**ACS 545 Cryptography and Network Security**

Lab 2: Packet Sniffing and Spoofing Lab

Name: Vijayagiridharan Subramanian

Task 1: Using Scapy to Sniff and Spoof Packets

Task 1.1: Sniffing Packets

1.1 A:

We are using Scapy as building blocks for sniffing and snoofing packets.

**Code:**

**This is code**



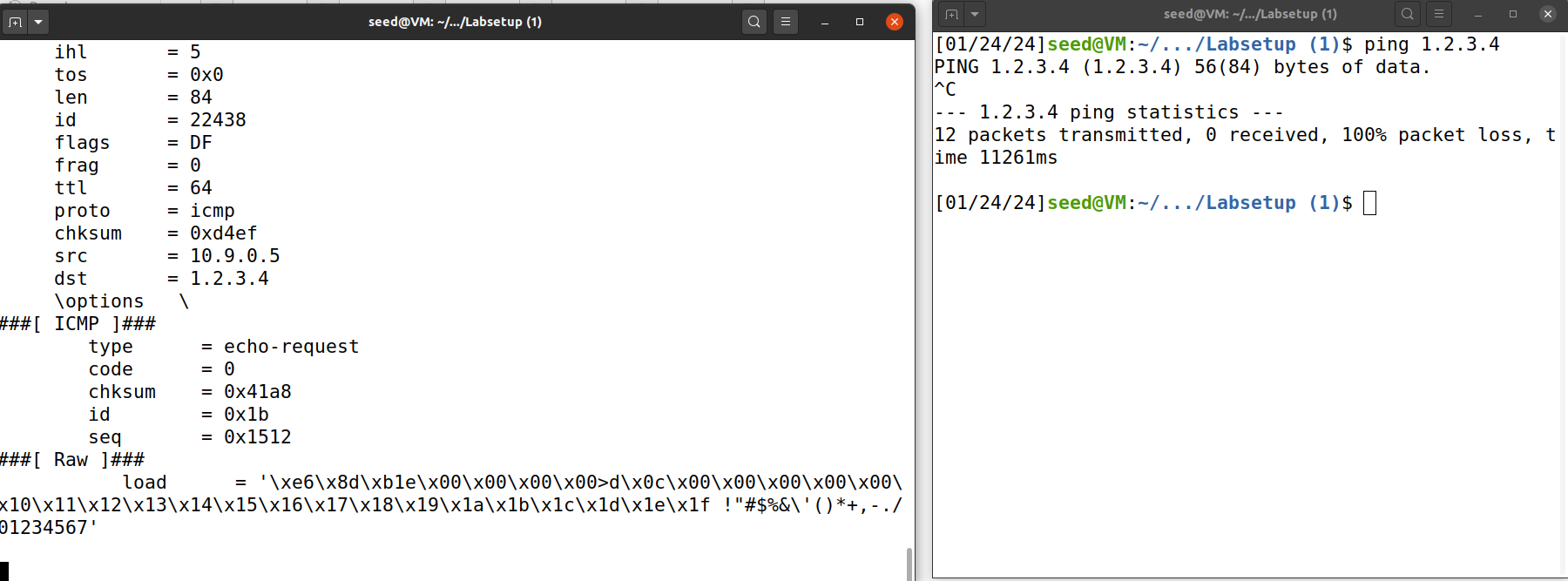
**Below operations are done in terminal in Ubuntu**

**Running with root privilege:**

We are now running this above file with sudo and we are pinging 1.2.3.4 and now we are getting output which is on left side.

