1
Name: Guy Cohen
ID: 301198099

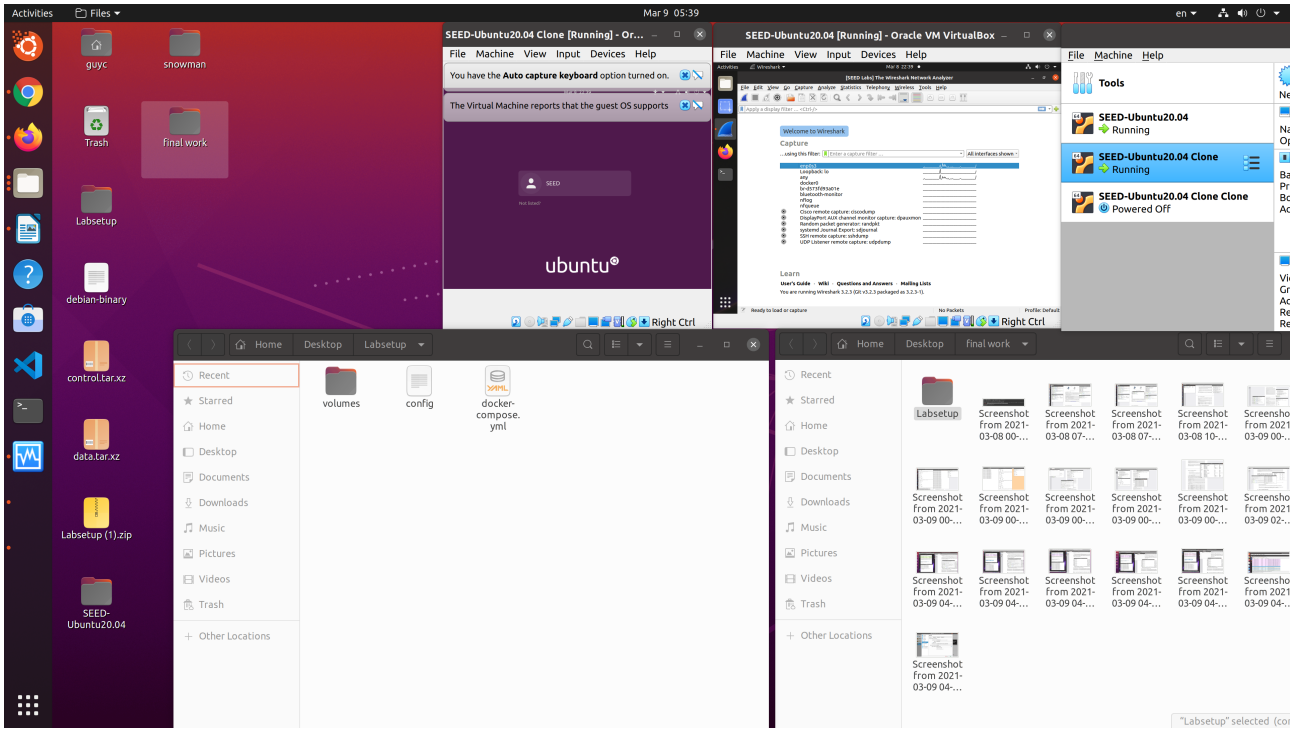
SEED Labs – Packet Sniffing and Spoofing Lab

introduction:

sniffing – many data packets passing in the network and sniffing is the process to monitroed them. Snifers are usally used by network administrators to monitor and troubleshot the network traffic, but not only network administrators use sniffers also attackers use sniffers to monitor and capture data packet to steal sensitive information containing password and user accounts.
Sniffers can be software or hardware installed on a wide range of devices like computers smartphone and today only on samrt watches and more.

Spoofing – Spoofing is the process in which intruder introduces fake trafic and pretends to be someone else. Spoofing is done by sending packets with incorrect source address over the network. The way to deal with spoofing is to use digital signature.

The project work enviroment is ubuntu 20.04 and seed ubuntu on virtual machine.
The programs i used during the project is virtual machine, wireshark, and seedlabs(ubuntu 20.04 seed version).



