CNS LAB

LAB-1

SNIFFING AND SPOOFING

NAME : SIRI S

SEMESTER : 5

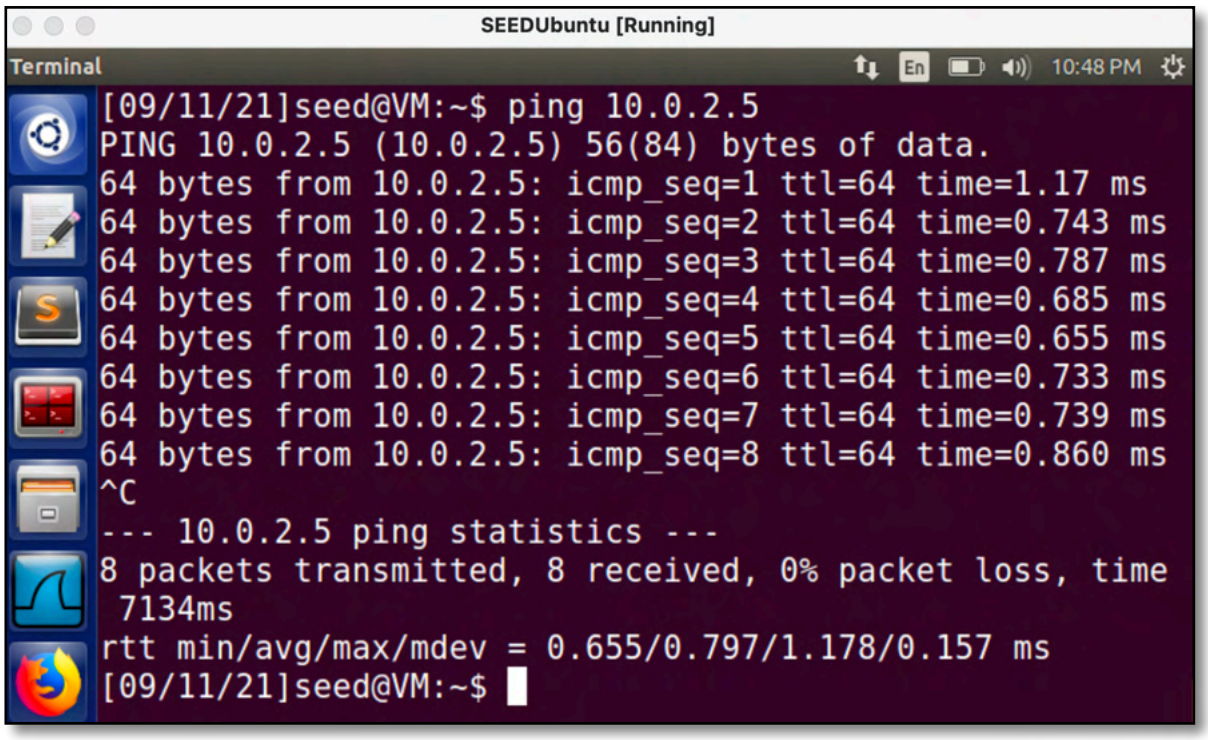
SECTION :H

SRN : PES1UG19CS485

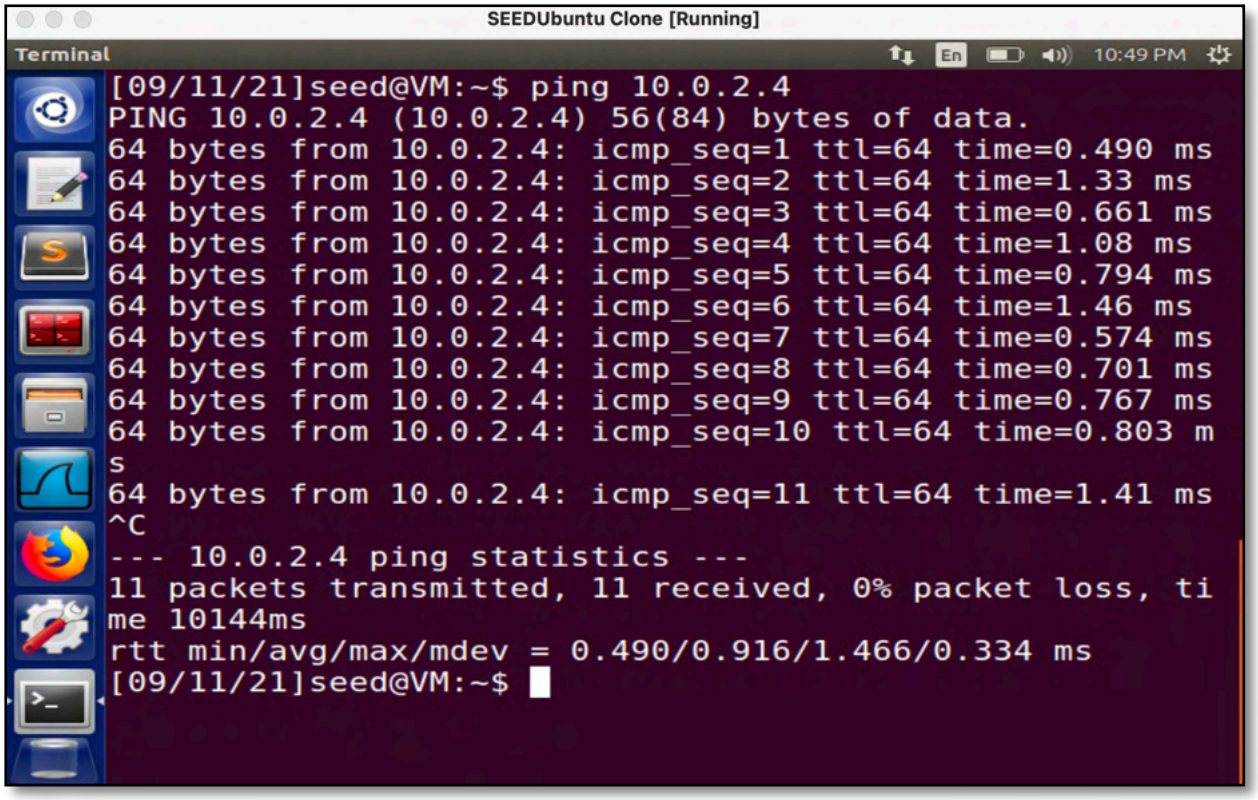
Screenshots of Execution:

Task 0: Checking connection between Attacker and Victim

Attacker (VM1): 10. 0. 2. 4
Victim (VM2 - Clone): 10. 0. 2. 5



Pinging victim from attacker to check connection

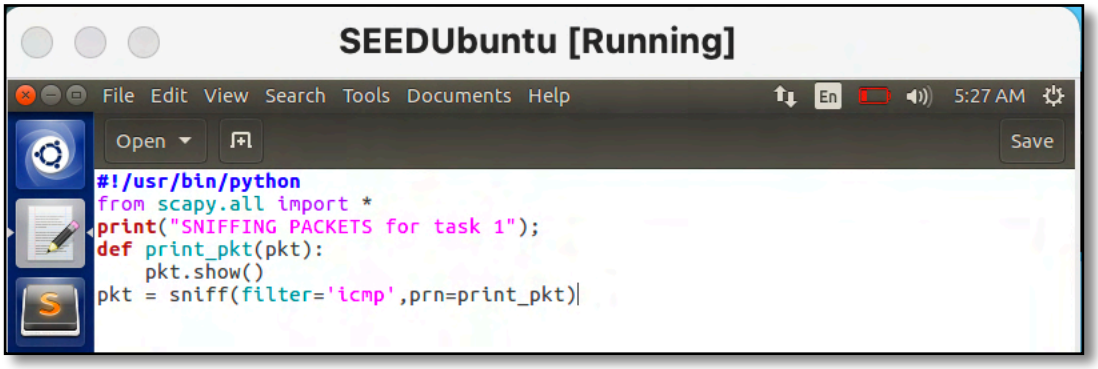


Pinging attacker from victim to check connection

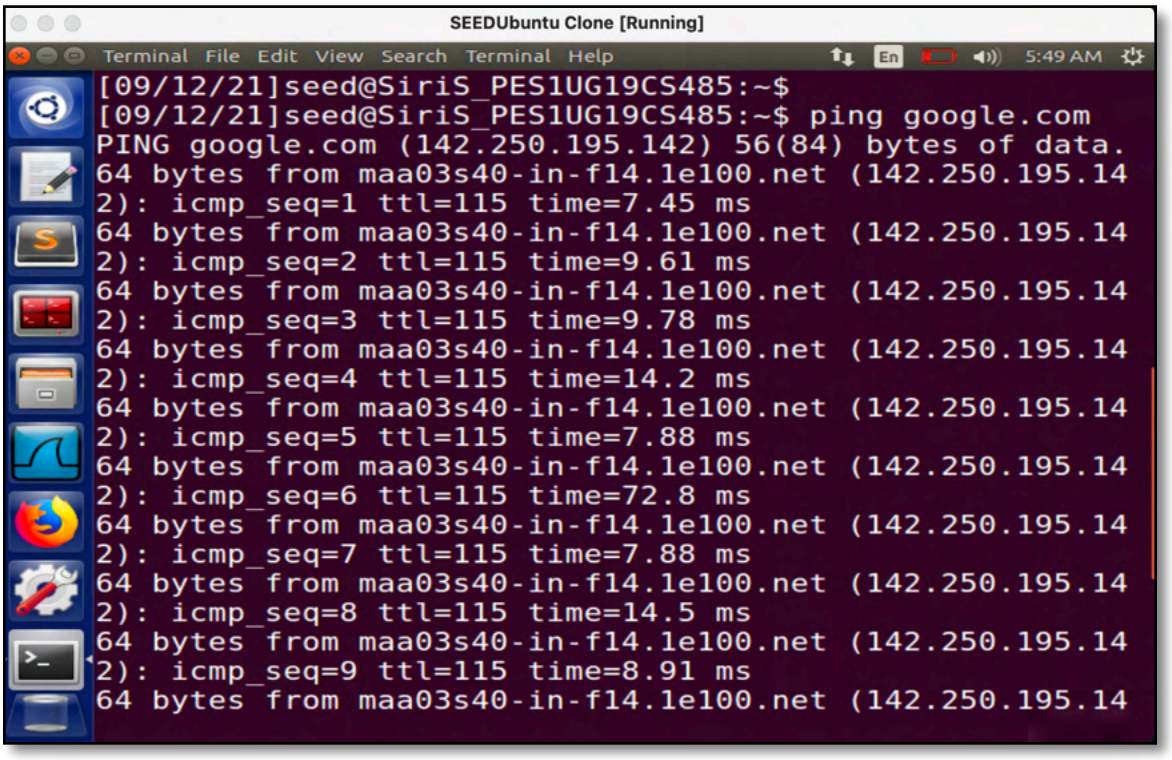
From the above screenshots, we can see that the connection has been successfully established.

