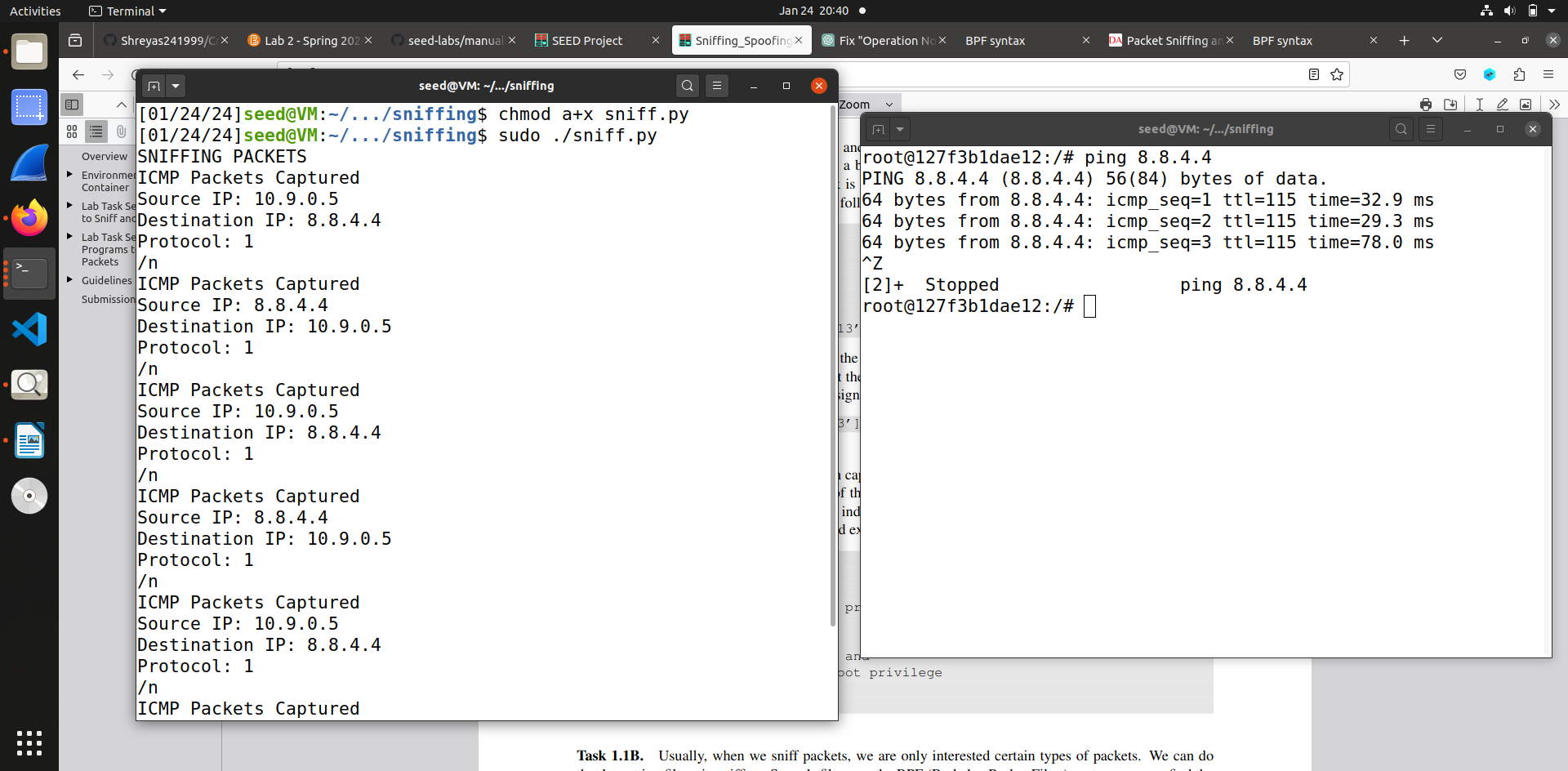
**Lab 2 Sniffing and Spoofing**

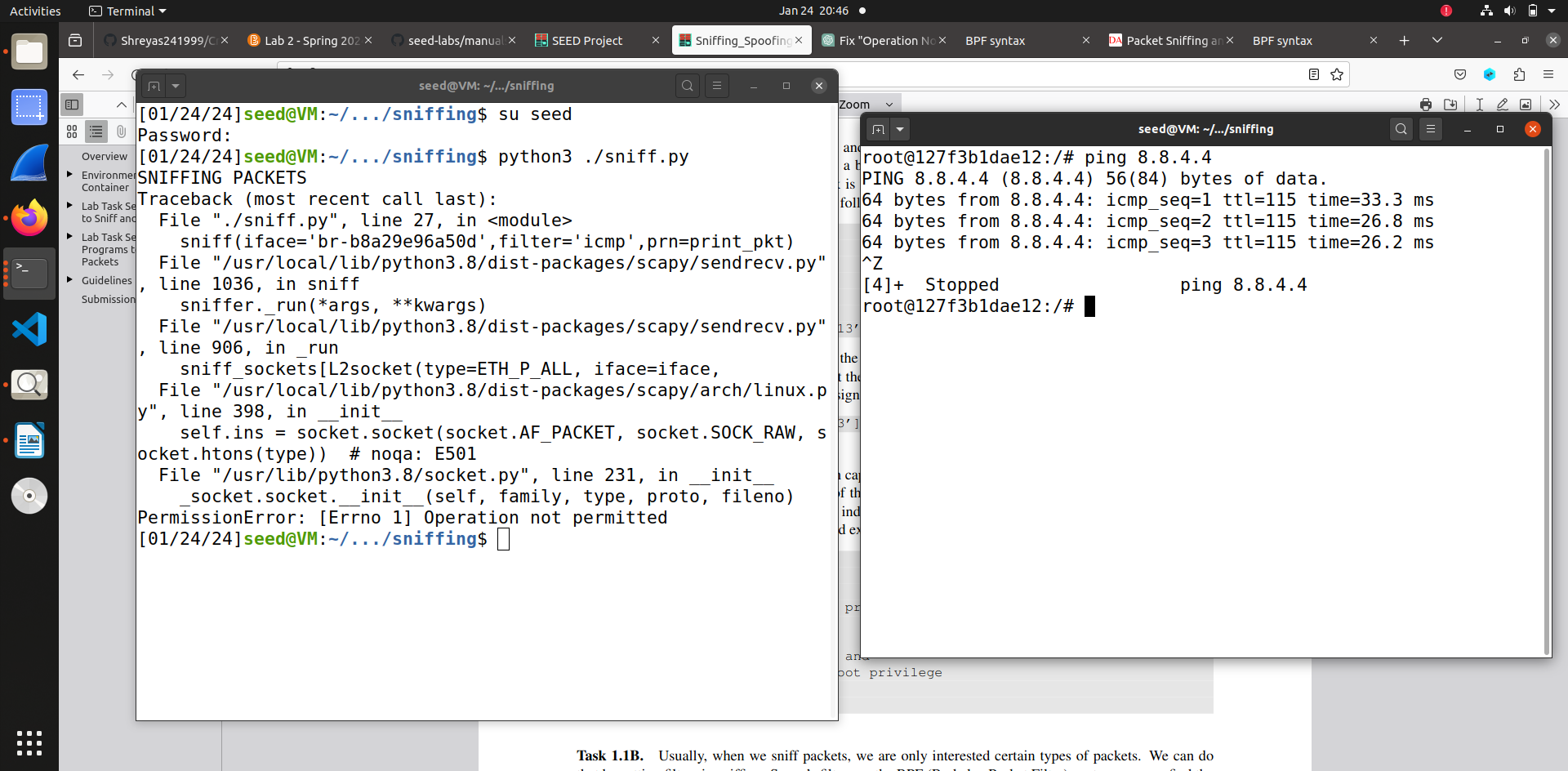
**Task 1.1 Sniffing Packets**

**Task 1.1.A**

****

In this we give executable permission to the file and using chmod a+x command and then run the program. When we run the code with sudo it allows us to see the network traffic in given interfaces.

