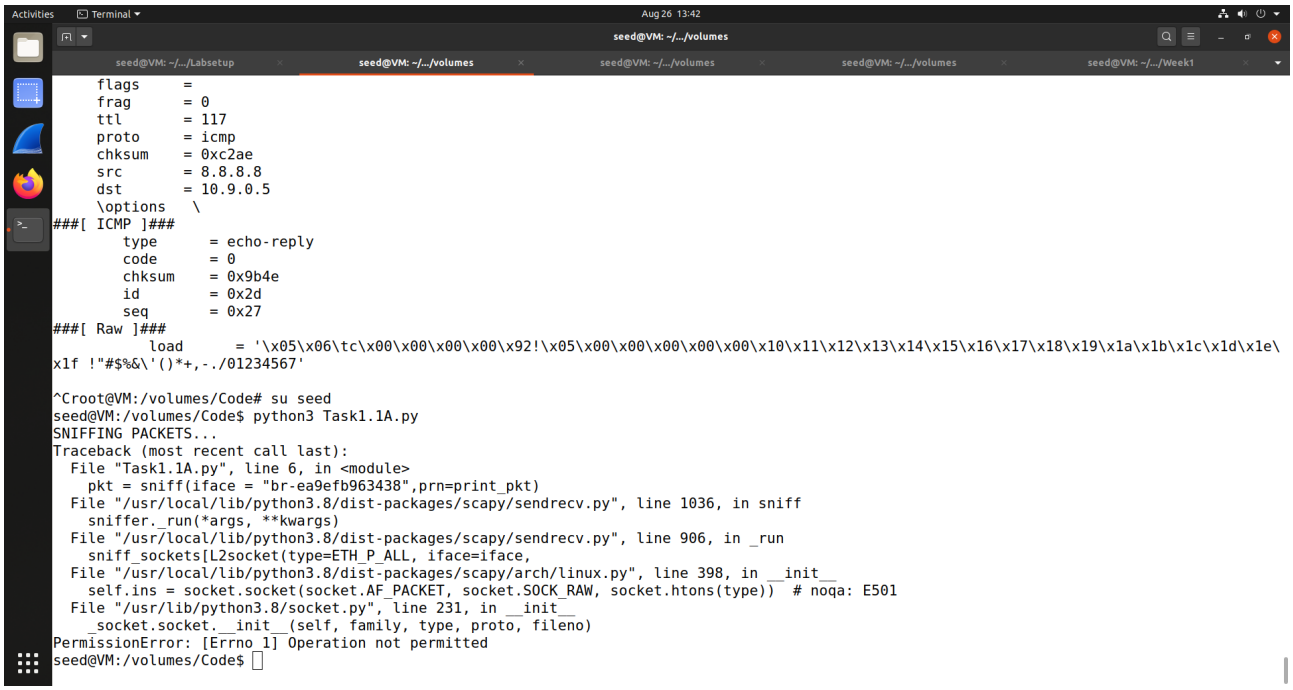
```
pkt = sniff(iface = "br-ea9efb963438",prn=print_pkt)
```

Here, a scapy function called sniff() is called. “iface” here stands for interface and the interface from which you want to sniff goes here. “prn” is the function that is going to be called after sniffing, where in we can do certain operations to the packet, in this case we just print it.

Output



Observation: Running the same command without root privileges doesn't work. A PermissionError is returned upon calling sniff()



Task 1.1B: Sniffing Packets

Filters:

In the real world, a network will have thousands if not hundreds of thousands of packets flowing to and from. Going through every packet would be impractical, hence we use Filters. Filters only show the relevant packets depending on the filter. For example, using an ICMP filter will only sniff ICMP packets, and so on.