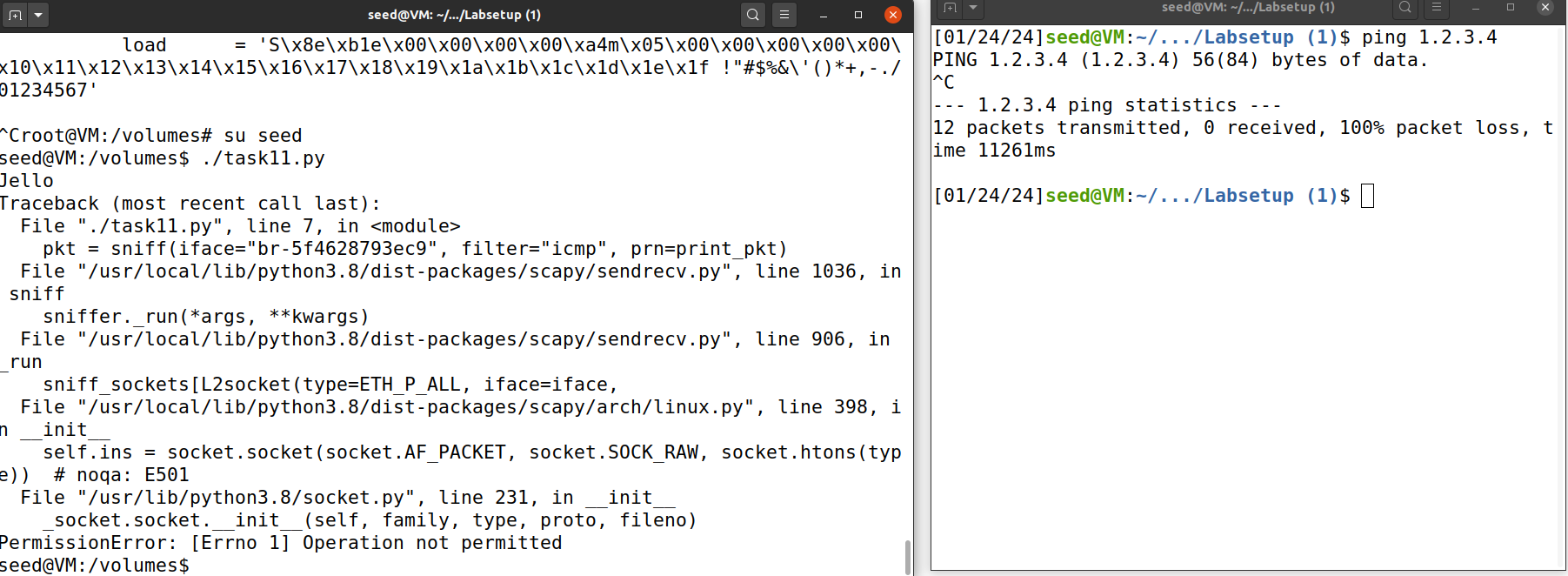
Right side shows how much packets got transmitted, received, time occurred and packet loss.

**Screenshot:**



**Running without root privilege:**

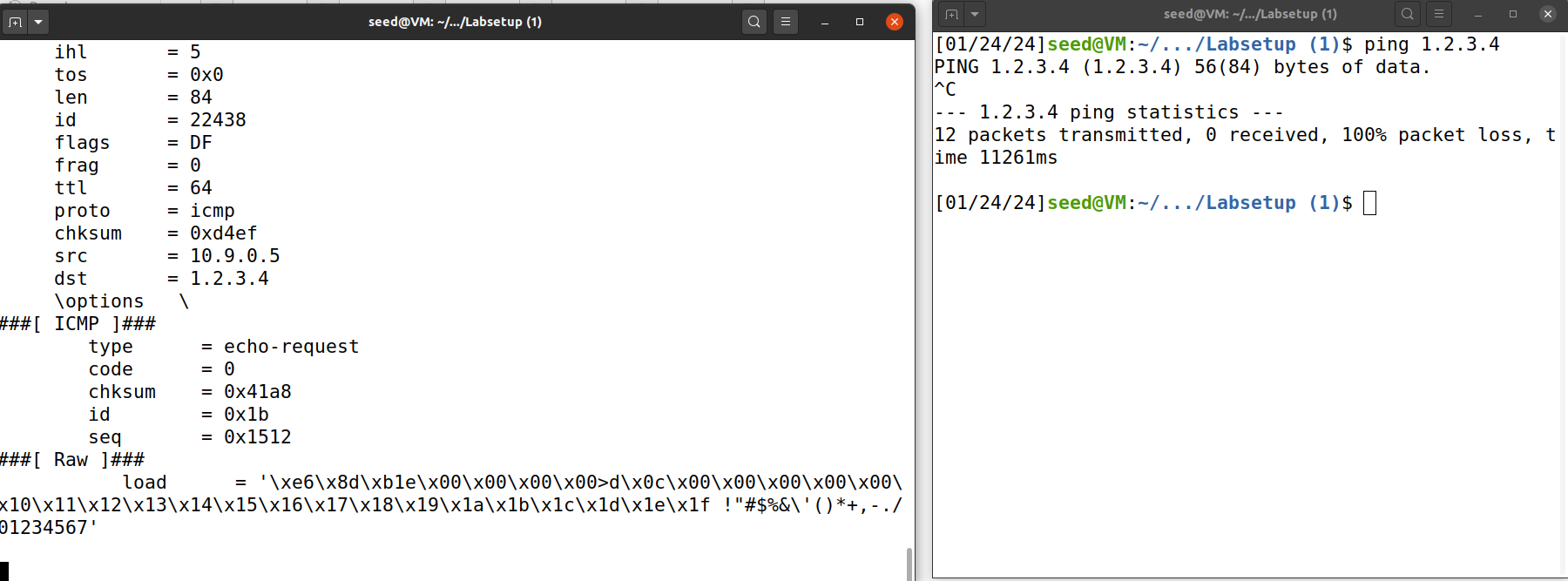
Now we are running without sudo comments , so that we are receiving Error along with “Permission Error: Operation not permitted”. This indicates scapy works only with root privileges(sudo comment).