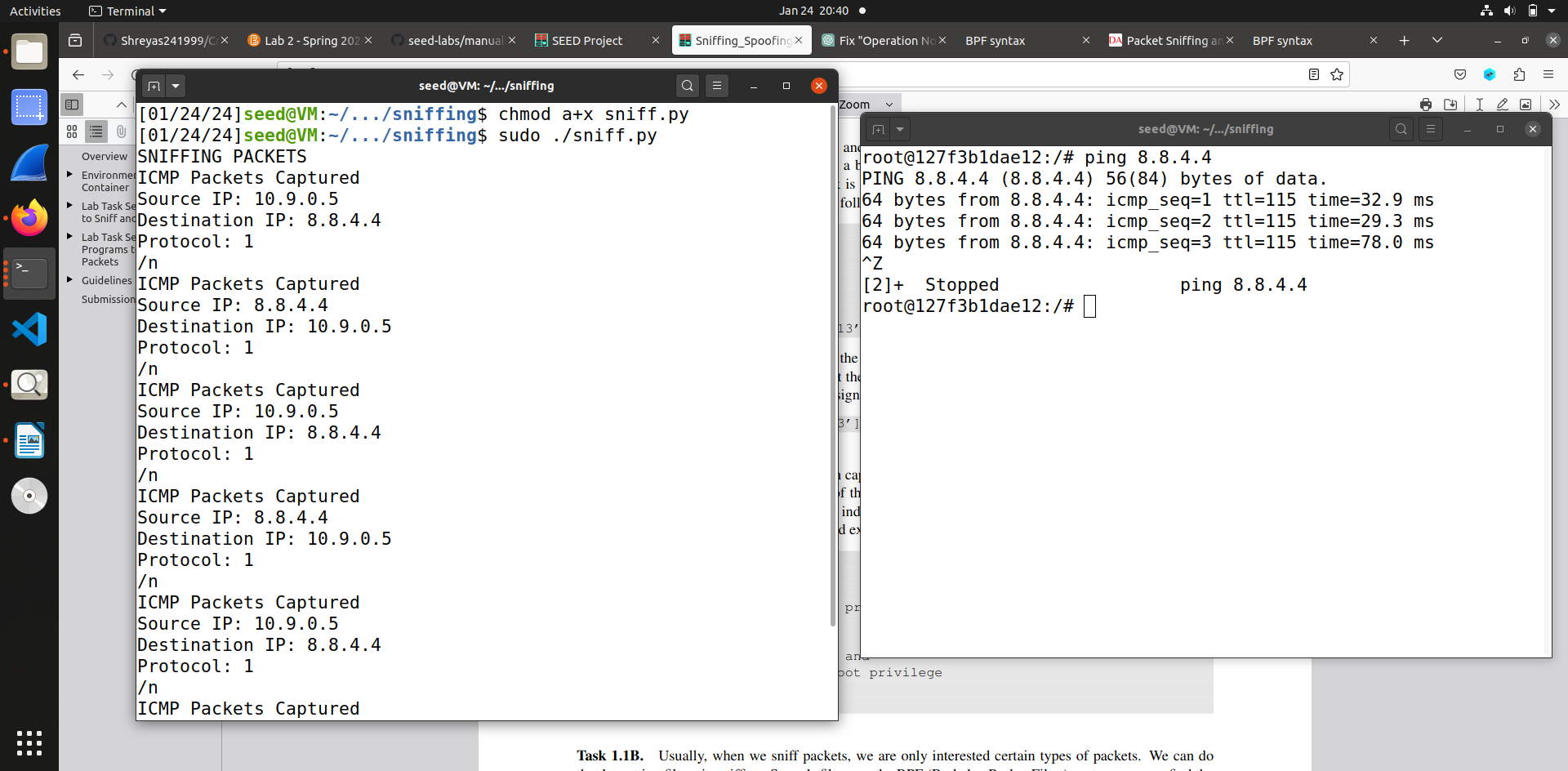
So using that we were able to run the code as sudo means superuser do.