Task 1: SNIFFING IP PACKETS

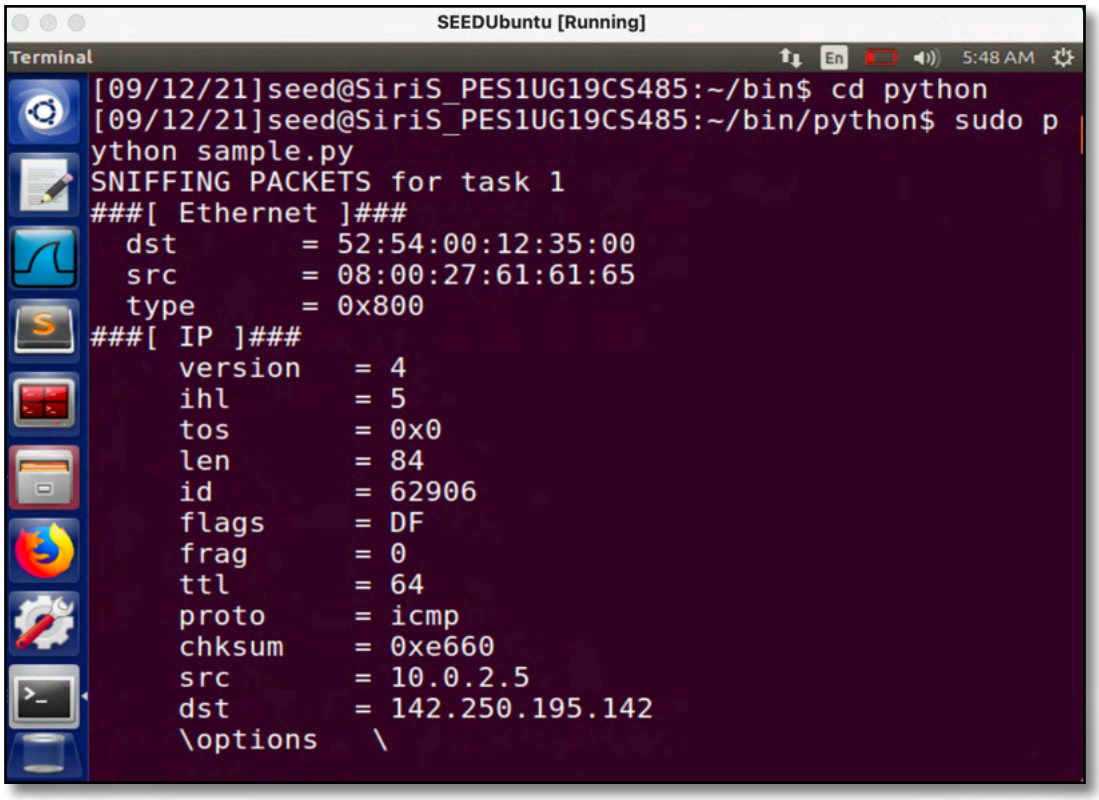
Task 1.1: Sniff IP packets using Scapy

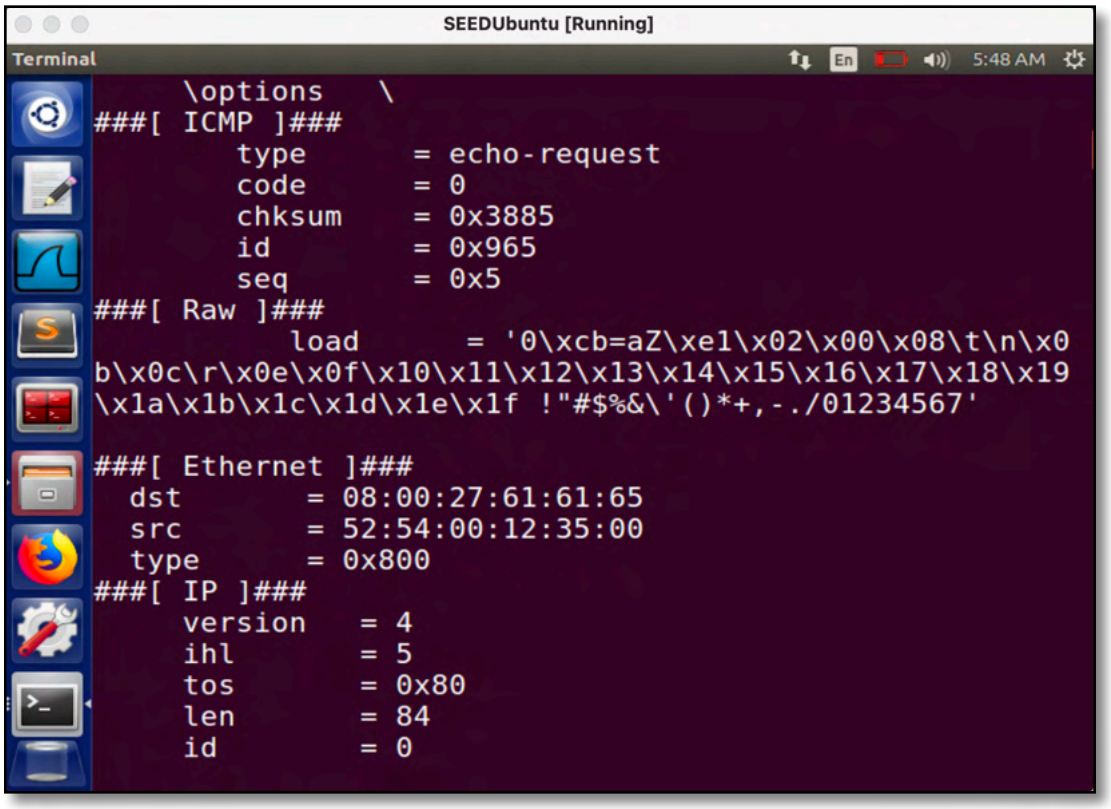


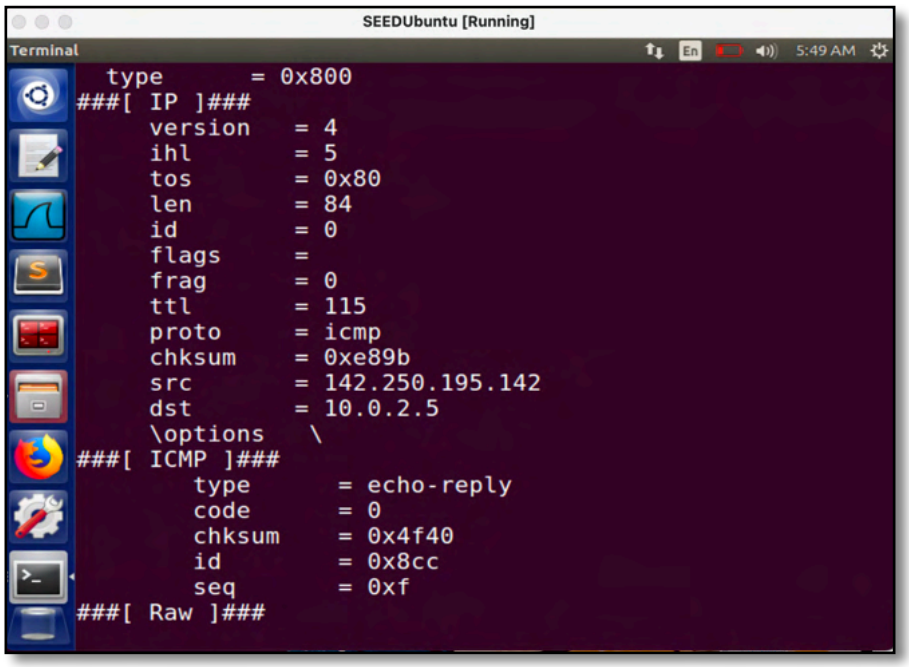
Python program to sniff packets on the attacker VM



In order to create some network activity, I pinged google.com from the victim VM



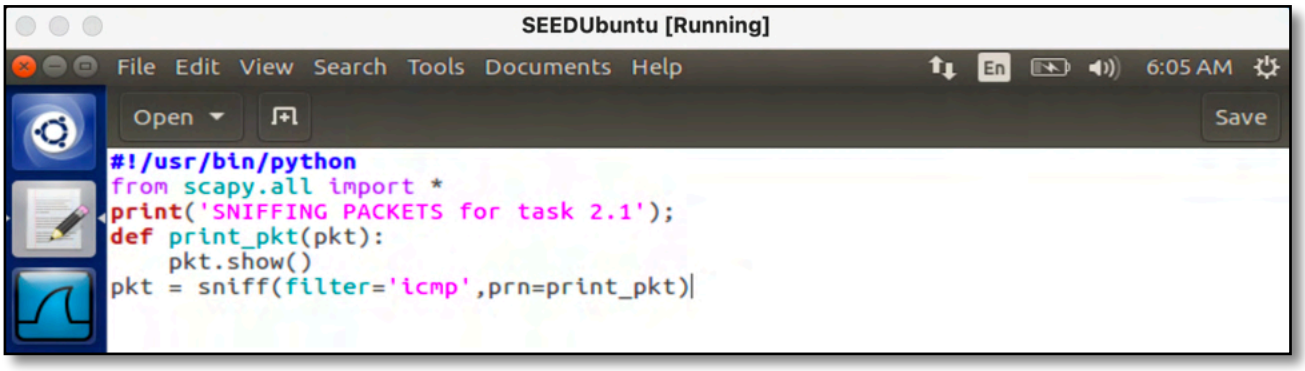




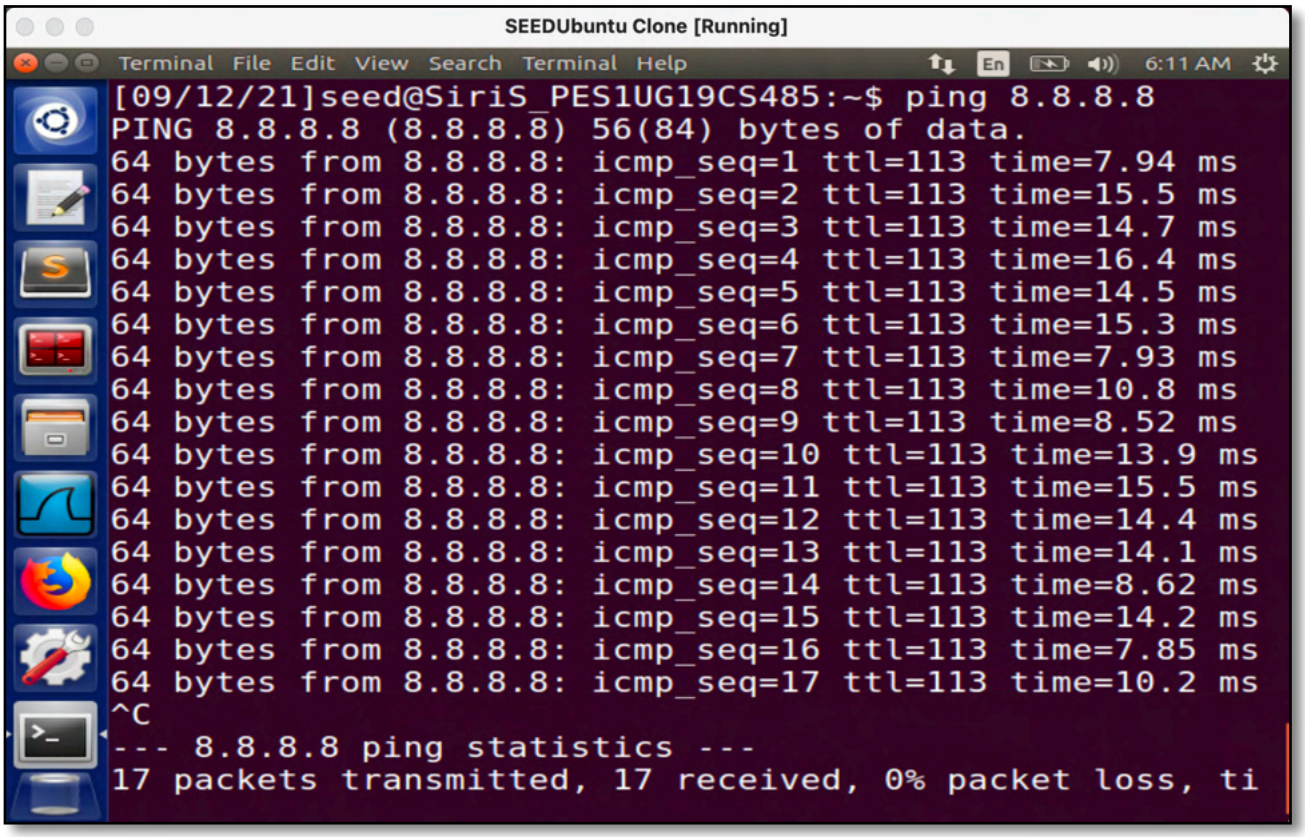
Simultaneously, we run the sample program on the attacker VM so that it can Sniff the network activity created by pinging google on the victim VM. It sniffs the packets because they are on the same network. The above screenshots gives us a detailed overview of every intercepted packet like source and destination mac, ip address of packet, checksum, ttl, the payload contents and so on.