1.1 B

* - **Capture only the ICMP packet**
* We are now setting filters in sniffing.
* **Below is updated code**
* 

Screen shot:



Now while running task11.py in terminal, we are pinging 1.2.3.4 in right side terminal and in left side we are executing sudo code and executing python code and now we are getting ICMP packets as outputs .Along with that we receive IP address, version , length and raw and load

Here only ICMP packets are captured.

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

* We are doing a similar process for TCP packets that come from a particular IP and with destination port number 23.

Here TCP packets from port 23 are received.

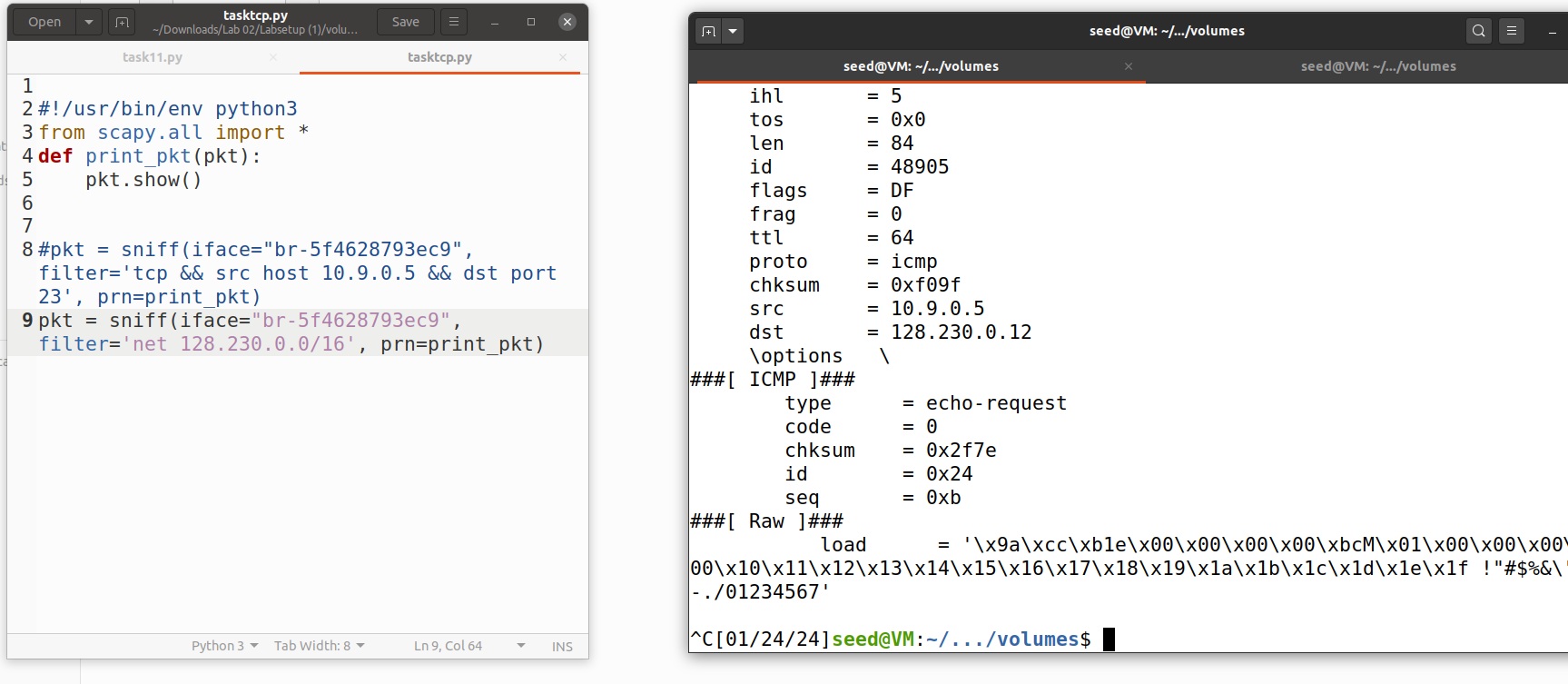
Below the left side is code modified for TCP packets and the right side is receiving output which includes Ethernet, IP entails, TCP details like source and destination port.