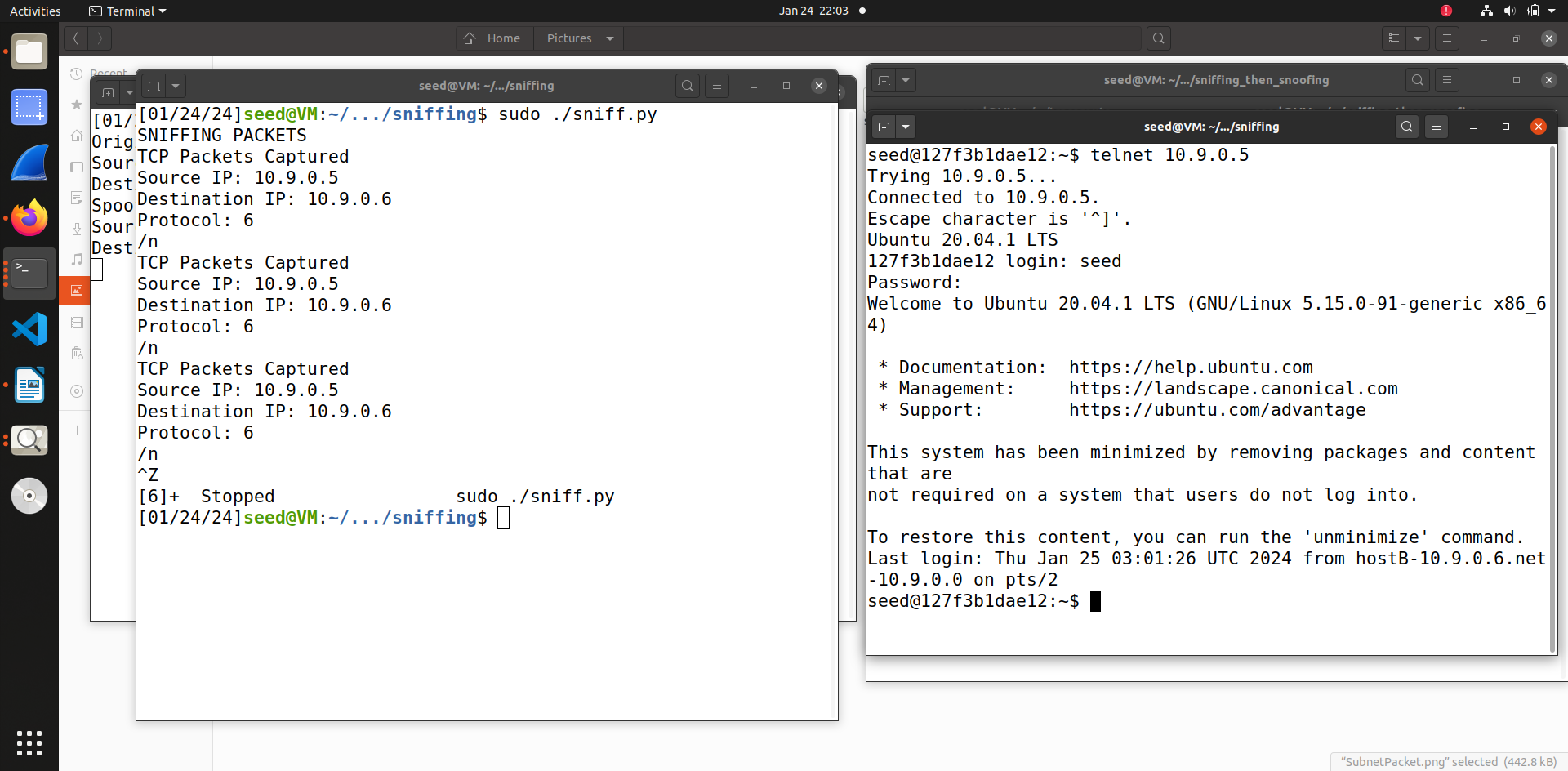
In the above we switched to seed user and then tried to execute the program without the privileges which causes error for execution showing “Operation not permitted”.

**Task 1.1.B**

**Capture ICMP Packet**

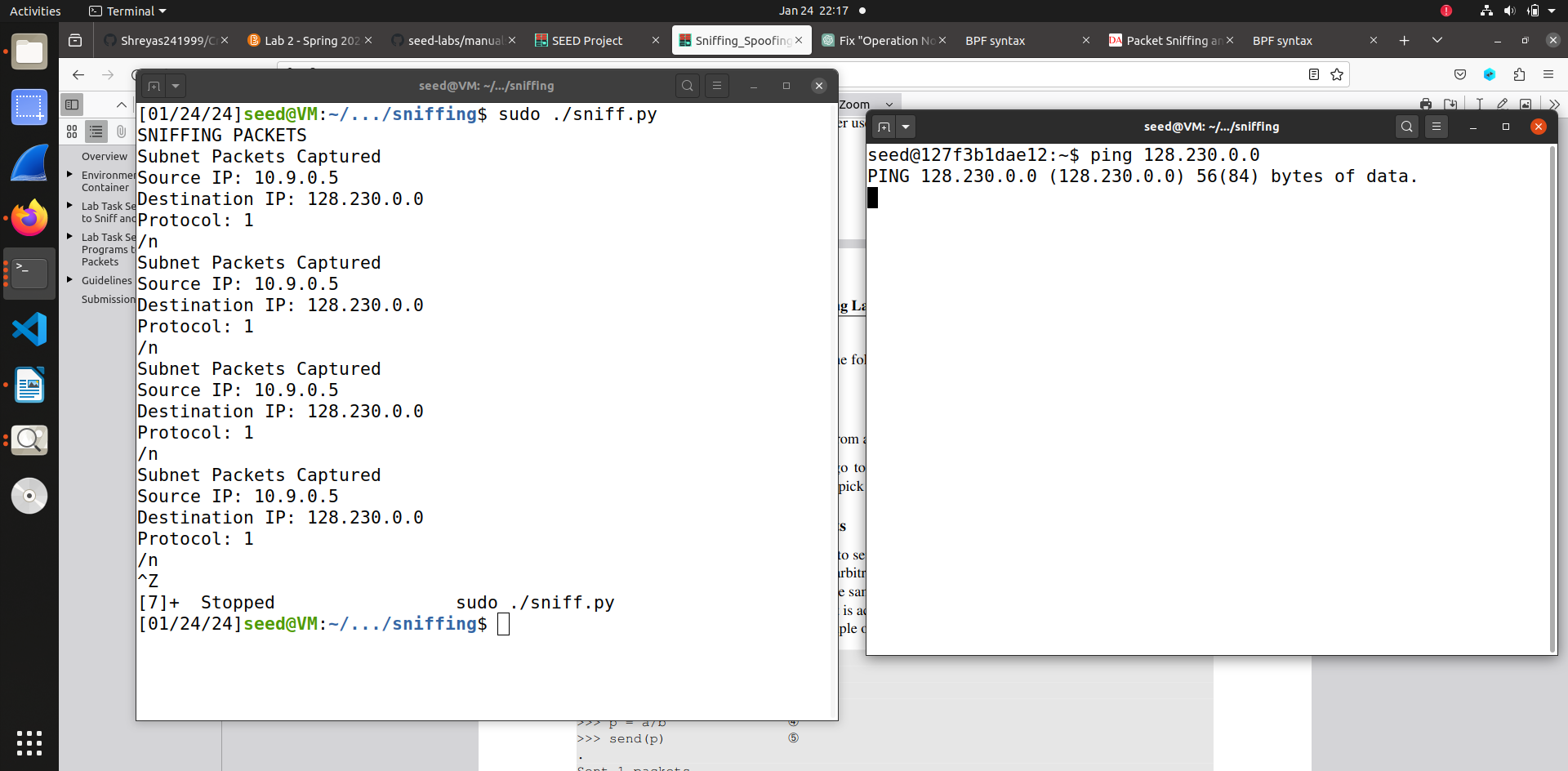
****In this code we captured ICMP packet using the filter function of BPF(Berkerley Packet Filter) and printed the Source IP Address, Destination IP Address and Protocol. So when we execute this code and send a ping to IP address it generates an ICMP echo request packet and reply the print out message.