This error is displayed on running the python program without the root (sudo) implicating that the sniffer doesn't run. This happens because the sniff function in the python sniffer program requires root privileges and since that is not granted, it does not run.

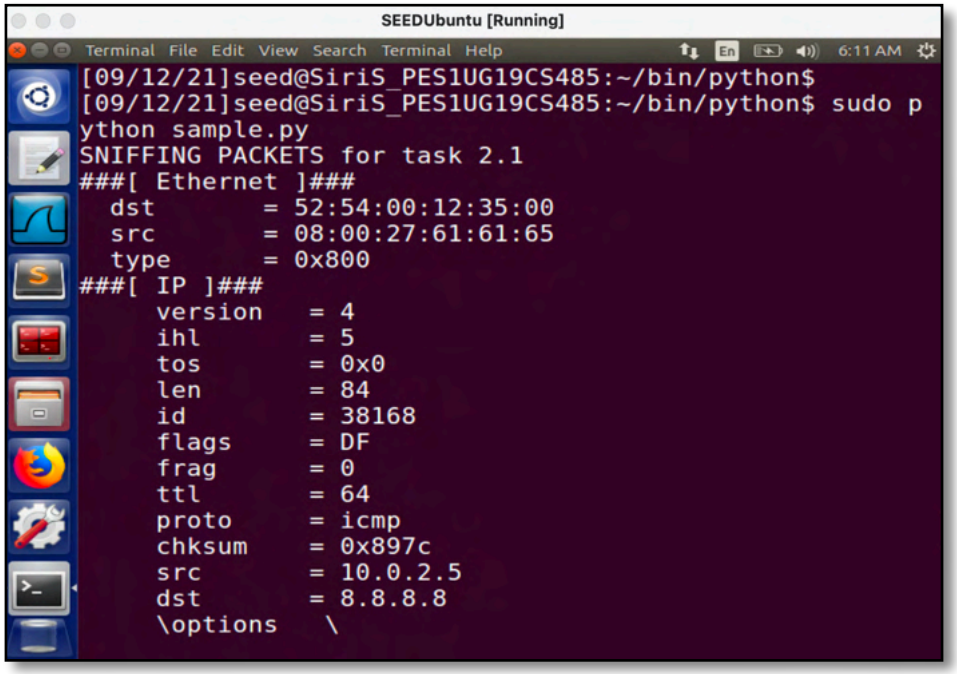
Task 1.2: Capturing ICMP, TCP packet and Subnet

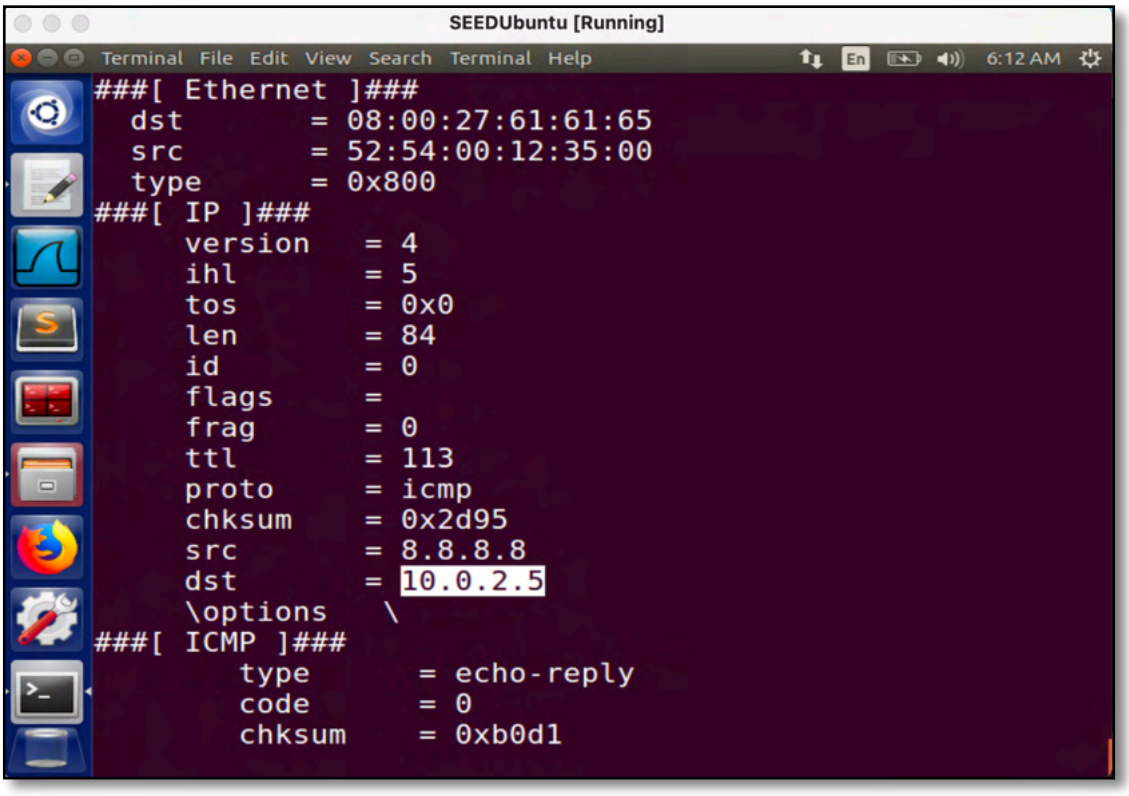


We create sniffer program using python with filter as icmp and each packet which is sniffed is printed by helper print_pkt function



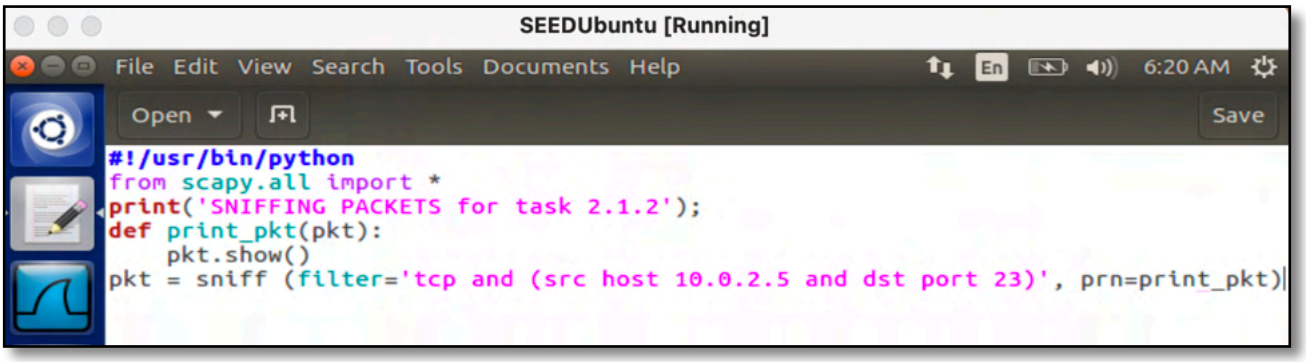
we ping 8.8.8.8 (google)on another terminal



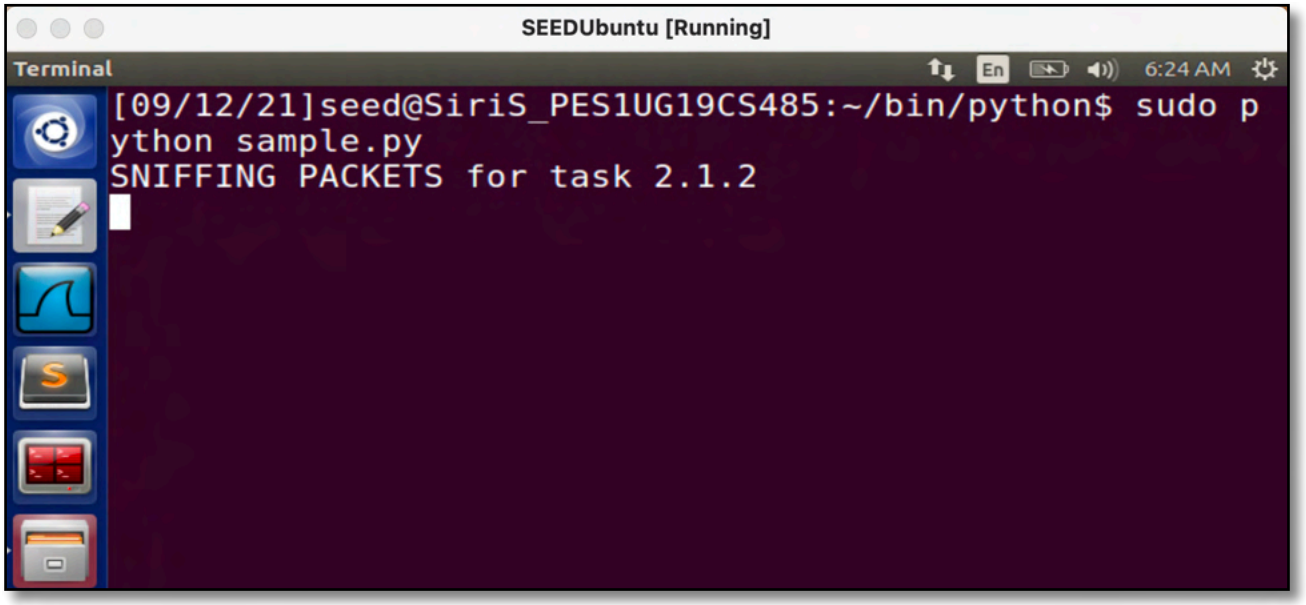


Simultaneously, we run the sample program on the attacker VM so that it can Sniff the network activity created by pinging google on the victim VM. It sniffs the packets because they are on the same network. The above screenshots gives us a detailed overview of every intercepted packet like source and destination mac, ip address of packet, checksum, ttl, the payload contents and so on.
Here dst shows as 10.0.2.5 which is the victim machine

Task 2.1.2.2: Capture any TCP packet that comes from a particular IP and with a destination port number 23

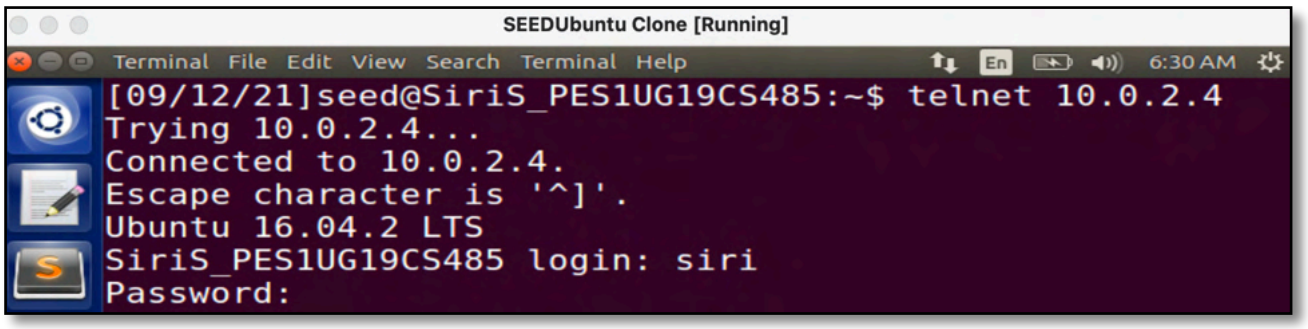


First we create the necessary sniffer program where we filter for tcp packets where source host is victim vm and dst port 23