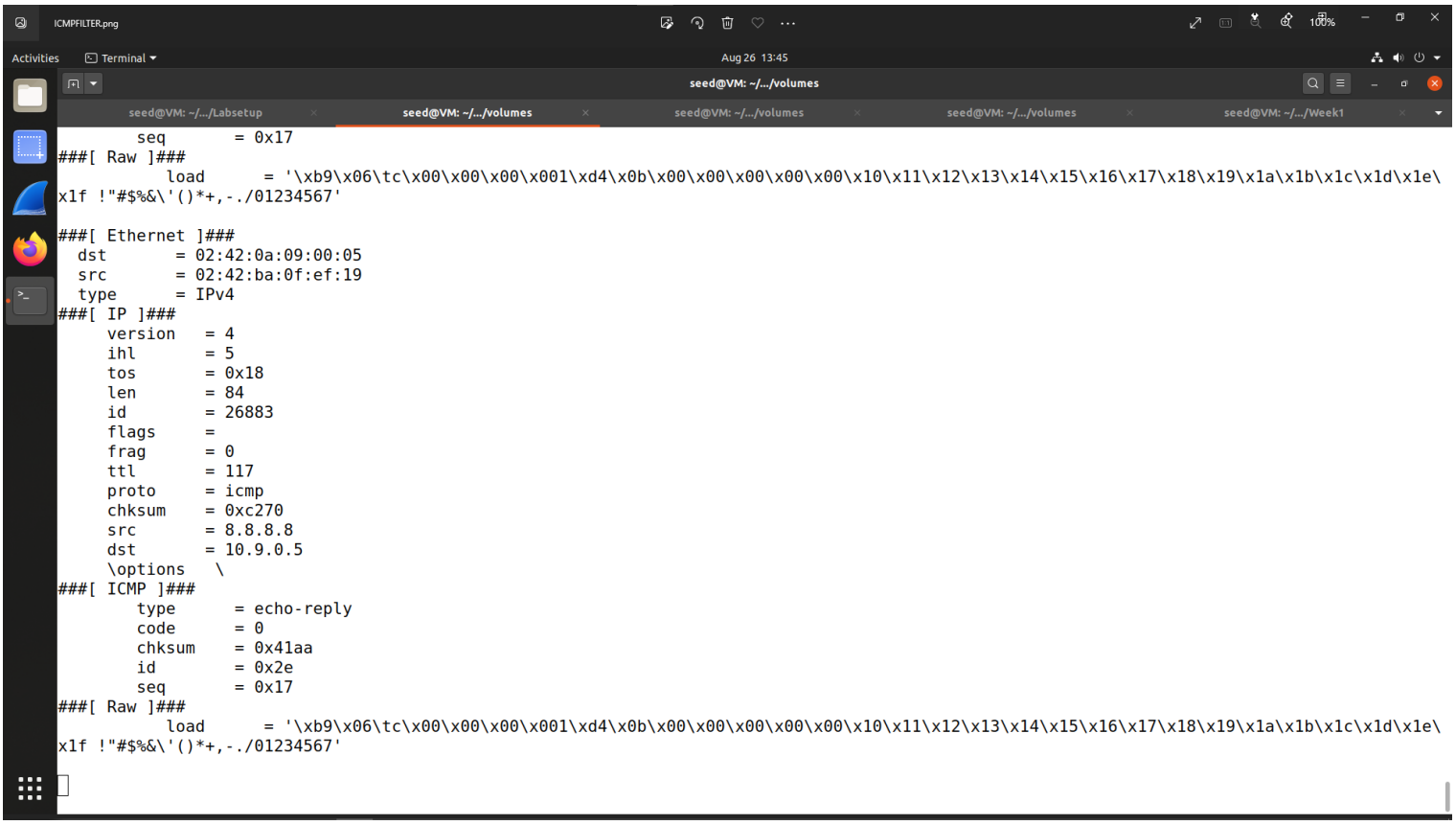
First, Host A pings Host B using the ping command. The attacker then runs the sniffing code with any filter they may desire.

Filter 1) ICMP

```
pkt = sniff (iface = "br-ea9efb963438",filter='icmp', prn=print_pkt)
```

Here, we are using the same code as before, but with a new argument, the filter argument. For now let us pass in “icmp” as the argument and see what happens.

Output



Here we can see that, the packets being sniffed are ICMP packets. We are sniffing the packets sent out by Host A, since “ping” is an ICMP protocol.