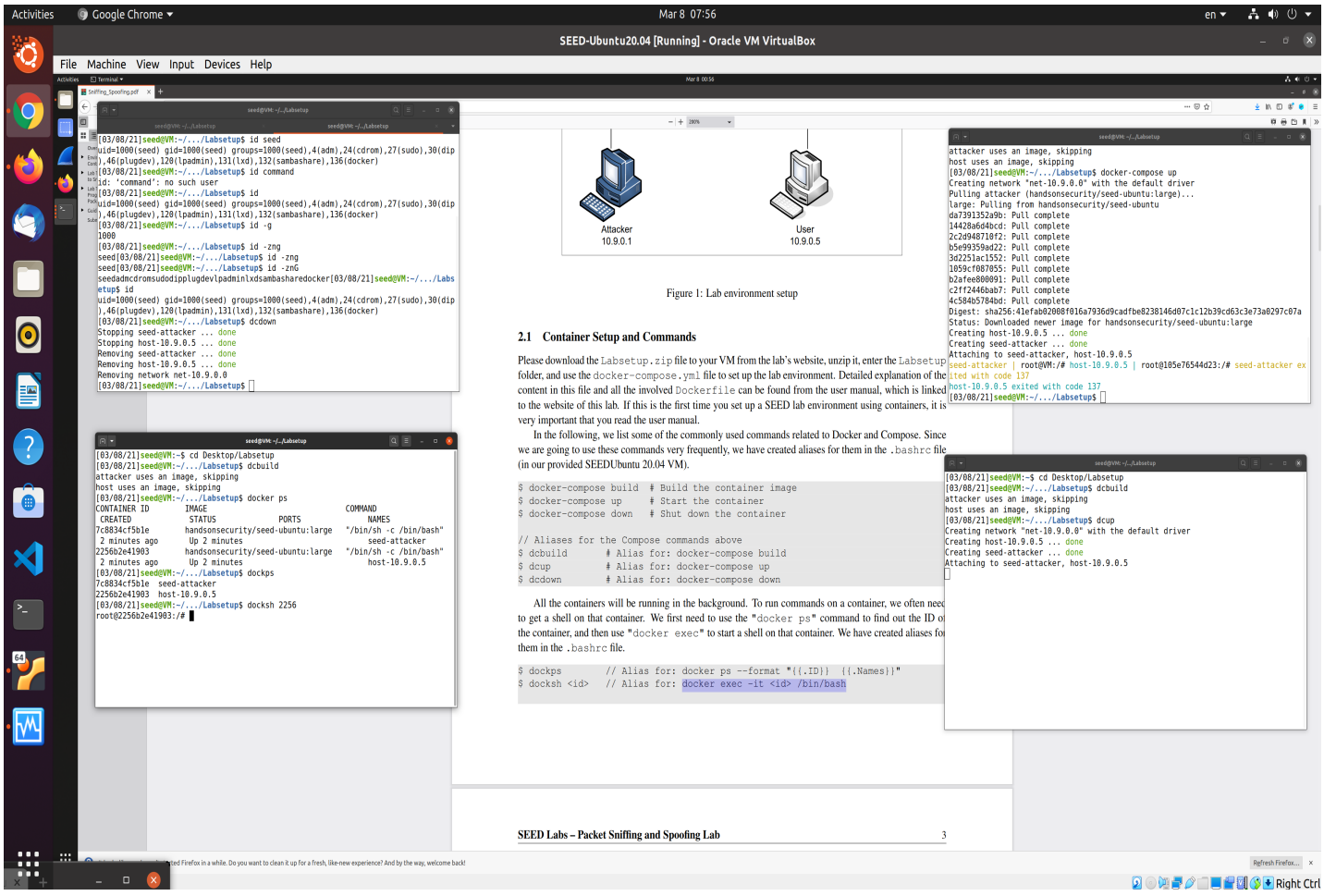
2
Name: Guy Cohen
ID: 301198099

2.1 Container Setup and Commands:

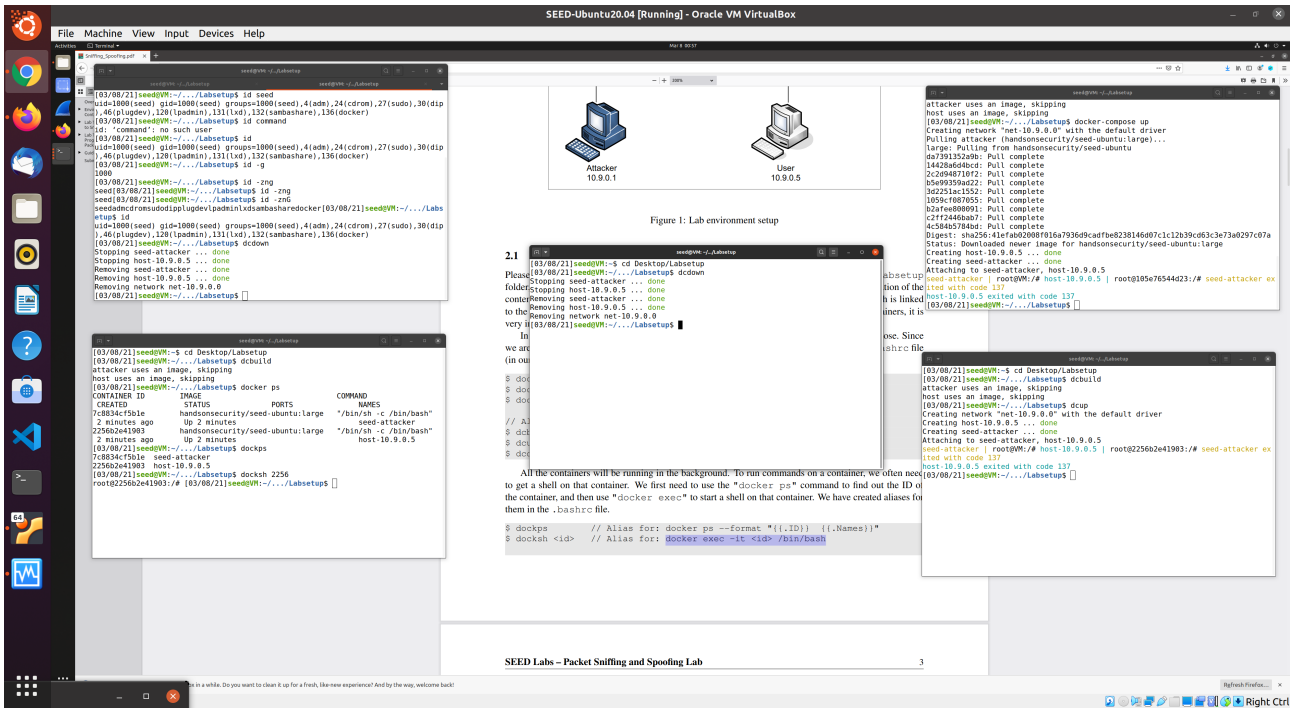
```
$ docker-compose build  # Build the container image
$ docker-compose up     # Start the container
$ docker-compose down   # Shut down the container

// Aliases for the Compose commands above
$ dcbuild       # Alias for: docker-compose build
$ dcup          # Alias for: docker-compose up
$ dcdown        # Alias for: docker-compose down
```