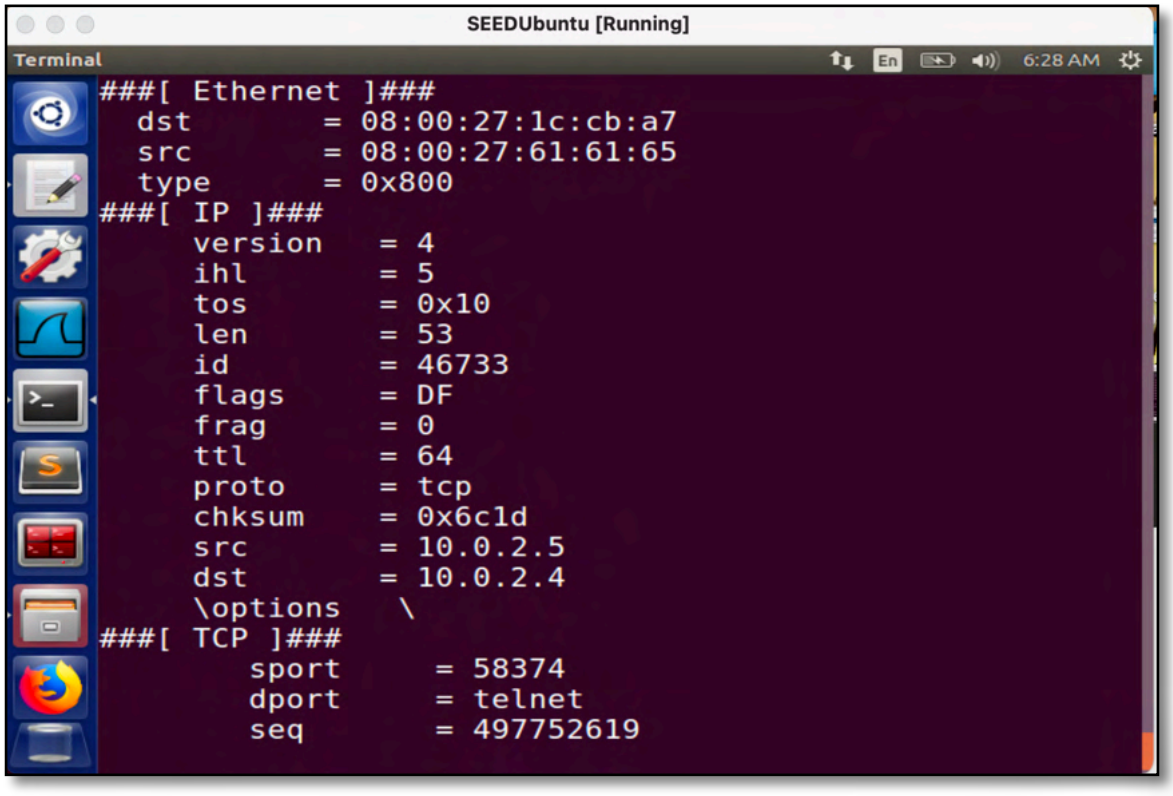


- Then we run the sniffer on the attacker vm

- No output since no telnet packets being sent. Therefore we run telnet on victim vm trying to connect to attacker vm thereby having destination port 23.

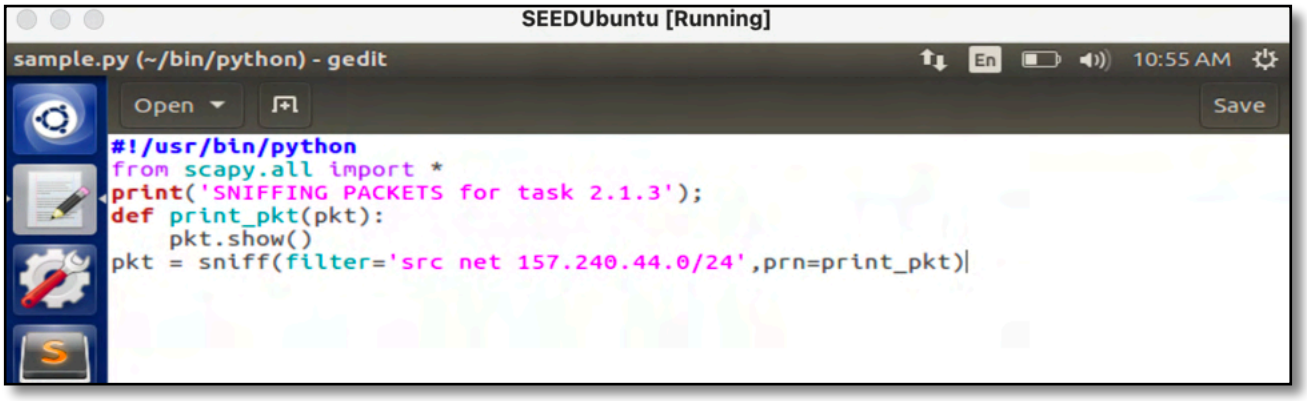


Immediately packets sent are interepted on attacker vm which



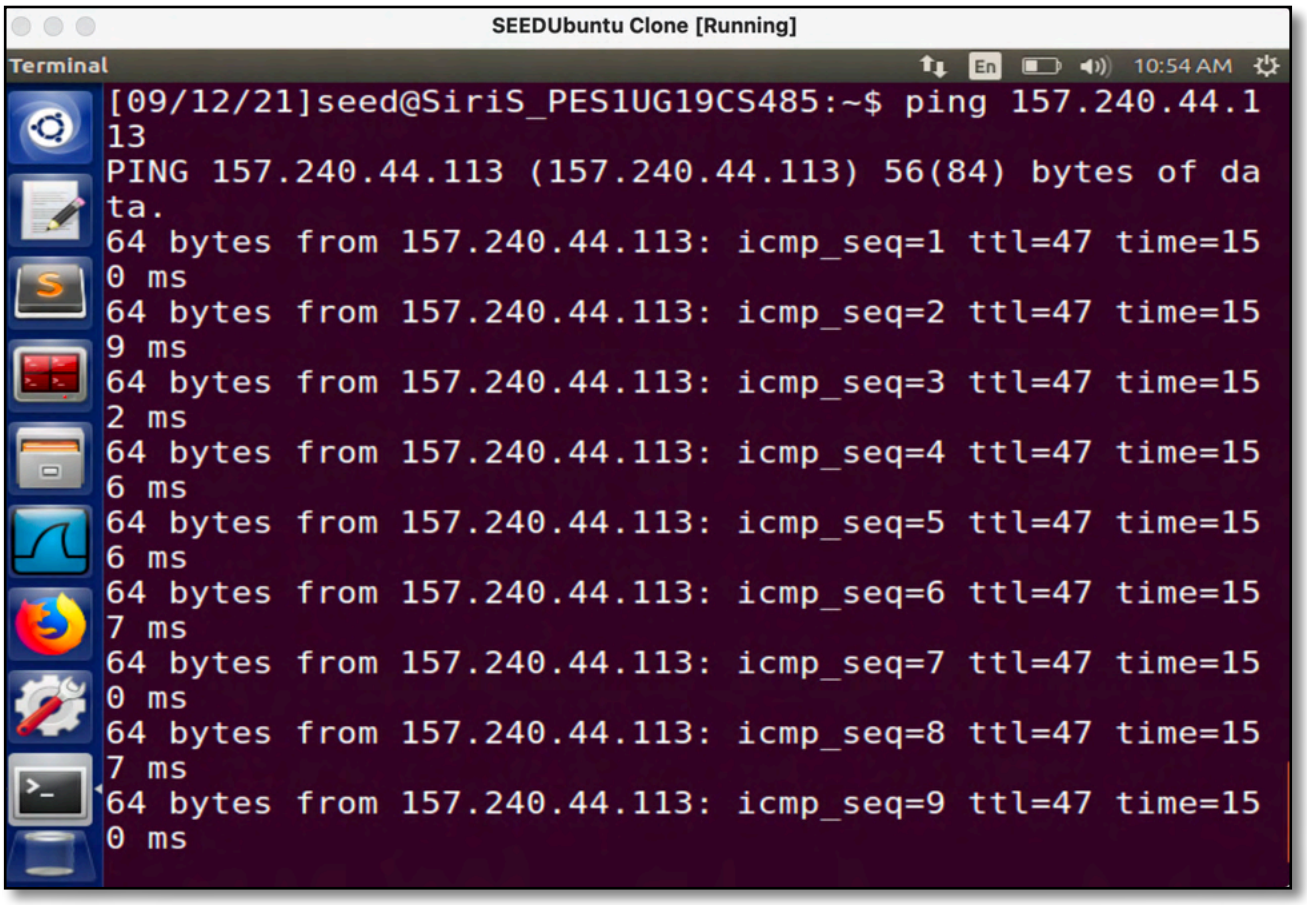
gives source port as 10.0.2.6 and dport telnet or 23.

Task 2.1.2.2.1: Capture packets comes from or to go to a particular subnet

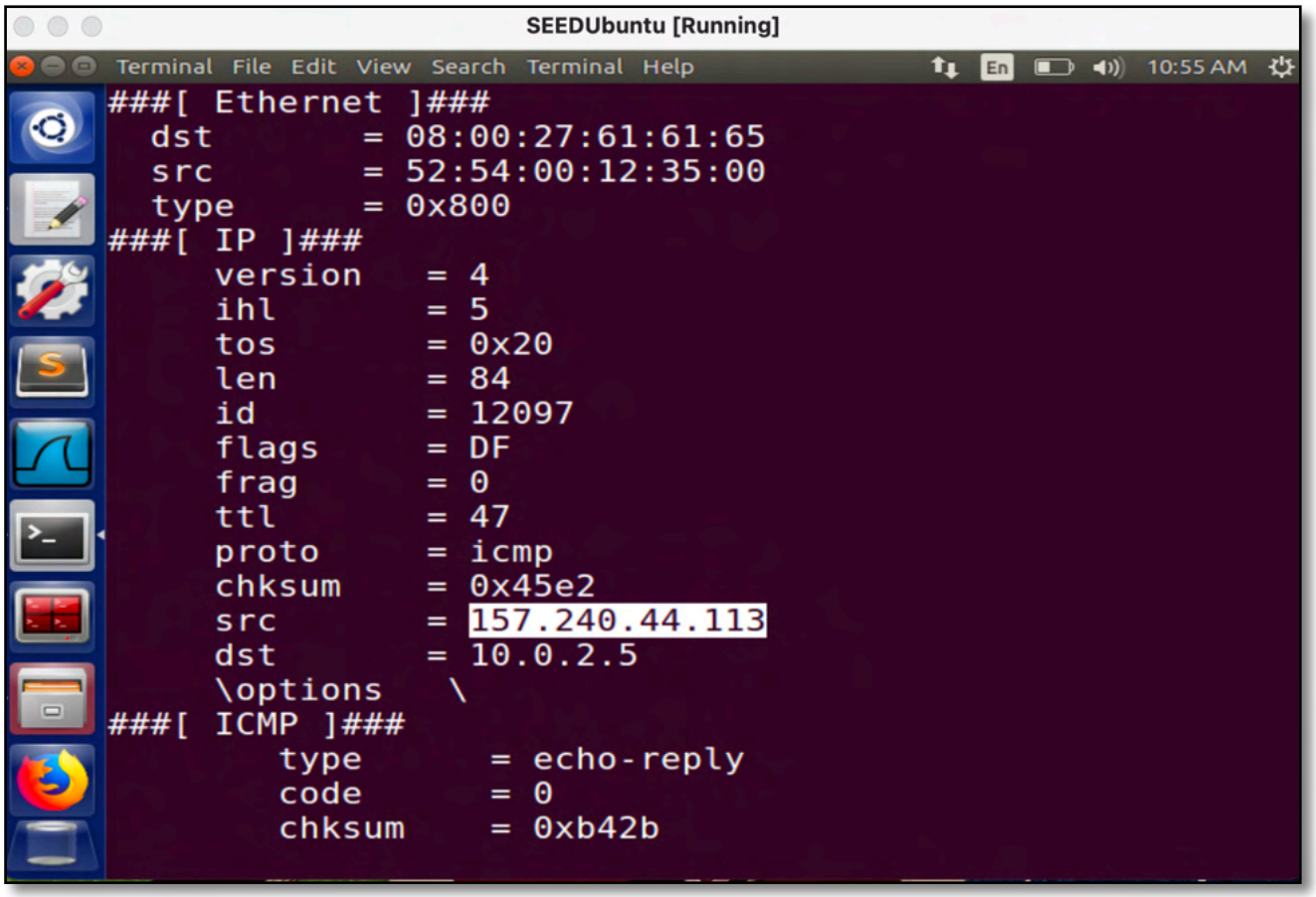


-First we create the necessary sniffer program with filter of source subnet being 157.240.44.0