* **- Capture packets comes 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.**

We have to capture packets coming from or to a particular subnet. We pick a subnet 128.230.0.0/16.

Below the left side is python code and the right side is output received.

We pinged 128.230.0.12 and we are receiving it in our output below.

****

**Task 1.2:Spoofing ICMP Packets:**

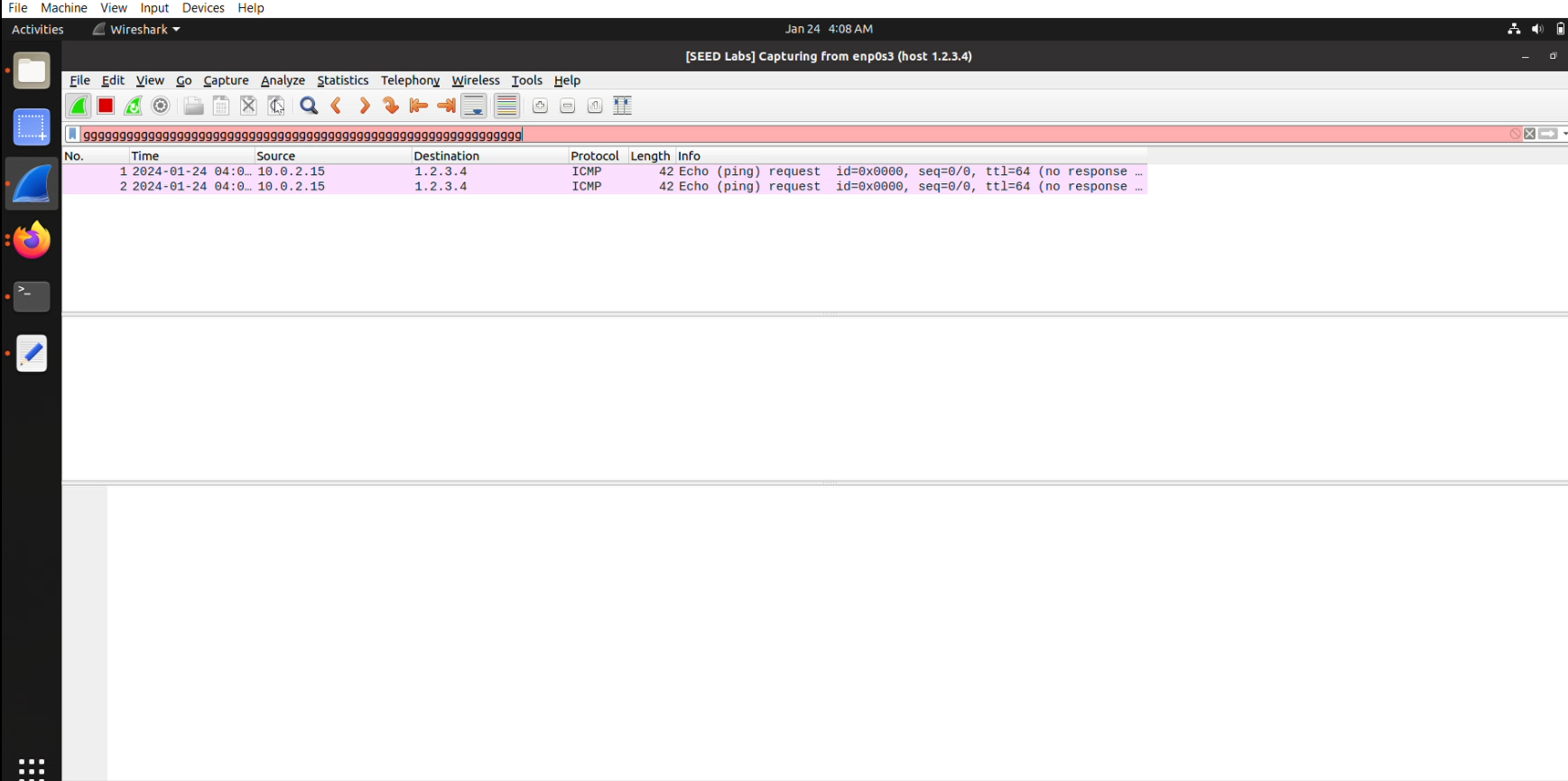
We have to spoof IP packets with an arbitrary source IP address.we have to spoof ICMP packets.