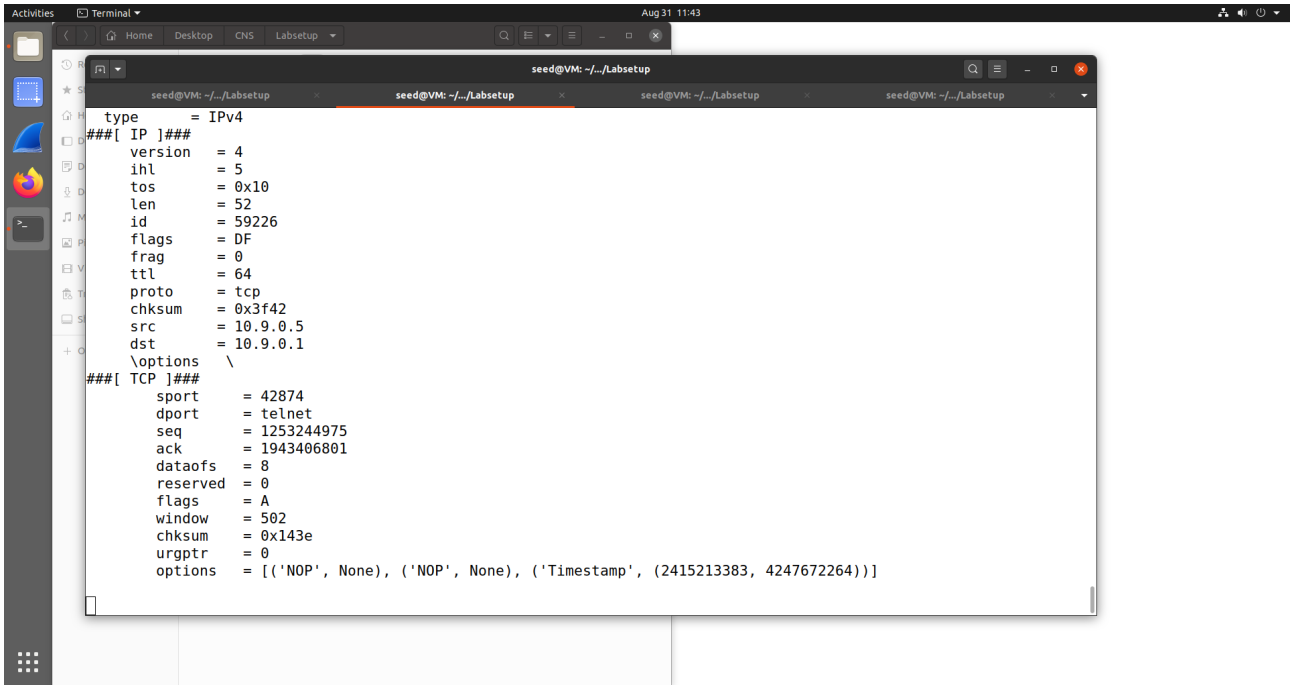
Filter 2) TCP, destination port number 23.

```
pkt = sniff (iface = "br-ea9efb963438",filter='tcp and src host 10.9.0.5 and dst port 23', prn=print_pkt)
```

Under filter, our first filter we are applying is “tcp”. It is followed by an and clause, meaning it has to be tcp AND whatever filter follows. The next filter is src host …. This means the packet must have a source IP of whatever we desire. The final filter being used is destination port number, in this case it is 23.

First we run a telnet command from host A. The telnet command uses tcp protocol and runs on port 23, hence our filter should be able to sniff these packets generated by the telnet command

Output



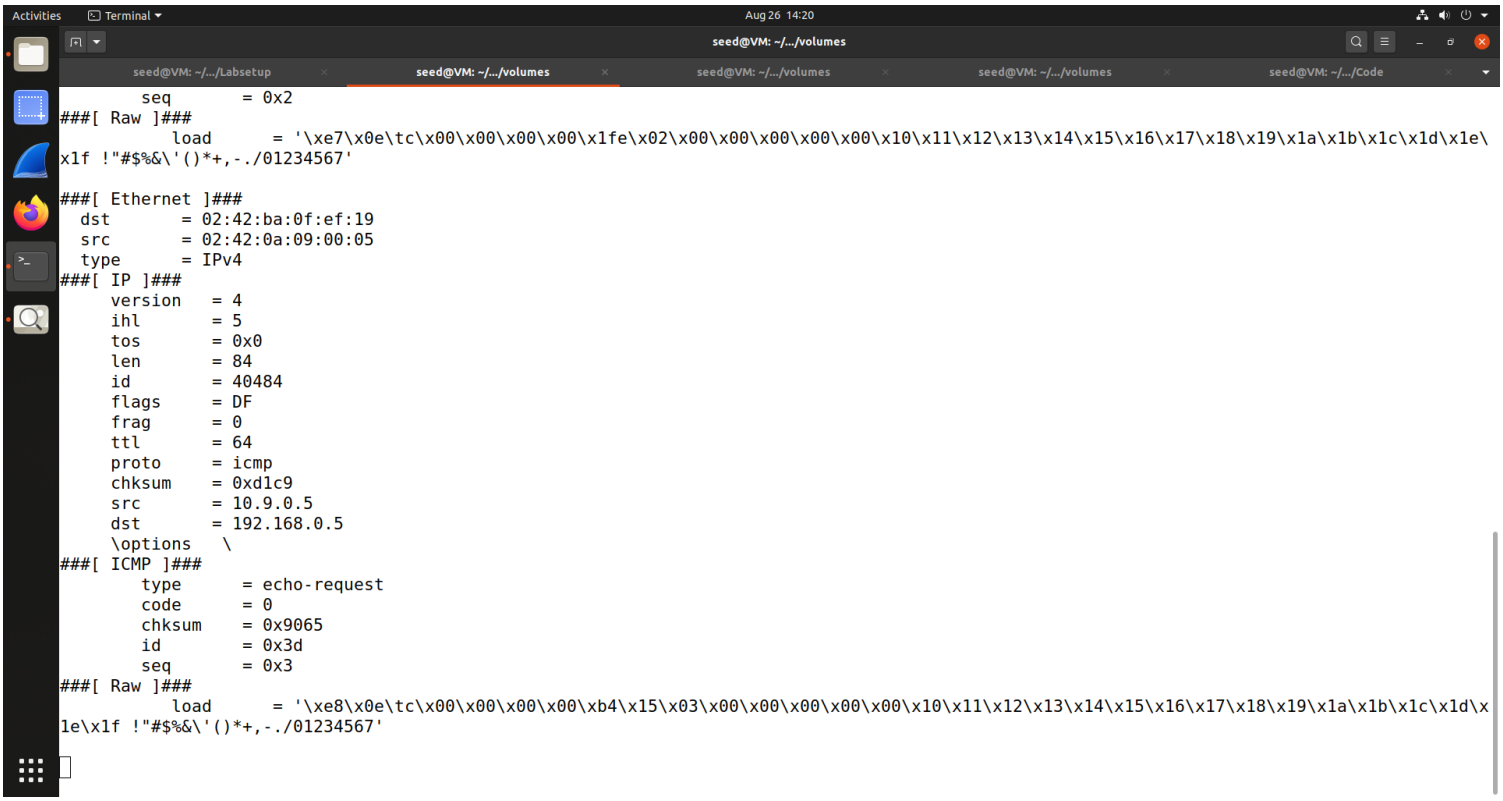
Filter 3) Packets from/to a particular subnet.

```
pkt = sniff(iface = "br-ea9efb963438",filter='src net 10.9.0.0/24', prn=print_pkt)
```