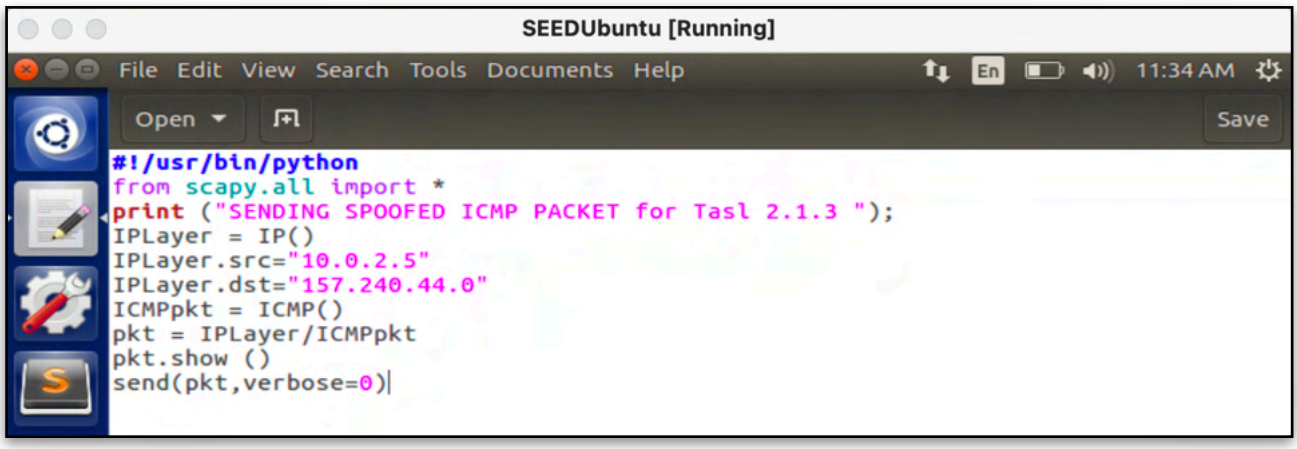
-Then we ping a device from another subnet from victim vm.



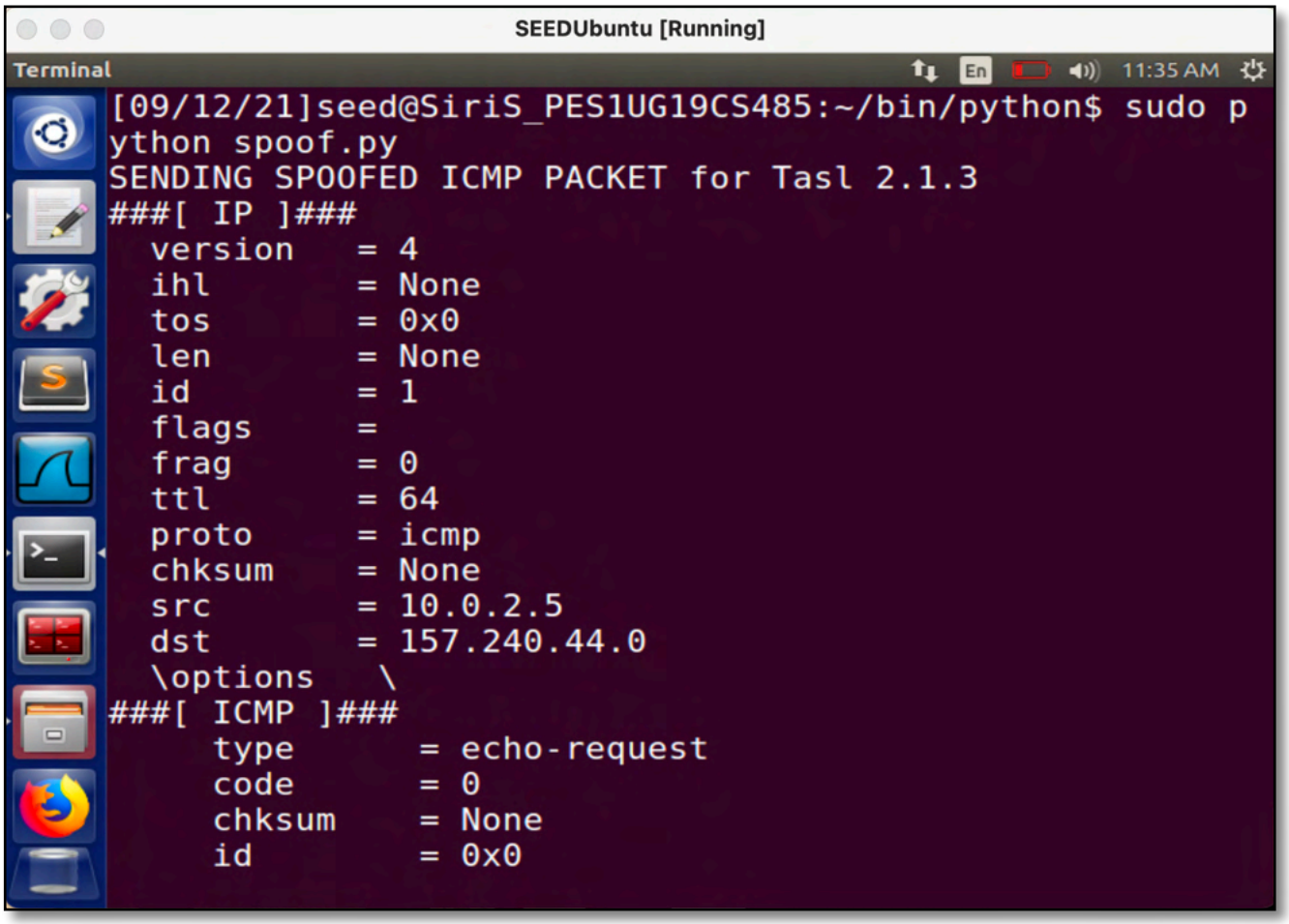
We then run the sniffer program on attacker vm and capture the intercepted packets which shows dst as victim vm


SEEDUbuntu [Running]
Terminal File Edit View Search Terminal Help
En
10:55 AM
###[Ethernet]###
dst = 08:00:27:61:61:65
src = 52:54:00:12:35:00
type = 0x800
###[IP]###
version = 4
ihl = 5
tos = 0x20
len = 84
id = 12097
flags = DF
frag = 0
ttl = 47
proto = icmp
chksum = 0x45e2
src = 157.240.44.113
dst = 10.0.2.5
\options \
###[ICMP]###
type = echo-reply
code = 0
chksum = 0xb42b

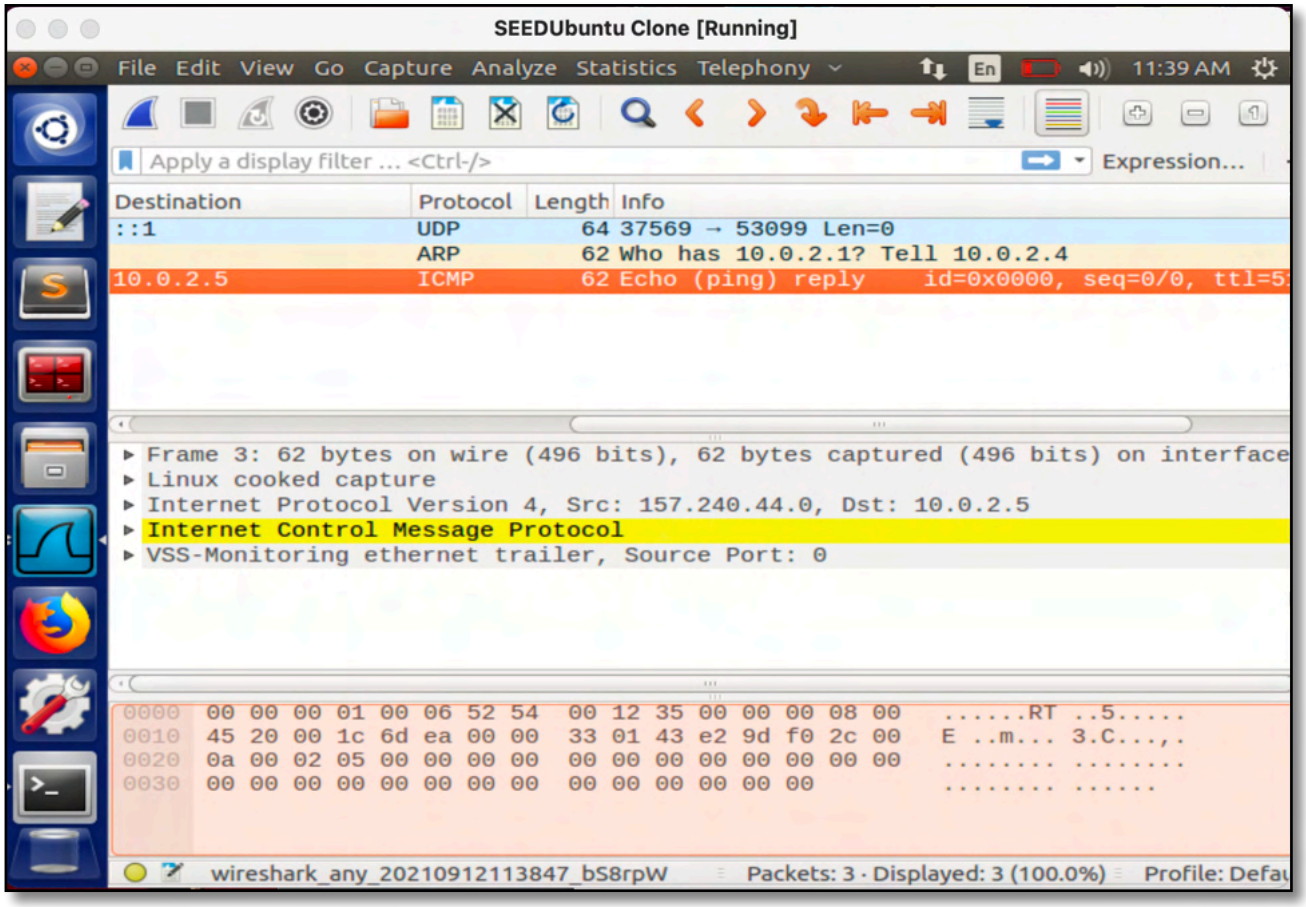
Task 3: Spoofing



First we create the necessary spoofer program where we use ip and icmp functions to create src ip as victim vm and dest ip a live machine in another subnet and create a packet out of the ip and icmp headers