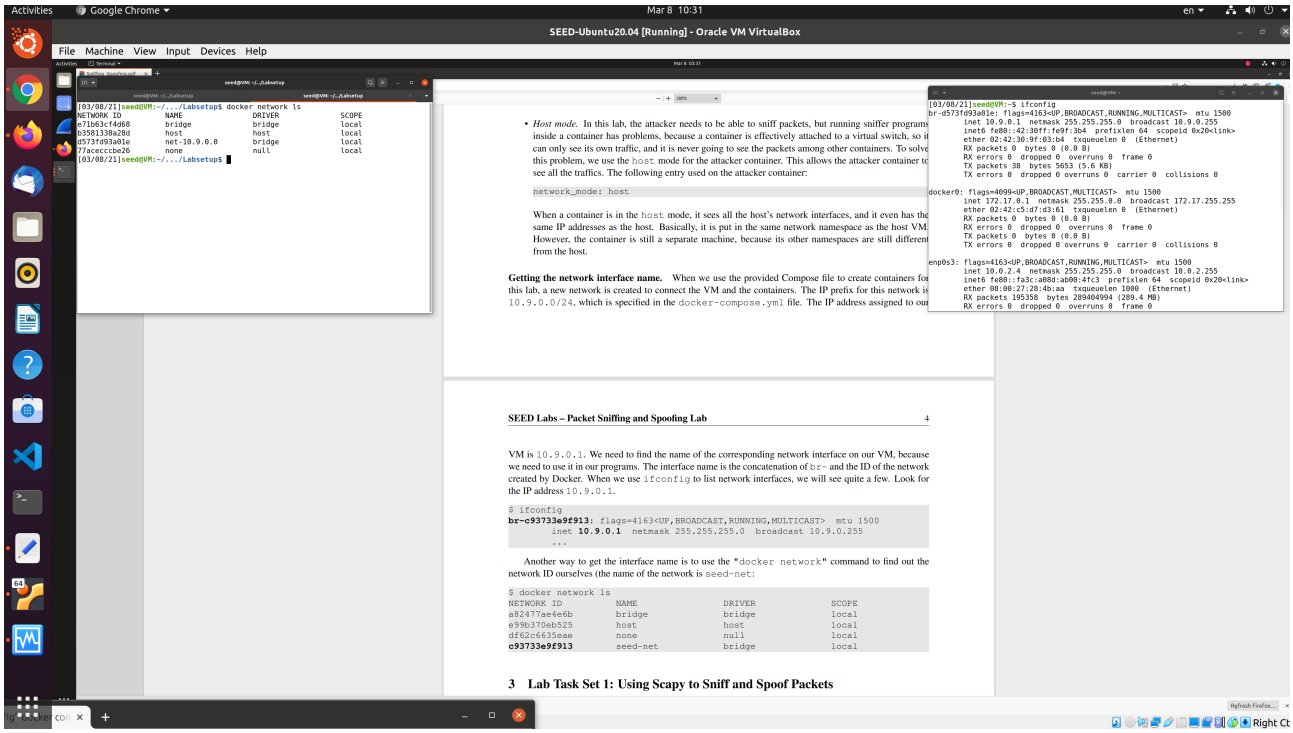
```
$ dockps         // Alias for: docker ps --format "{{.ID}}  {{.Names}}"
$ docksh <id>    // Alias for: docker exec -it <id> /bin/bash
```

Docker It is basically a container engine that uses the Linux Kernel features like namespaces and control groups to create containers on top of an operating system.
I download from the project site yml file(docker-compose) and use it to emulate 2 VM's.



3
Name: Guy Cohen
ID: 301198099





4
Name: Guy Cohen
ID: 301198099

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

i used ifconfig command to saw the name of the network(the last picture at page 3).
br-d53fd93a01e

Open | scapy_sniff.py ~/Desktop/final work/Labsetup/python | Save

```
1 #!/usr/bin/env python3
2 from scapy.all import*
3
4
5 def print_pkt(pkt):
6         pkt.show()
7
8
9 pkt = sniff(iface = ["br-d573fd93a01e", "enp0s3"], filter = "icmp" , prn = print_pkt)
```