Here we are using a subnet filter. The ip format follows cidr Notation. The /24 means the first 24 bits are for host IP's, and the last 8 bits are for the subnet. Hence the subnet mask here is going to be 255.255.255.0. This filter will sniff all packets originating from this subnet.

We will ping an arbitrary IP in the subnet "10.9.0.0/24" from Host A to observe the working of this filter.

Output



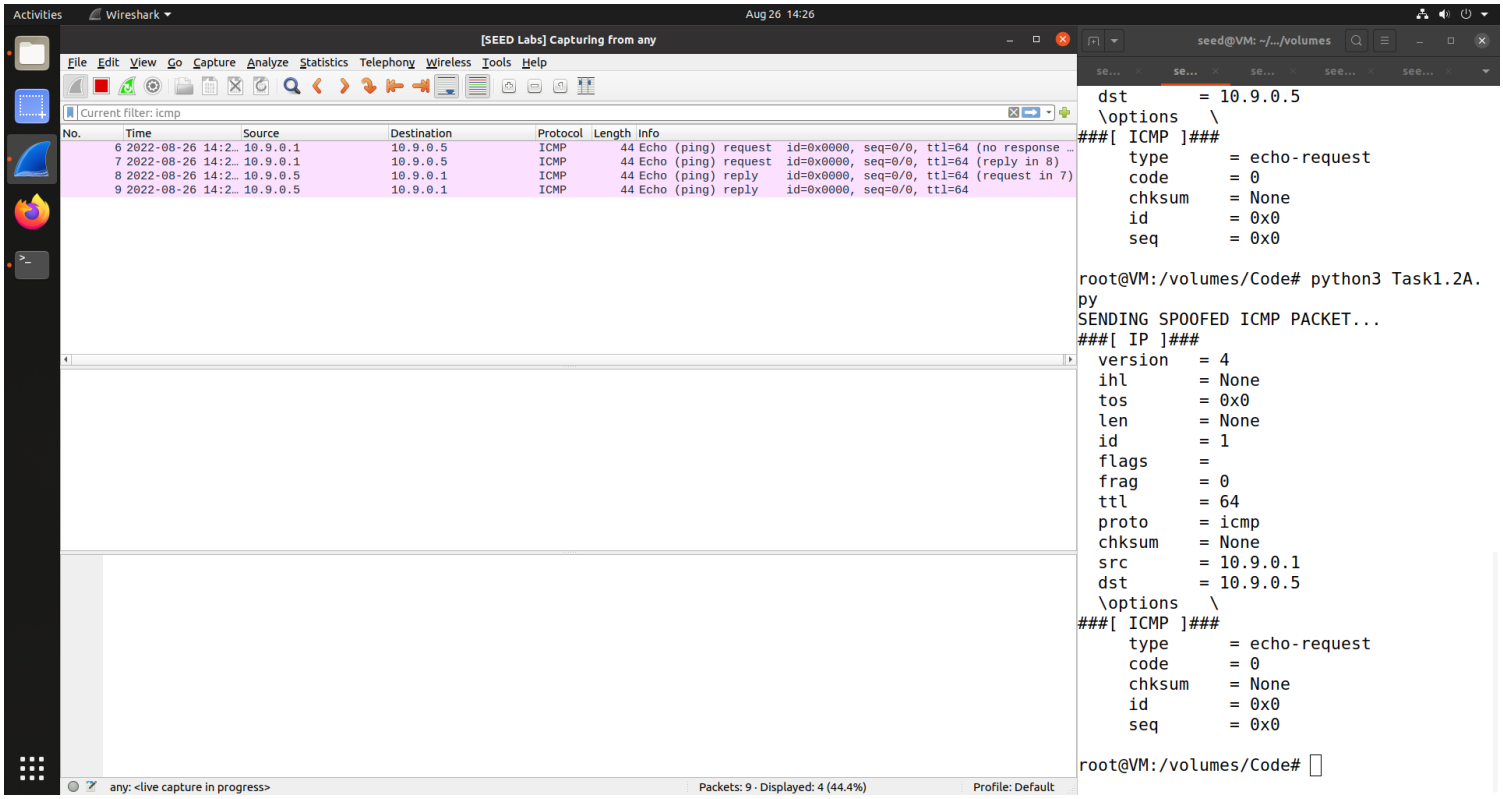
Task 1.2A: Spoofing Packets

Spoofing is the process of pretending/assuming someone else's ID to trick the victim into accepting communication with the attacker. The attacker's actual ID is hidden.