**Capture TCP packet coming from particular IP with destination port number 23**

****

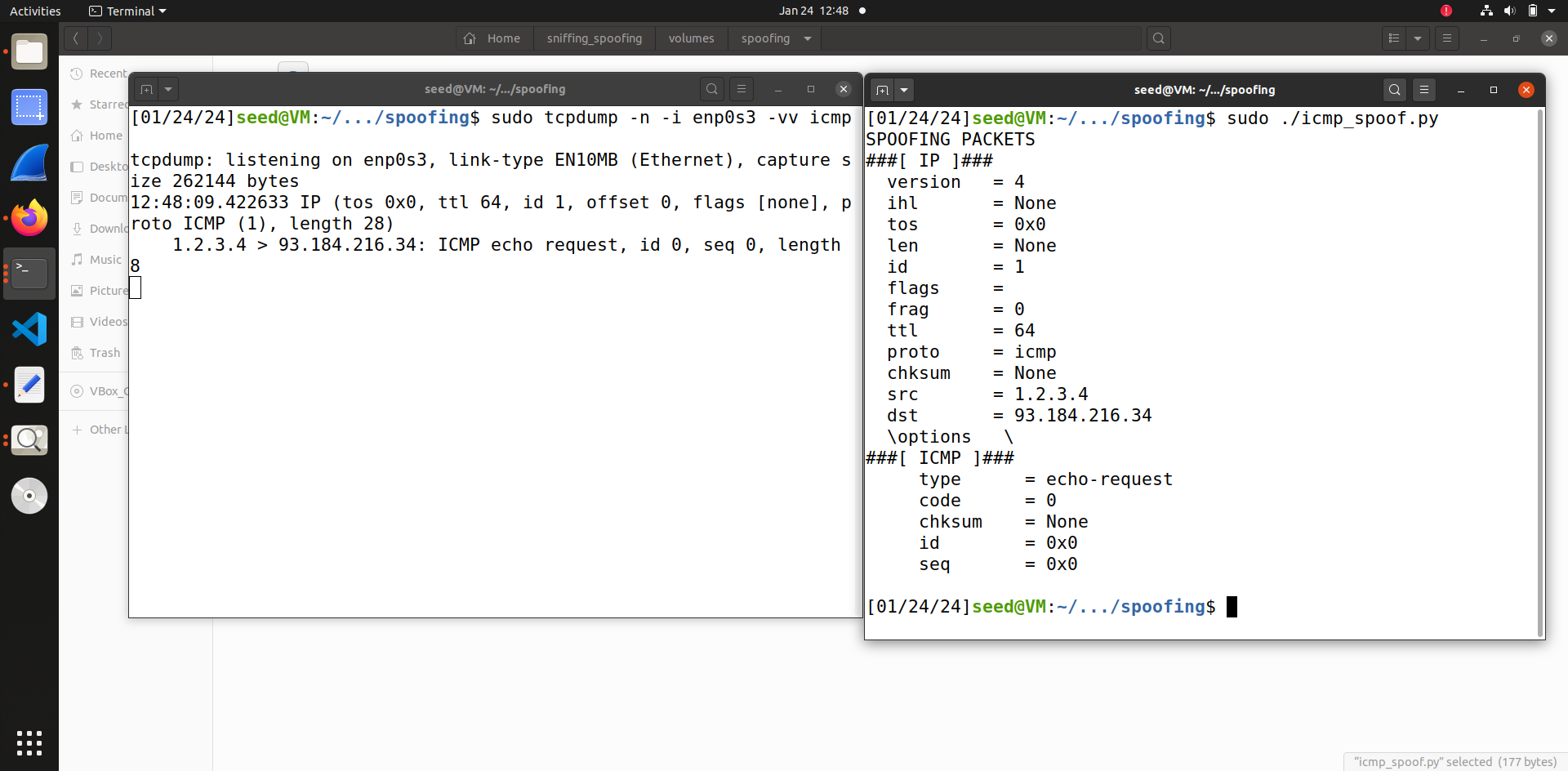
In this code I made changes in the filter argument by using “tcp port 23 and src port as 10.9.0.5”. I got the syntax from BPF website from where I also got the syntax for ICMP packet . So I first did docksh and entered host b and from there I telnet 10.9.0.5 from which I got the TCP packets captured with destination port as 23.