Below is the code used for spoofing ICMP packets:



**Screenshot from Wireshark:**

Wireshark checks whether request is accepted.. Below we can find Echo(Ping) request along with the source and destination. Protocol details are also mentioned below



**Snippet from Terminal:**

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 = '10.0.2.15' (None)

dst : DestIPField = '1.2.3.4' (None)

options : PacketListField = [] ([])

.

Sent 1 packets.

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)

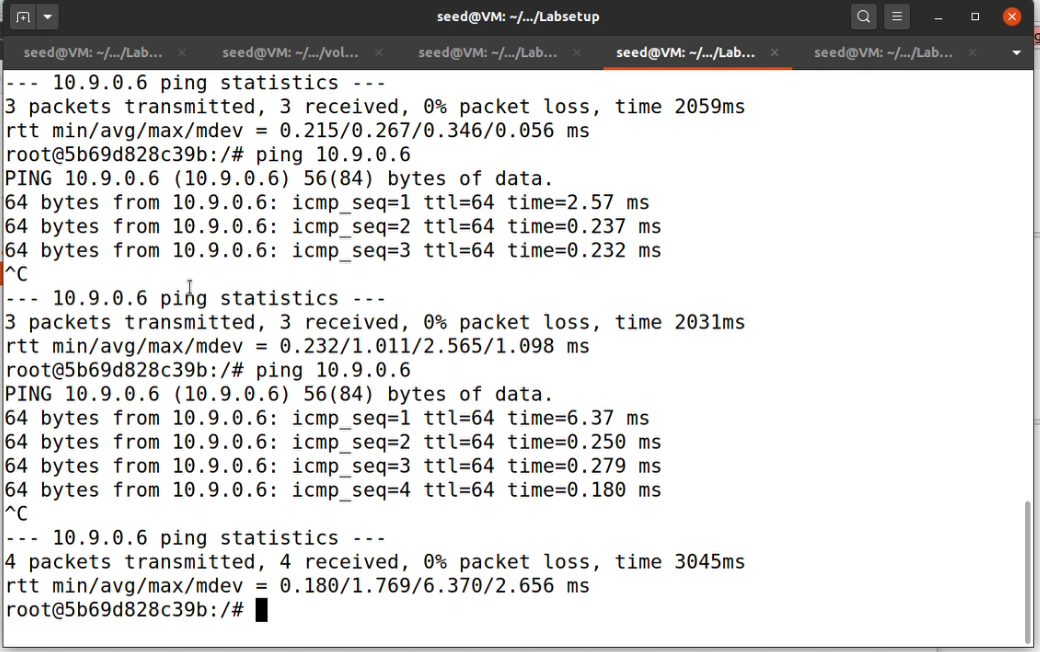
src : SourceIPField = '10.0.2.15' (None)

dst : DestIPField = '1.2.3.4' (None)

options : PacketListField = [] ([])

Above shows sent packets with their version and fields.