```
IPLayer = IP()
IPLayer.src="10.9.0.1"
IPLayer.dst="10.9.0.5"
ICMPpkt = ICMP()
pkt = IPLayer/ICMPpkt
pkt.show()
send(pkt,verbose=0)
```

Here, we are creating an IPLayer using scapy's IP() function, and we are setting the source to "10.9.0.1". This IP address can be set to any arbitrary IP and hence we can pretend to be some other machine and hide our own identity.

Output



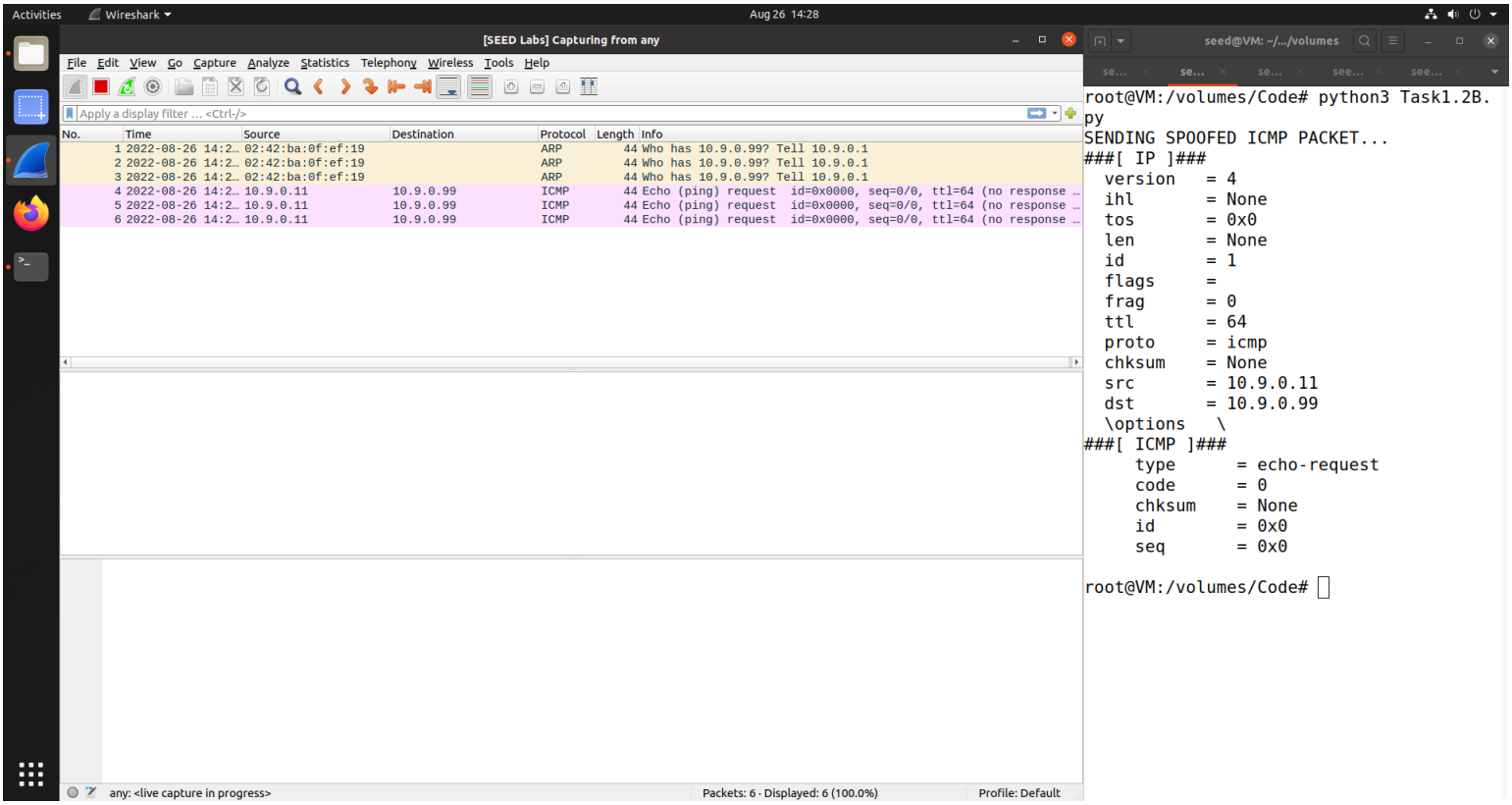
Here we can see that, there is a reply from "10.9.0.5" (the victim), and that the victim has fallen for the spoof.