**Capture packets from a particular subnet**

In the above code I used once again the BPF filter and added the destination subnet as 128.230.0.0/16

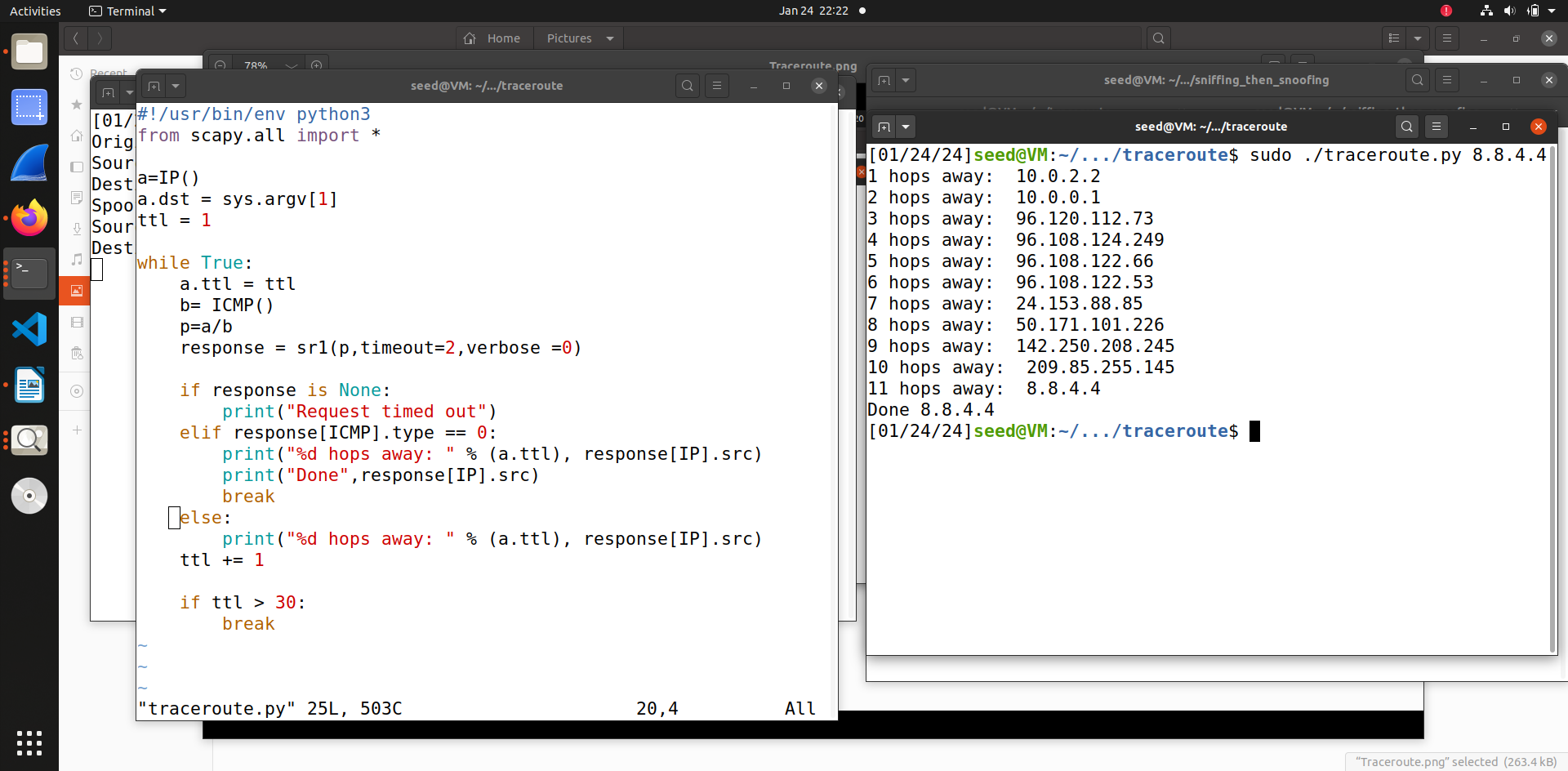
with dst which is the possible direction. The net returns true for the subnet provided. So using ping I send a packet to the particular subnet which was captured with the source IP address and Destination IP Address.