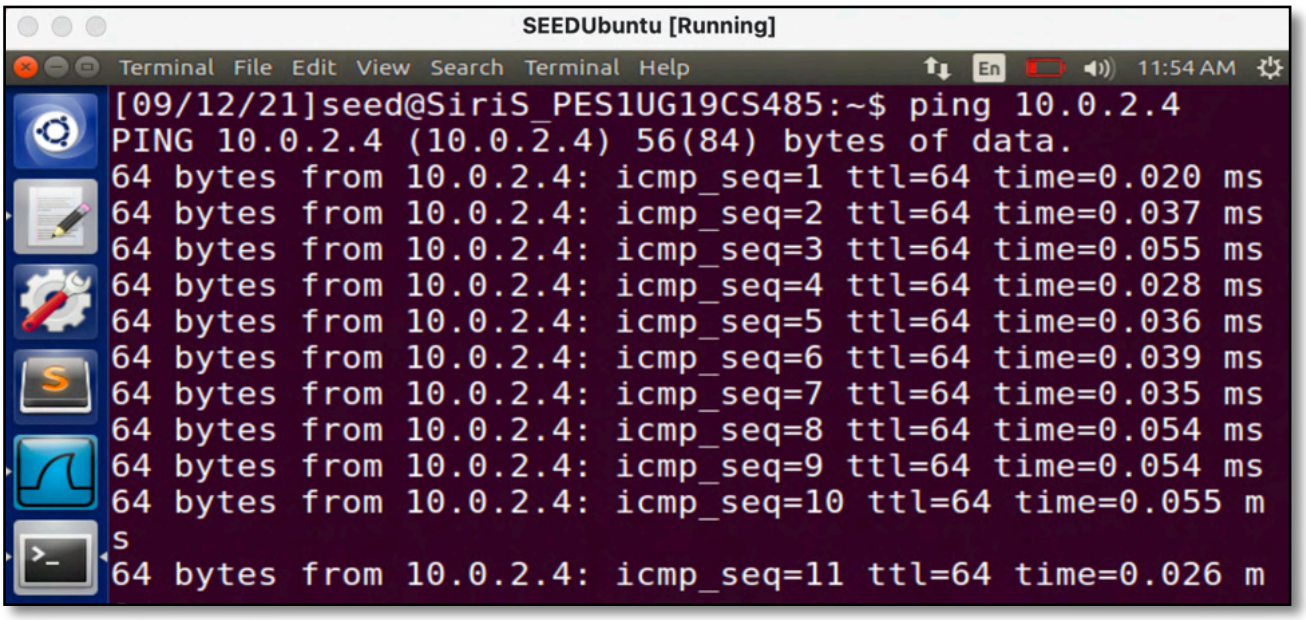


Then we run the spoofer program on attacker vm which prints the details of the packet sent.

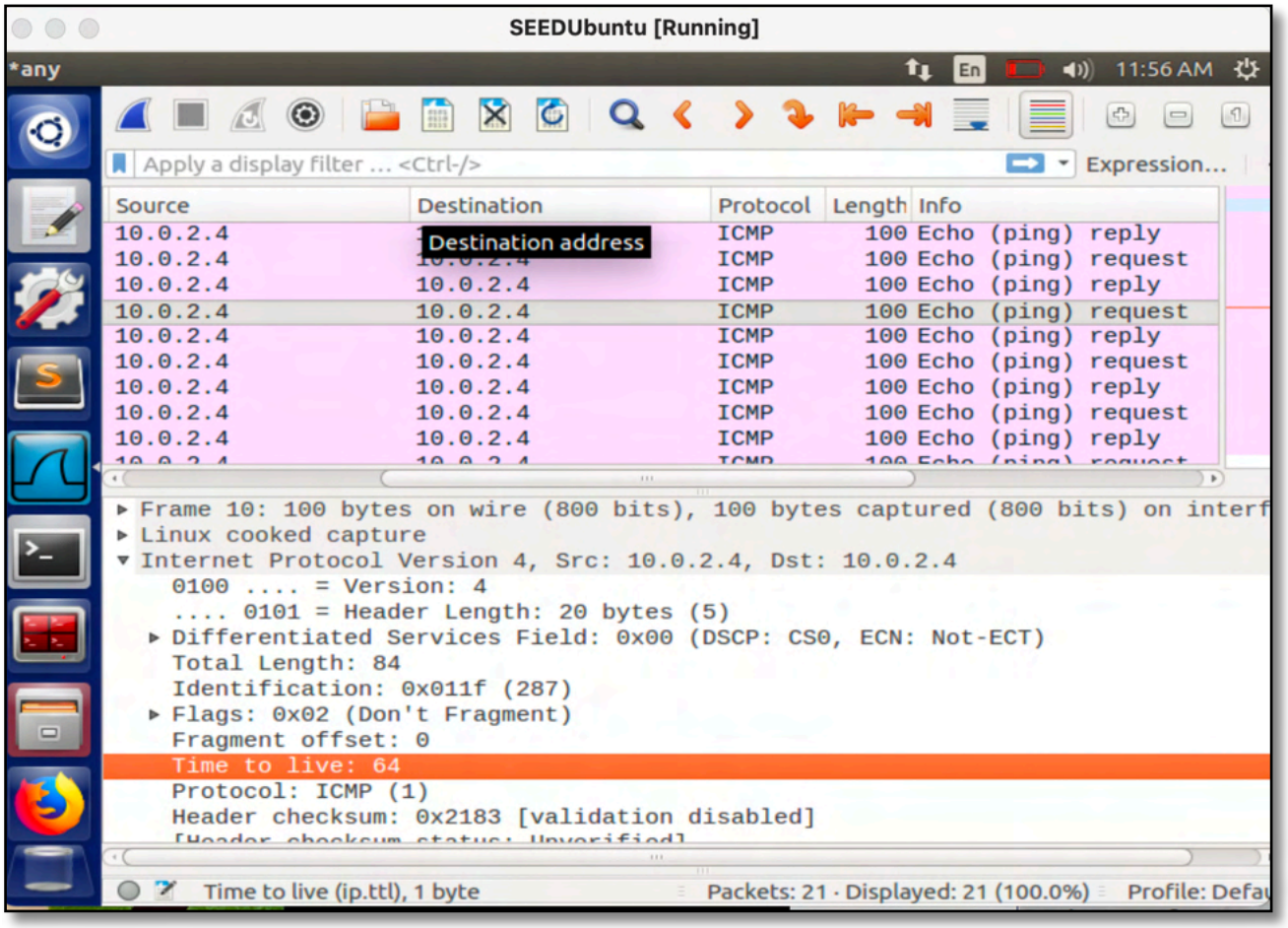


We can verify that the packet was indeed spoofed and delivered by capturing the wireshark details in victim vm

This shows that packet src ip was 10.0.2.6 when it was actually 10.0.2.5 and the other machine pings back the victim vm.