Task 1.2A: Spoofing Packets

```
IPLayer = IP()
IPLayer.src="10.9.0.11"
IPLayer.dst="10.9.0.99"
ICMPpkt = ICMP()
pkt = IPLayer/ICMPpkt
pkt. show()
send(pkt,verbose=0)
```

Here we are changing the src and dst to an arbitrary address.

Output



We can see that an ARP packet is sent out, since the location of “10.9.0.99” is unknown.

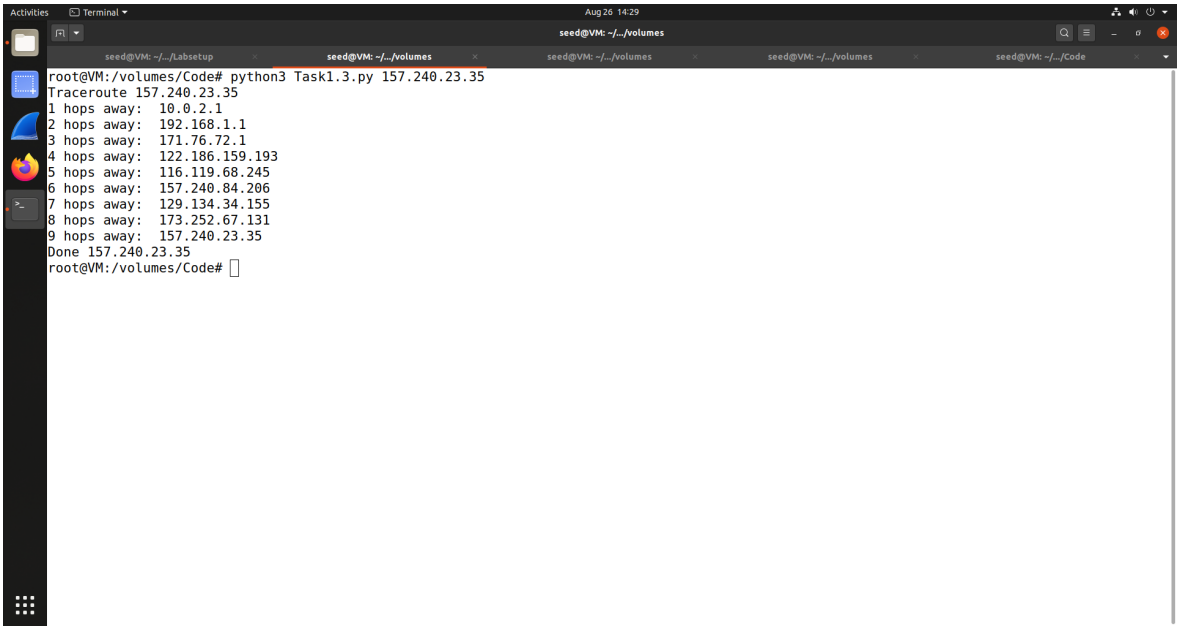
Task 1.3: Traceroute

The traceroute command provides a map of how data flows from your machine to destination in the network. It gives all the hops that your packet takes to reach the destination.

We can recreate the traceroute command in python using scapy.