**Task 1.2 Spoofing ICMP Packets**

****

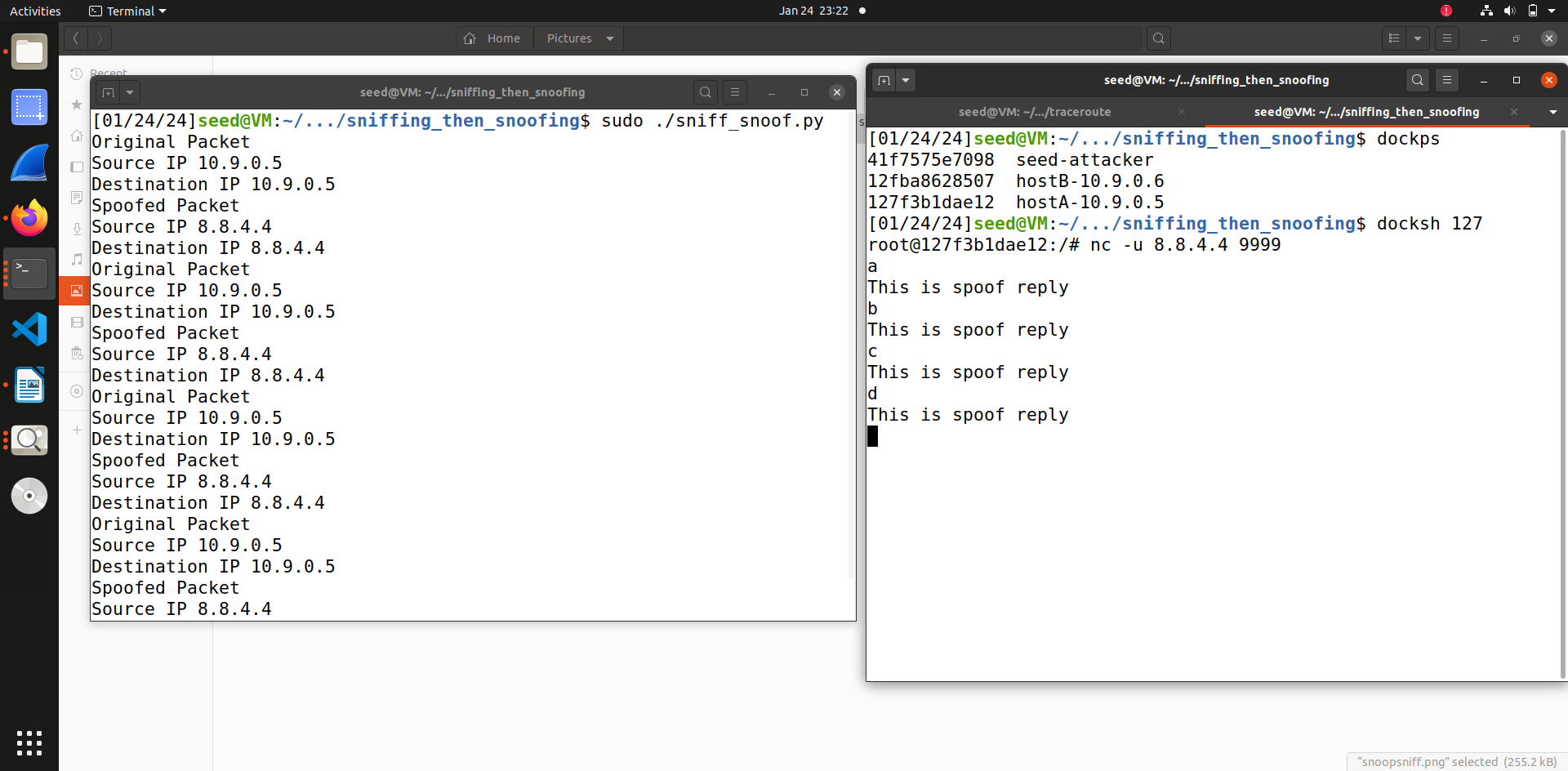
In this code I sent an ICMP packet to defined destination IP address ‘’93.184.216.34’ and random source as ‘1.2.3.4’. So the ICMP packet was spoofed and sent to another subnet. Using Scapy library I overwrote with my own random IP and sent the packet to destination IP address, then the destination received the packet.