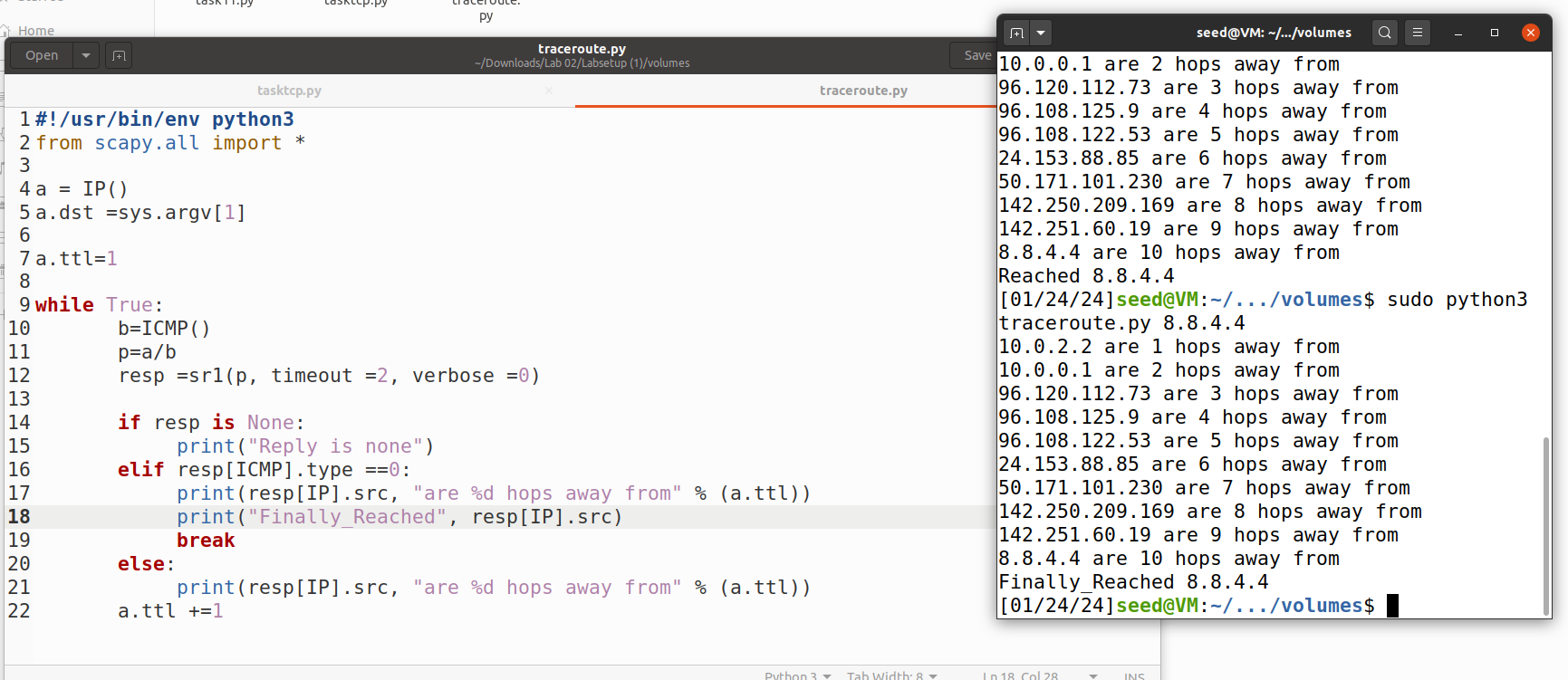
Below terminal shows packets transmitted and received for ping 10.9.0.6.



**Task 1.3:Traceroute: Traceroute. Select '8.8.4.4' as the target**

Use Scapy to calculate the distance, measured in the number of routers, from your virtual machine (VM) to a chosen destination. This functionality aligns with the traceroute tool. Our objective in this task is to create a custom tool. The approach is simple: initiate a packet transmission (of any type) to the destination, setting its Time-To-Live (TTL) field to 1 initially.

Below is image (left side showing **code for** traceroute and right side shows ping details along with details about how many hops does pinged address away from.