Let us see the route a packet takes from machine to facebook's servers.

Output



An interesting observation we can make here is that, this is not a set path. Running the command at a later time gives different results. This is because of changes in network, routers going offline/online, packets taking a shorter path, etc.

```
root@VM:/volumes/Code# python3 Task1.3.py 157.240.23.35
Traceroute 157.240.23.35
1 hops away:  10.0.2.1
2 hops away:  192.168.1.1
3 hops away:  171.76.72.1
4 hops away:  125.21.0.185
5 hops away:  116.119.57.97
6 hops away:  182.79.153.40
7 hops away:  116.119.106.152
8 hops away:  157.240.84.206
9 hops away:  129.134.34.151
10 hops away:  173.252.67.25
11 hops away:  157.240.23.35
Done 157.240.23.35
root@VM:/volumes/Code#
```

Task 1.4: Sniffing and then Spoofing

First the victim, Host A, pings a non-existing IP address, let's say "1.2.3.4". Normally, there would be no response, however, the attacker can sniff this ping request and can pretend to be "1.2.3.4", and thus establish connection and attack the victim (spoofing).

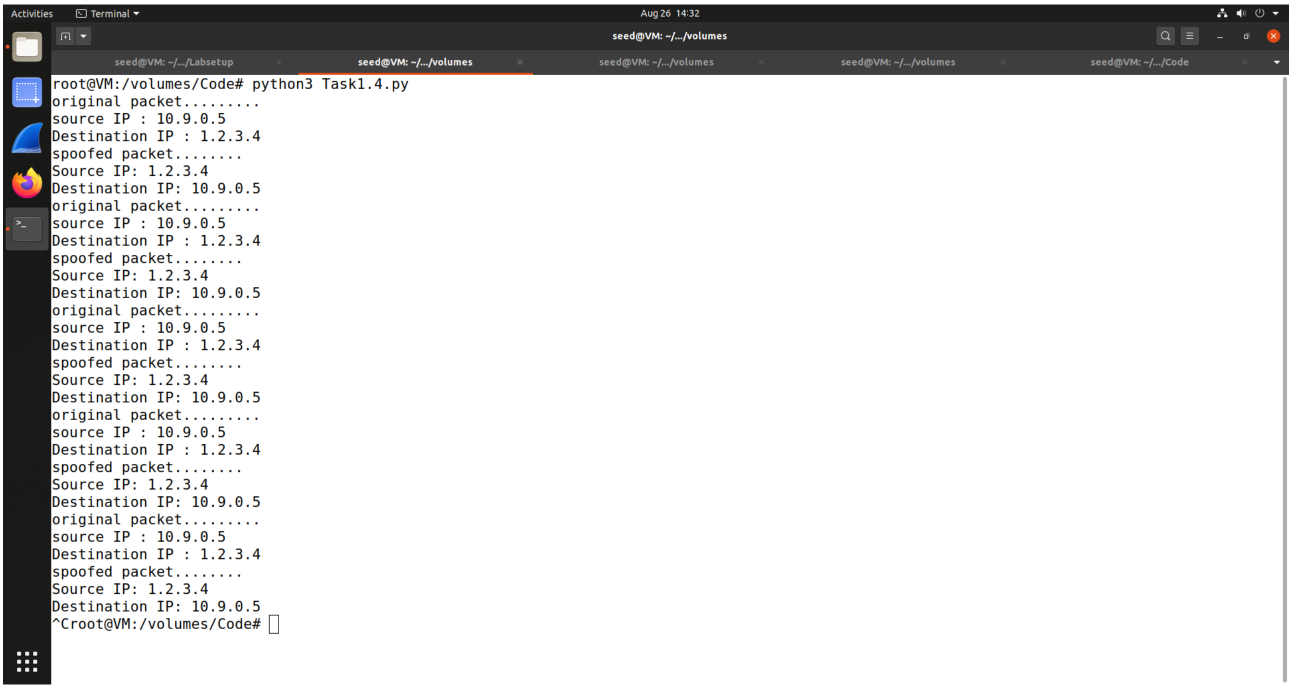
```
pkt = sniff (iface="br-ea9efb963438",filter='icmp and src host 10.9.0.5',
prn=spoof_pkt)
```

Initially, we are sniffing the packets with the filter as seen above. However this time, instead of just printing the sniffed packets, we can extract the src and dst of the packet, and spoof our own packet to pretend to be "1.2.3.4"