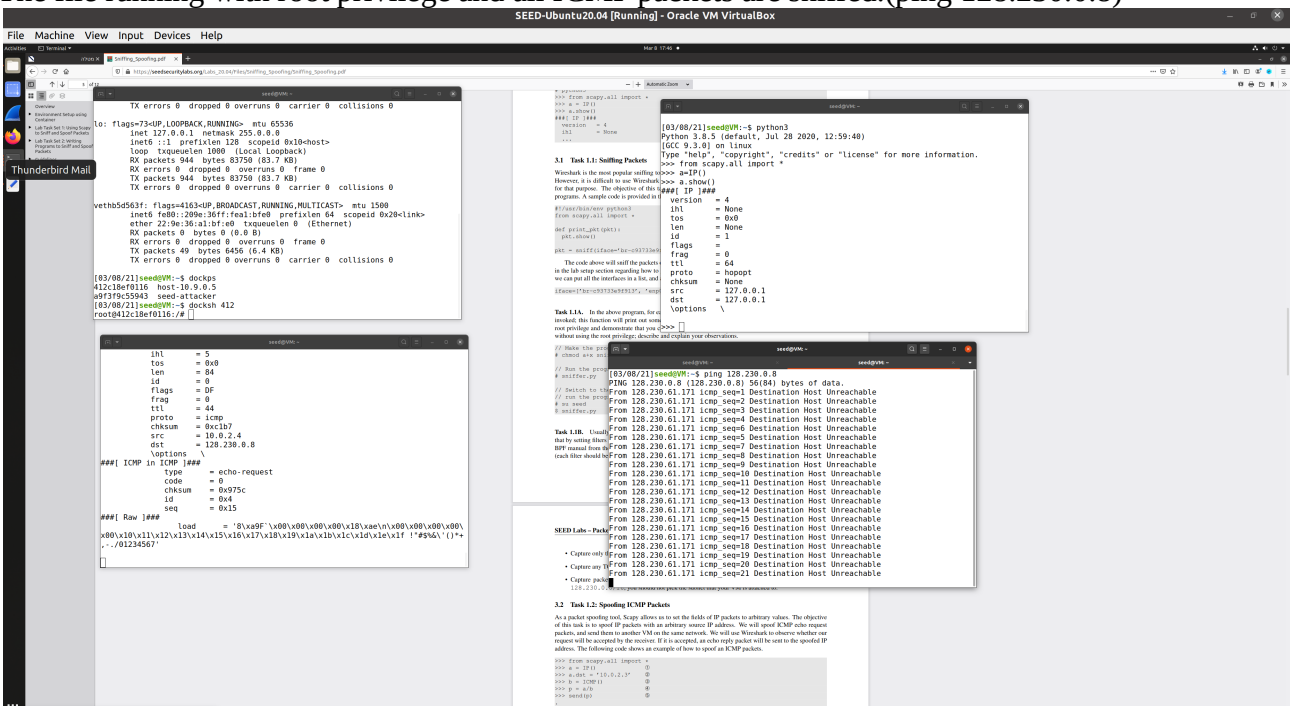
about scappy

scapy is a library that used for interacting with packets on the network.
It has several functionalities through which we can forge and manipulate the packets.
Scapy is a python program that enables the user to send, sniff and dissect forge network packets.
The python program above help me to do packet sniffing easily similar to wireshark program.
it is difficult to use Wireshark as a building block to construct other tools, so we used the scapy program to do it.

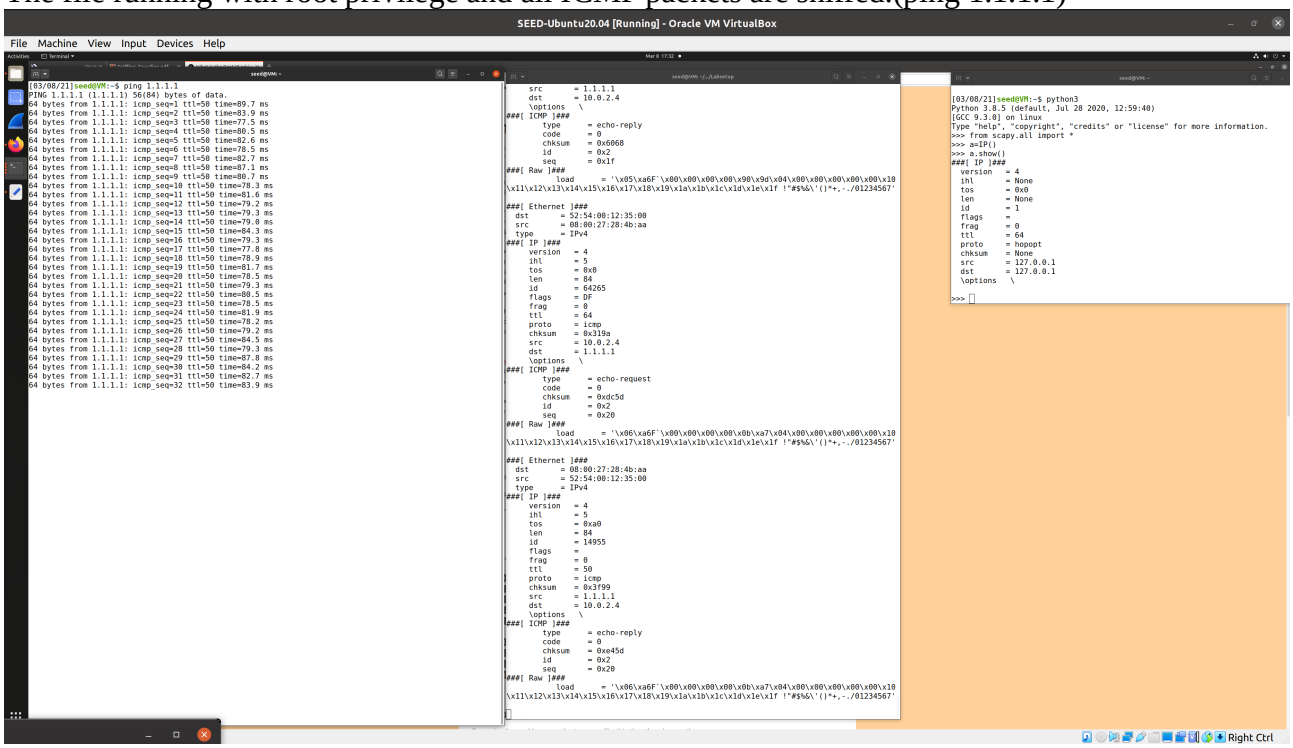
The python code above will sniff the packets on the br-c93733e9f913 interface.
I run the python scapy script twice once with the root privilege and demonstrate and second without using the root privilege.
The different between the two is at the first running i used sudo before the name of the scapy file at the terminal to get root privilege and demonstrate.
In the second attempt i run the scapy file at the terminal withoud sudo.
At the first running file with root privilege the icmp packets sent to 3 differents ping({128.230.0.8}, {1.1.1.1}, {8.8.8.8}).
when i run the scapy file without root privilege i get permission error.