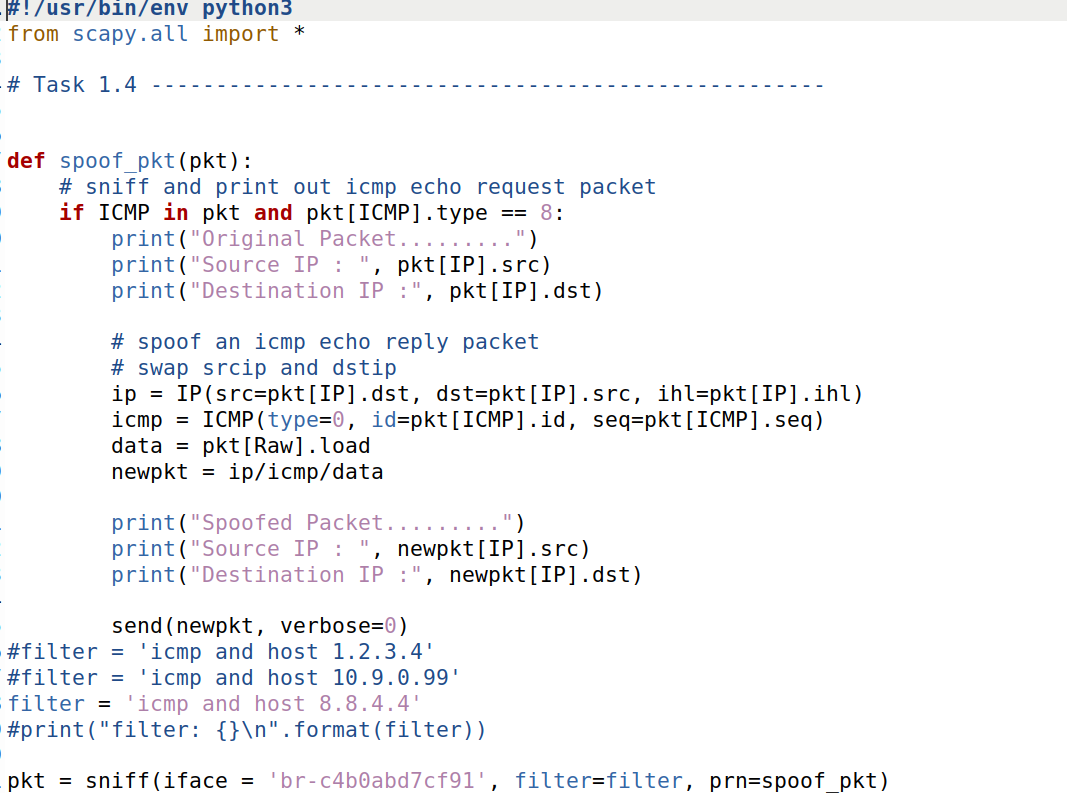


**Task 1.4: Sniffing and-then Spoofing**

**Here, w**e will integrate sniffing and spoofing techniques to create a sniff-and-then-spoof application. Two machines within the same LAN are required: the virtual machine (VM) and the user container. From the user container, initiate a ping to an IP address X, resulting in the generation of an ICMP echo request packet. If X is operational, the ping program will receive an echo reply and display the response. The sniff-and-then-spoof program, executed on the VM, actively monitors the LAN through packet sniffing. Upon detecting any ICMP echo request, regardless of the target IP address, the program promptly sends out an echo reply using packet spoofing. Consequently, irrespective of the status of machine X, the ping program will consistently receive a reply, indicating that X is operational.

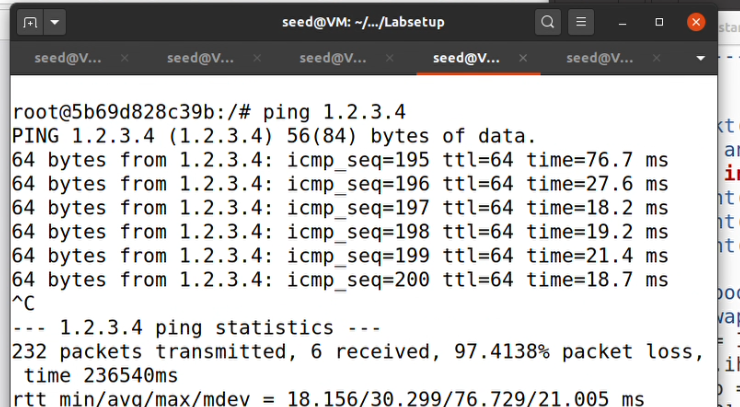
**Code USED:**

**Below is code used for sniffing and snoofing.**

****

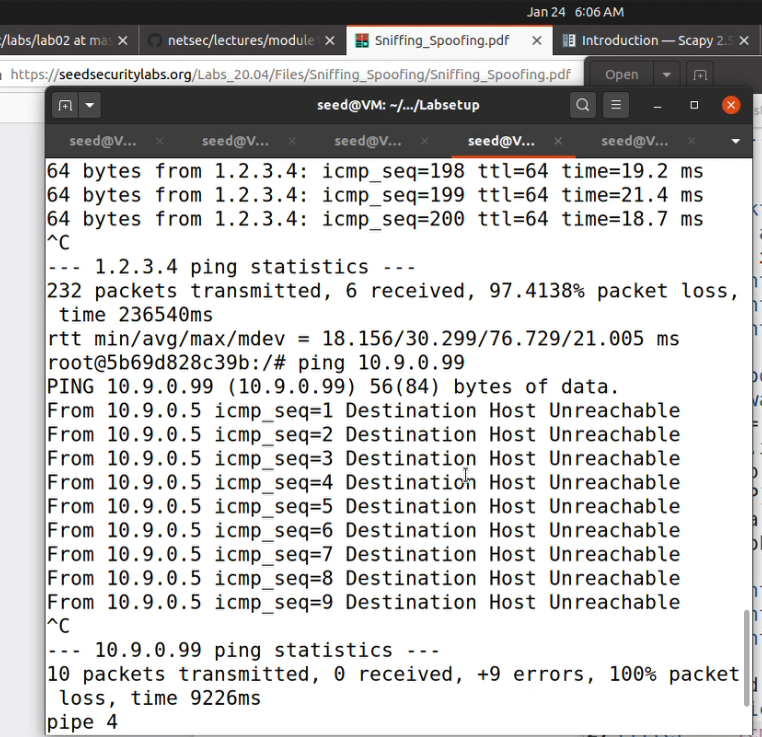
**ping 1.2.3.4 # a non-existing host on the Internet:**

Now we are pinging 1.2.3.4 which is not existent host. 232 packets ate transmitted , 6 are received.