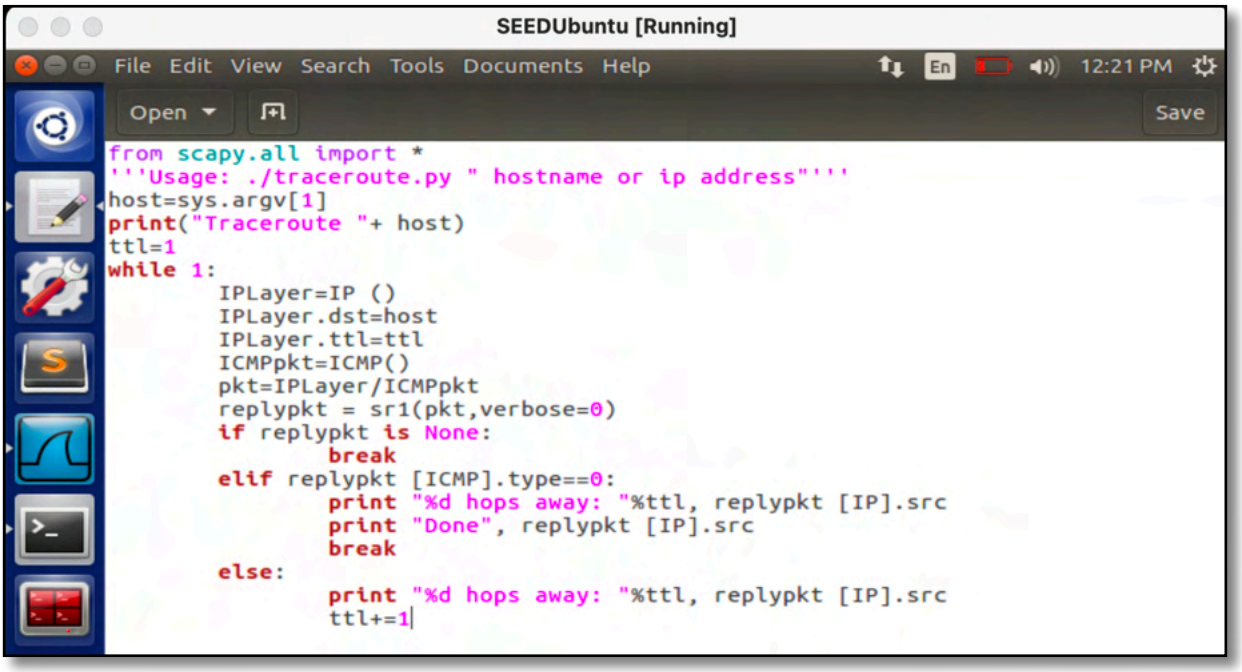
We then make attacker ping itself



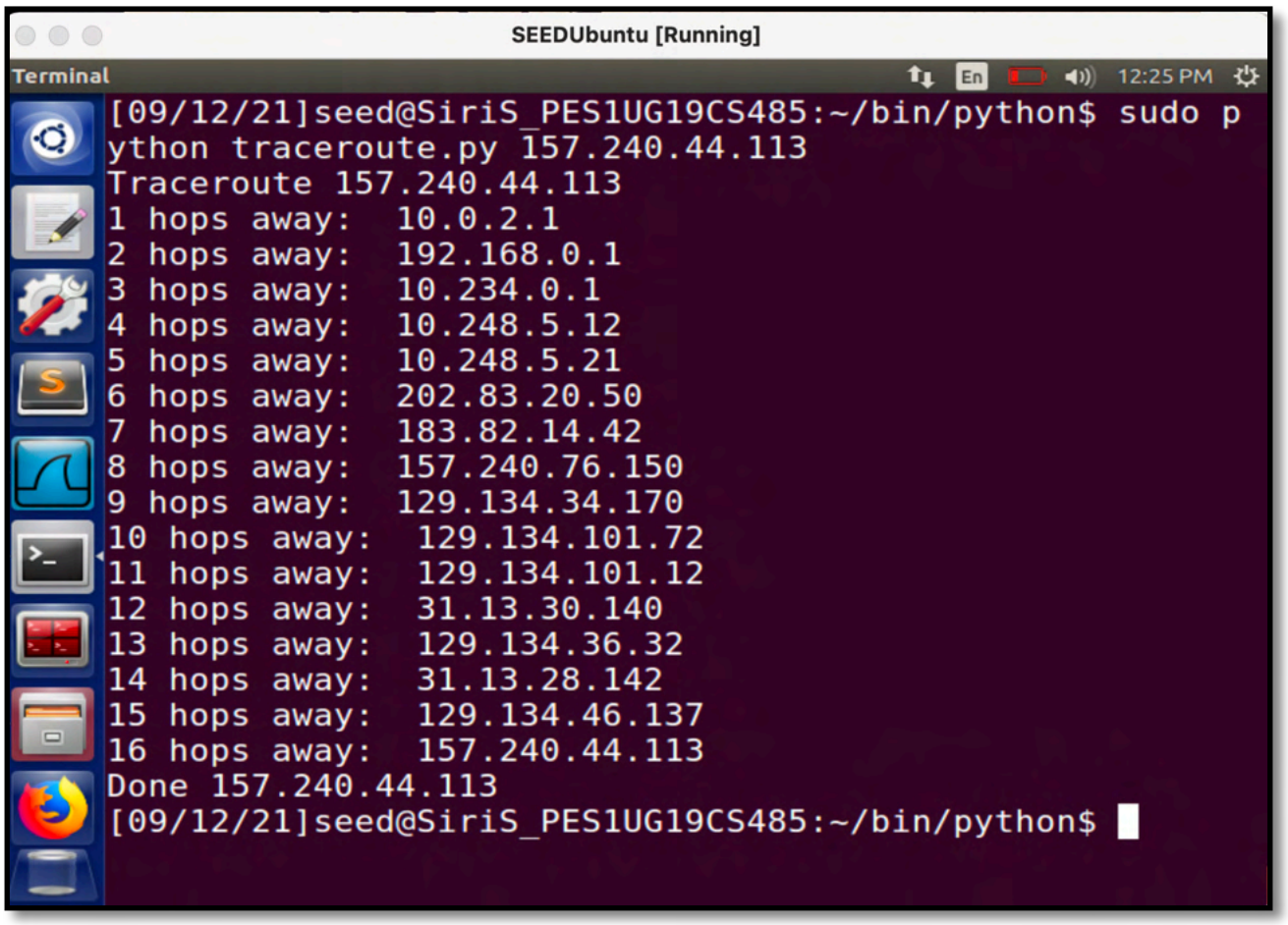
From the wireshark capture we can see that both mac addresses are 0 filled and source and des tip same with replies being fragmented and ttl as 64 on pinging the attacker from the VM

Task 4: Traceroute

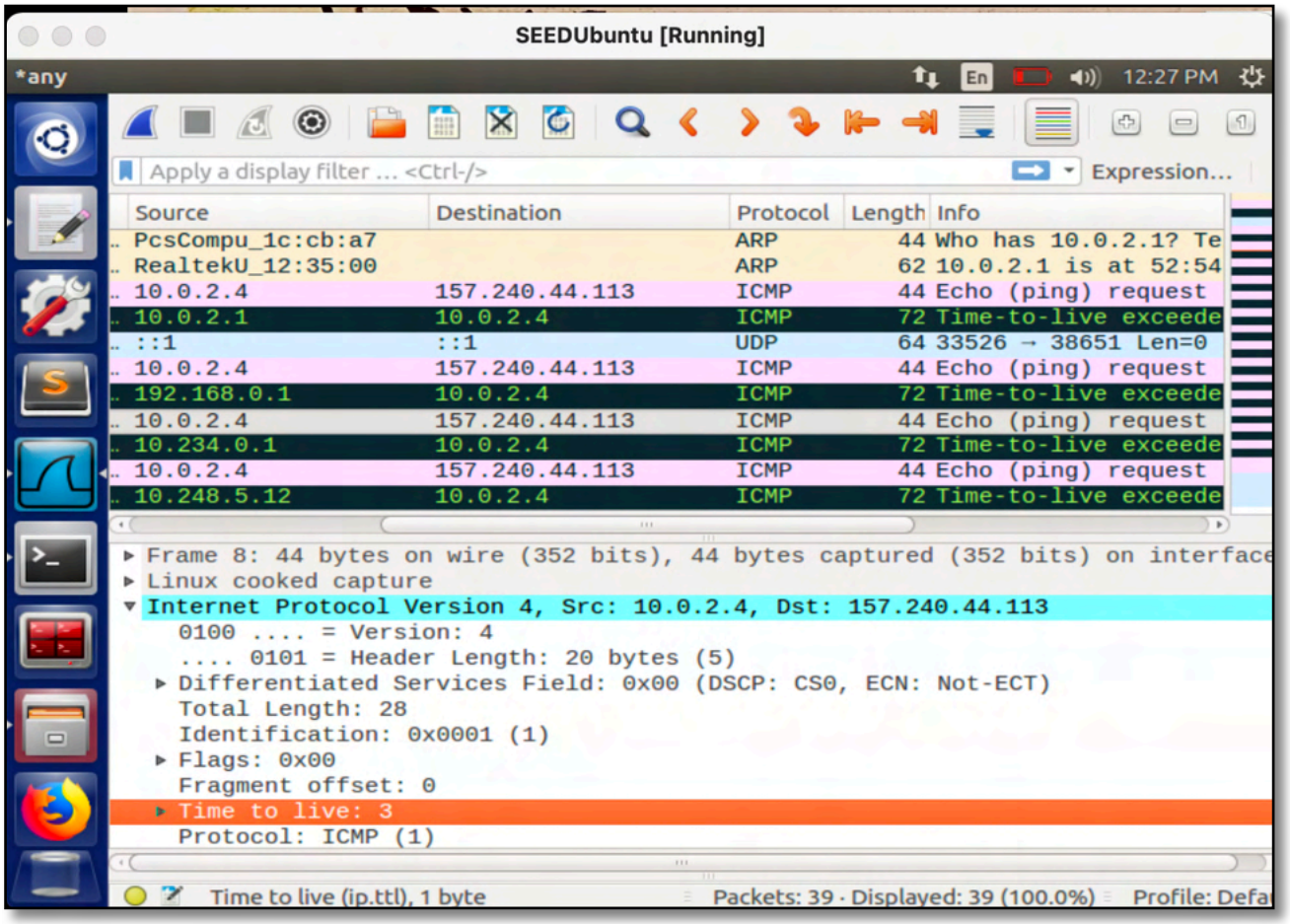
The objective of this task is to implement a simple traceroute tool using Scapy to estimate the distance, in terms of number of routers, between your VM and a selected destination. We will send a packet (any type) to the destination, with its Time-To-Live (TTL) field set to 1 first. This packet will be dropped by the first router, which will send us an ICMP error message, telling us that the TTL has exceeded. Hence, we get the IP address of the first router. We then increase our TTL field to 2, send out another packet, and get the IP address of the second router. We will repeat this procedure until our packet finally reach the destination.



We first run the traceroute program on attacker machine

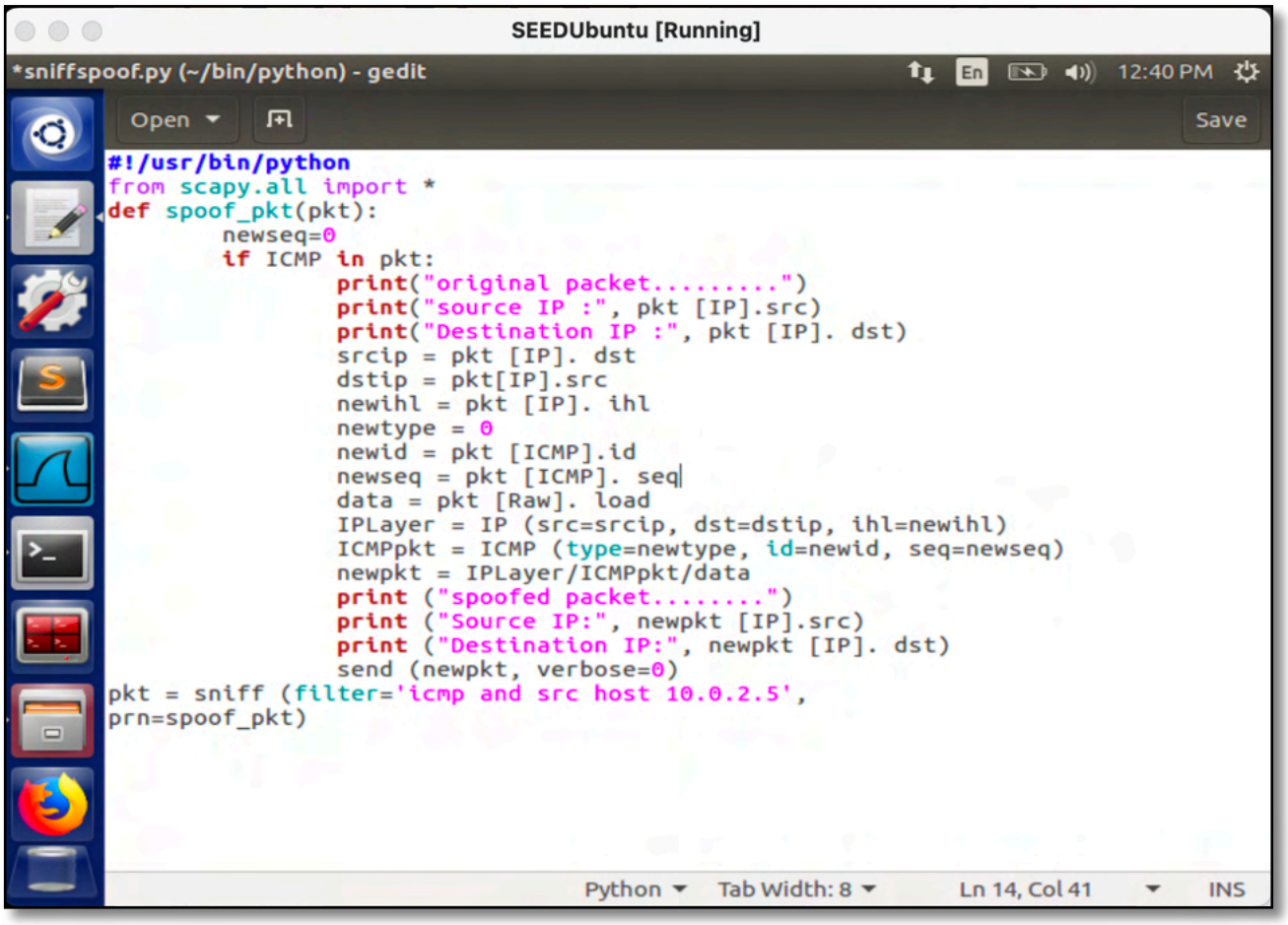


We see that the destination pc is 16 hops away or just outside the vm network.