**Task 1.3 Traceroute**

****

In this program I used traceroute from the scapy library and added the destination IP address as ‘8.8.4.4’. The ttle flag that is time to leave is incremented after giving one packet. I used While loop in this to keep iterating as long as it is routing. The sr1() method of scapy will listen and wait for the packet response. The arguments in this methods are timeout = the time limit for response and verbose ignore printing unnecessary details. So the program figure out how many hops it takes to send out the packet all the way to destination address. If we do not get any response then we will print the request time out else if the response type is 0 then it will print how many hops is away and print done if found that source IP and loop will break or else condition for printing the current hop and IP address till we reach the destination IP address and the last condition is if the ttl incremented over 30 it will break that means the destination IP address is more than 30 hops away.

**1.4 Task Sniffing and then Spoofing**

****

In this code we flip the src and destination so whenever it sees an ICMP request, regardless of the target IP address the program should immediately reply an echo using the packet spoofing technique. In this code we have printed we send a packet that is sniff it we immediately get replied back with the spoofed packet which is shown as a printed message.

**ALL CODES**  
  
For **Task 1.1.B** I have merged code into one for all the three conditions

#!/usr/bin/python3

from scapy.all import \*

print("SNIFFING PACKETS")

def print\_pkt(pkt):

print("ICMP Packets Captured")

print("Source IP:",pkt[IP].src)

print("Destination IP:",pkt[IP].dst)

print("Protocol:",pkt[IP].proto)

print("/n")

def print\_tpkt(pkt):

print("TCP Packets Captured")

print("Source IP:",pkt[IP].src)

print("Destination IP:",pkt[IP].dst)

print("Protocol:",pkt[IP].proto)

print("/n")

def print\_subpkt(pkt):

print("Subnet Packets Captured")

print("Source IP:",pkt[IP].src)

print("Destination IP:",pkt[IP].dst)

print("Protocol:",pkt[IP].proto)

print("/n")

sniff(iface='br-b8a29e96a50d',filter='icmp',prn=print\_pkt)

sniff(iface='br-b8a29e96a50d',filter='tcp port 23 and src host 10.9.0.5',prn=print\_tpkt)

sniff(iface='br-b8a29e96a50d',filter='dst net 128.230.0.0/16',prn=print\_subpkt)

Task 1.2 Spoofing ICMP Packets

#!/usr/bin/python3

from scapy.all import \*

print("SPOOFING PACKETS")

ip = IP(src = "1.2.3.4",dst = "93.184.216.34")

icmp = ICMP()

pkt = ip/icmp

pkt.show()

send(pkt,verbose=0

Task 1.3 Trace route

#!/usr/bin/env python3

from scapy.all import \*

a=IP()

a.dst = sys.argv[1]

ttl = 1

while True:

a.ttl = ttl

b= ICMP()

p=a/b

response = sr1(p,timeout=2,verbose =0)

if response is None:

print("Request timed out")

elif response[ICMP].type == 0:

print("%d hops away: " % (a.ttl), response[IP].src)

print("Done",response[IP].src)

break

else:

print("%d hops away: " % (a.ttl), response[IP].src)

ttl += 1

if ttl > 30:

break

Task 1.4 Sniffing and then Snoofing

#!/usr/bin/python3

from scapy.all import \*

def spoof\_pkt(pkt):

if UDP in pkt:

print("Original Packet")

print("Source IP",pkt[IP].src)

print("Destination IP",pkt[IP].src)

ip = IP(src=pkt[IP].dst,dst=pkt[IP].src,ihl=pkt[IP].ihl,ttl=50)

udp = UDP(sport=pkt[UDP].dport,dport = pkt[UDP].sport)

data = "This is spoof reply\n"

newpkt = ip/udp/data

print("Spoofed Packet")

print("Source IP",newpkt[IP].src)

print("Destination IP",newpkt[IP].src)

send(newpkt,verbose=0)

pkt = sniff(iface ='br-b8a29e96a50d',filter = 'udp and src host 10.9.0.5',prn=spoof\_pkt)