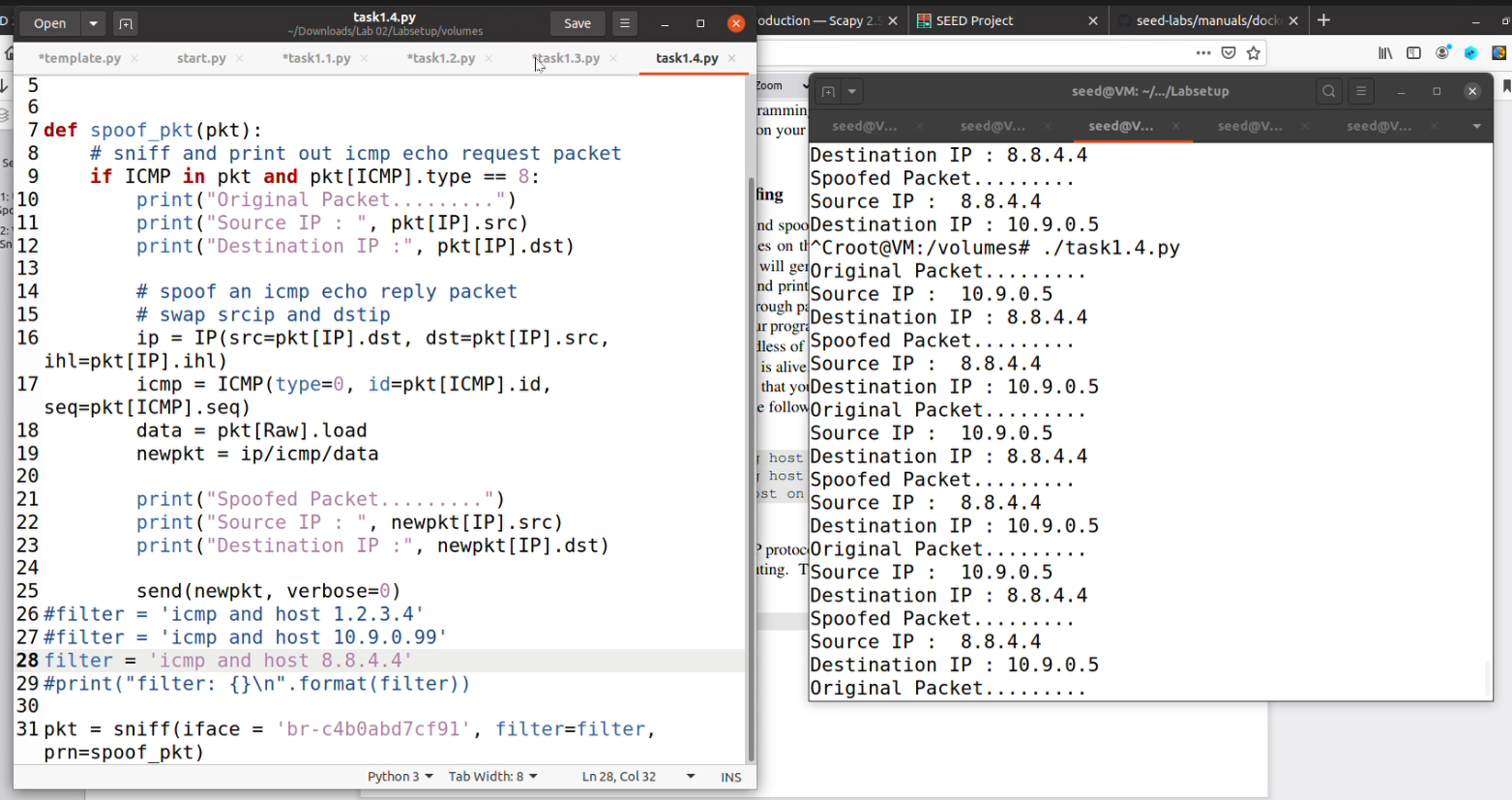
**ping 10.9.0.99 # a non-existing host on the LAN:**

Now we are pinging 10.9.0.99 , a non existing host on LAN which indicates wich icmp sequence and message that Destination Host Unreachable.



**ping 8.8.4.4 # an existing host on the Internet**

Below indicates pinging existing host 8.8.4.4 on the internet . Everything goes fine and we are receiving original and spoofed packets with source and Destination Ip.



Below indicates 5 packets transmitted, % packets received along with 5 duplicates.