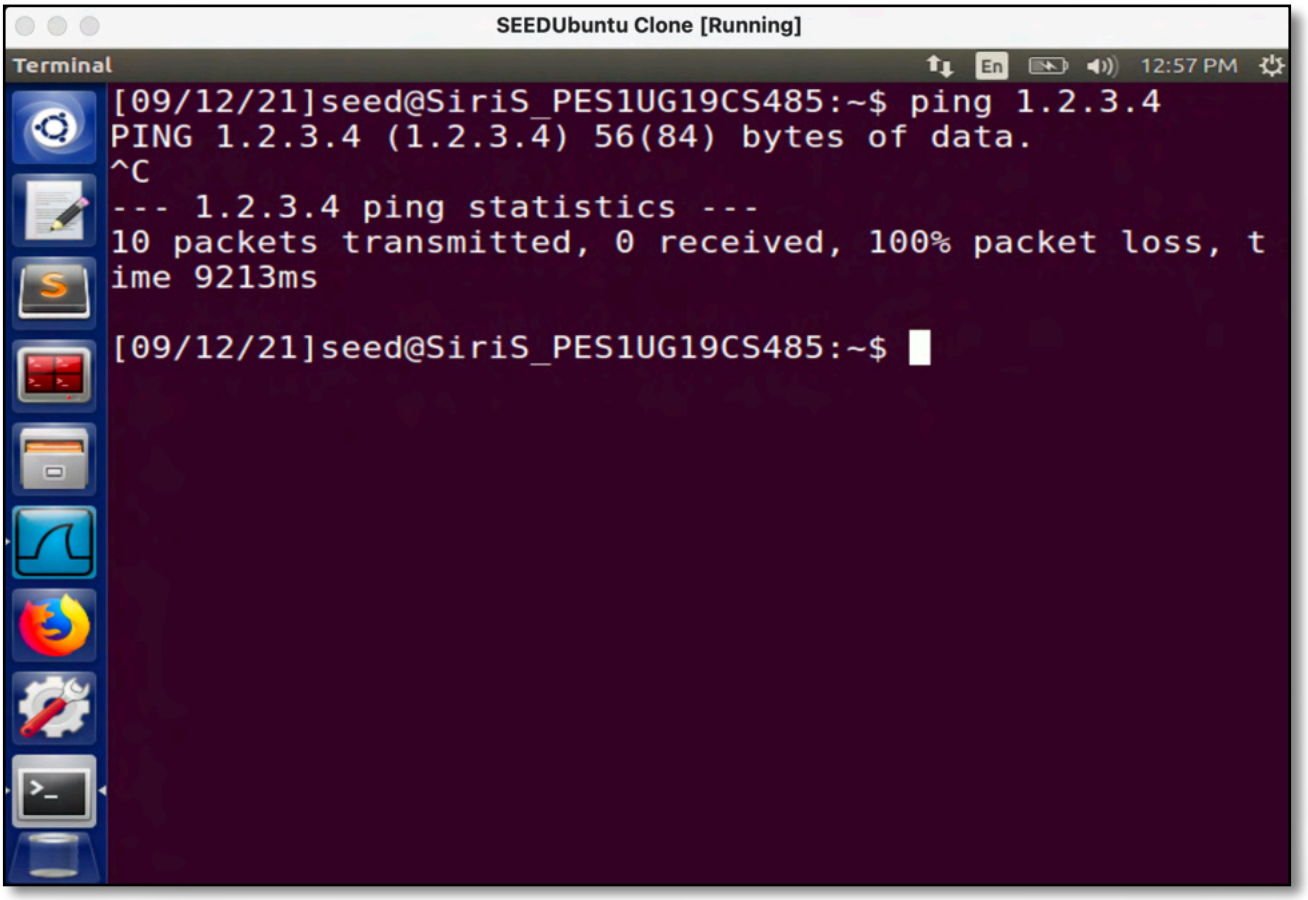


In wireshark we observe that the first packet sent has ttl value 1 and the gateway returns ttl exceeded while the second packet sent has ttl value 2 with successful reply for dest pc

TASK 5: Sniffing and-then Spoofing



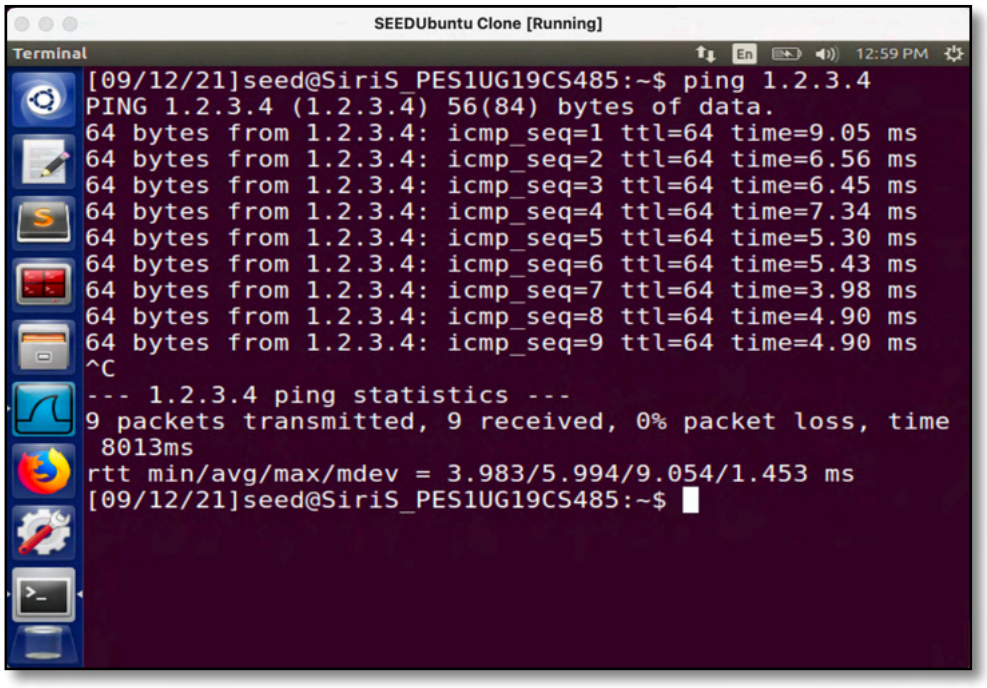
In this task, victim machine pings a non-existing IP address "1.2.3.4". As the attacker machine is on the same network, it sniffs the request packet, creates a new echo reply packet with IP and ICMP header and sends it to the victim machine. Hence, the user will always receive an echo reply from a non-existing IP address indicating that the machine is alive.



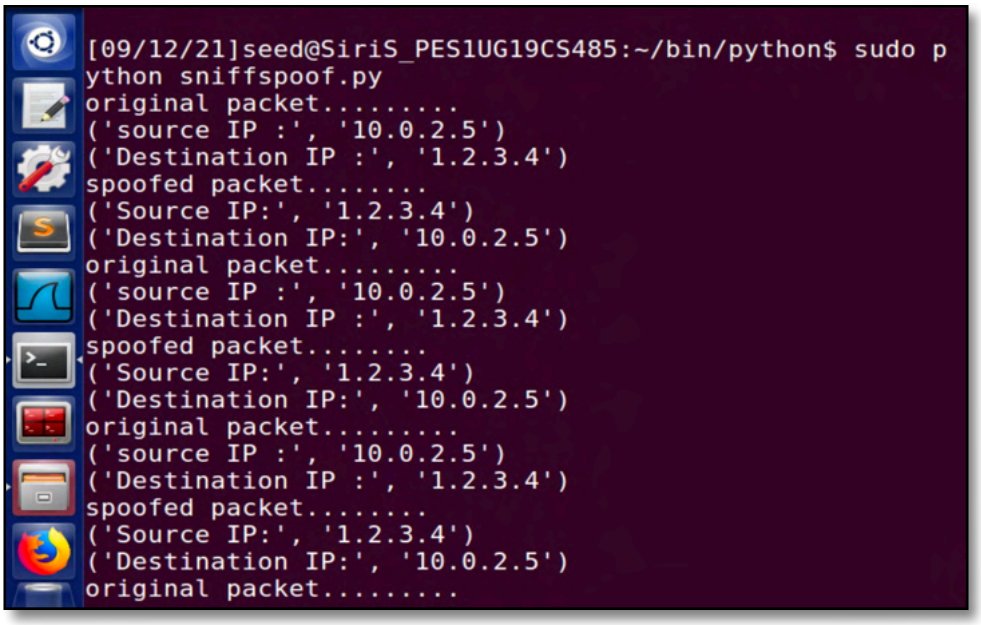
- We then ping 1.2.3.4 on victim vm

- We see that no response occurs due to it being a nonexistent ip. However we repeat the same experiment after running the sniffer spoofer program on attacker vm.

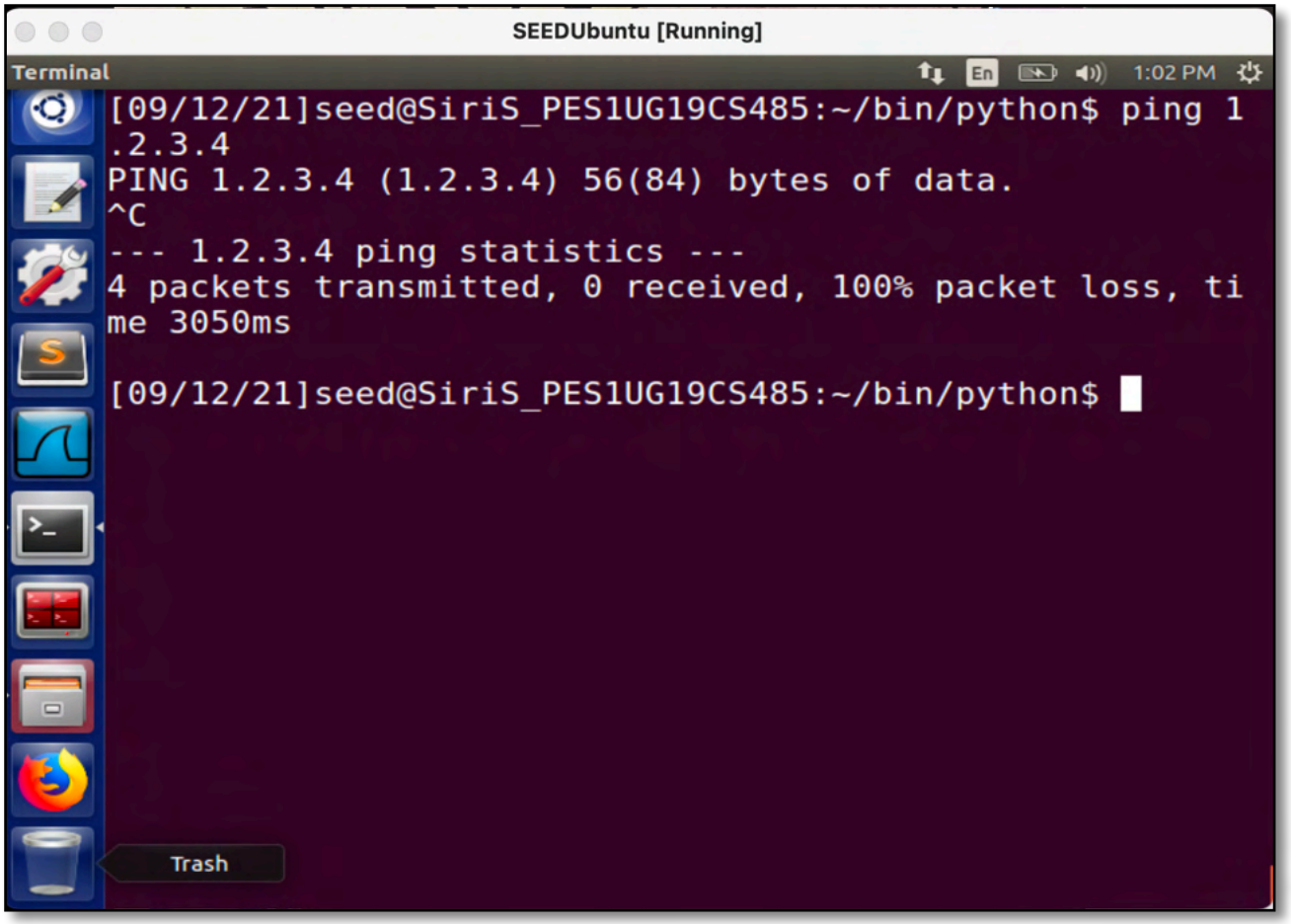
- We see that we get back replies from the ip



The attacker vm displays the source and destination details of every packet received and sent.



However on pinging the ip from attacker vm no response is seen



This is because we have put source ip filter as only victim's ip. Therefore anyone else pinging will not get any reply back.