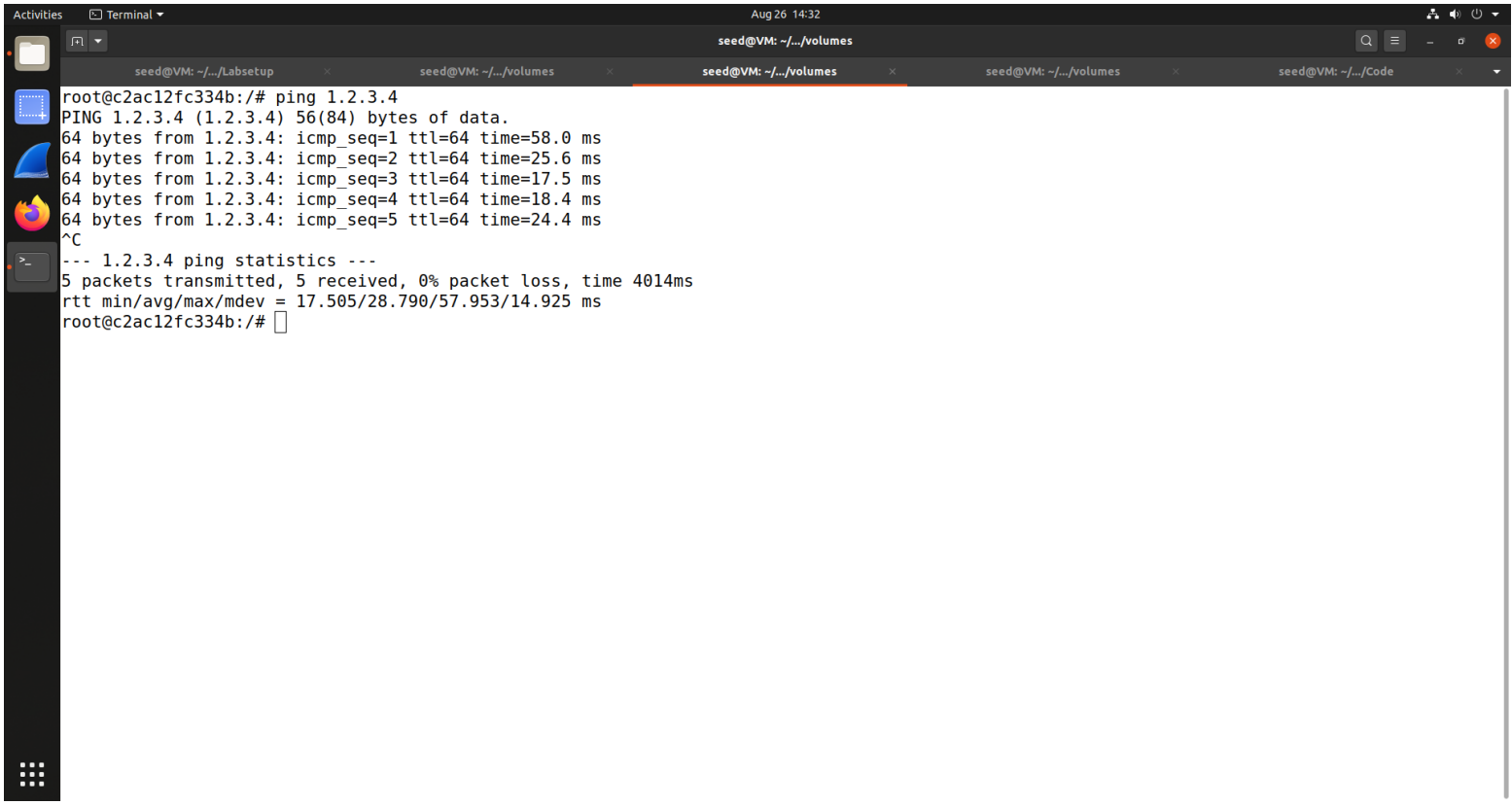
```
srcip = pkt [IP]. dst
dstip = pkt[IP].src
newihl = pkt [IP]. ihl
newtype = 0
newid = pkt [ICMP].id
newseq = pkt [ICMP]. seq
data = pkt [Raw]. load
IPLayer = IP (src=srcip, dst=dstip, ihl=newihl)
ICMPpkt = ICMP (type=newtype, id=newid, seq=newseq)
newpkt = IPLayer/ICMPpkt/data
send (newpkt, verbose=0)
```

We are extracting the ip of the packet using pkt[IP]. Using ".dst" and ".src" will return the destination and source IP respectively. We are then creating an IPLayer, similar to what we did in the Spoofing task, and then sending a response to the victim.

Output (Attacker)



Output(Victim)



We can see that on the victim's machine, there is a response to our ping from "1.2.3.4" even though the machine doesn't exist. This may lead to the victim believe that they are talking to "1.2.3.4", but they're actually talking to the attacker.