5
Name: Guy Cohen
ID: 301198099

The file running with root privilege and all ICMP packets are sniffed.(ping 128.230.0.8)

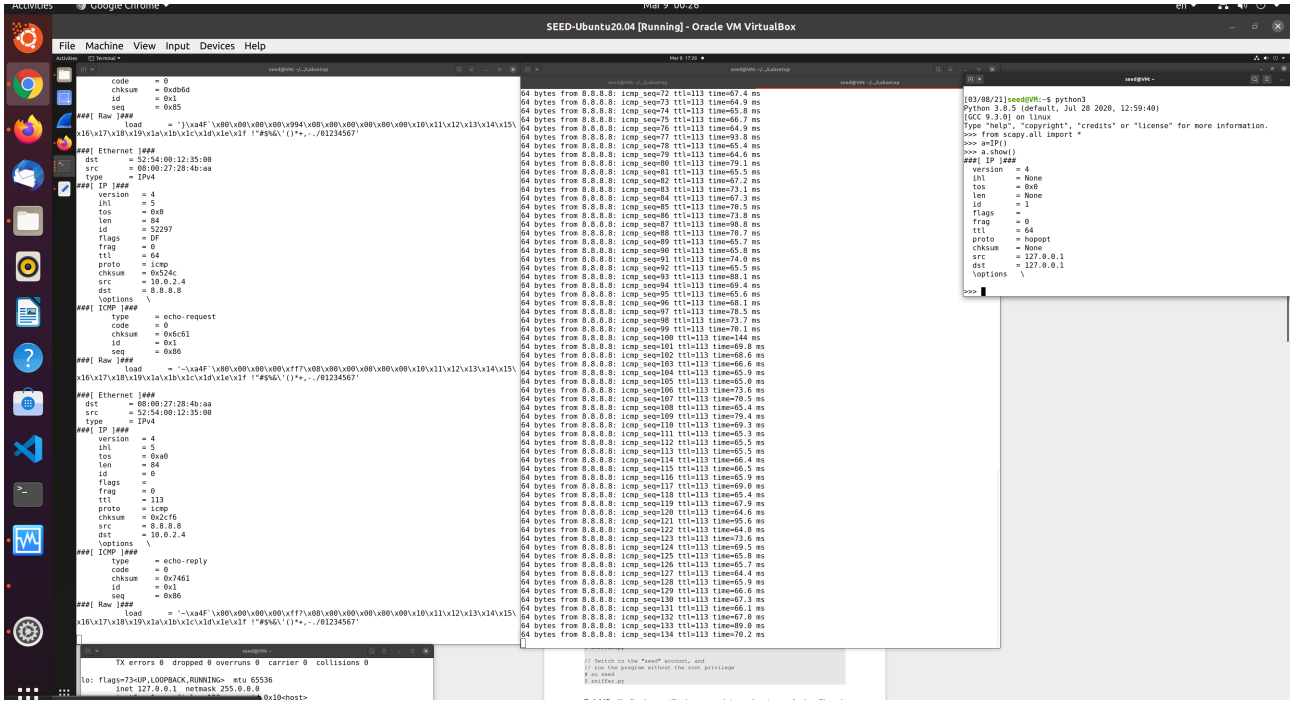


The file running with root privilege and all ICMP packets are sniffed.(ping 1.1.1.1)

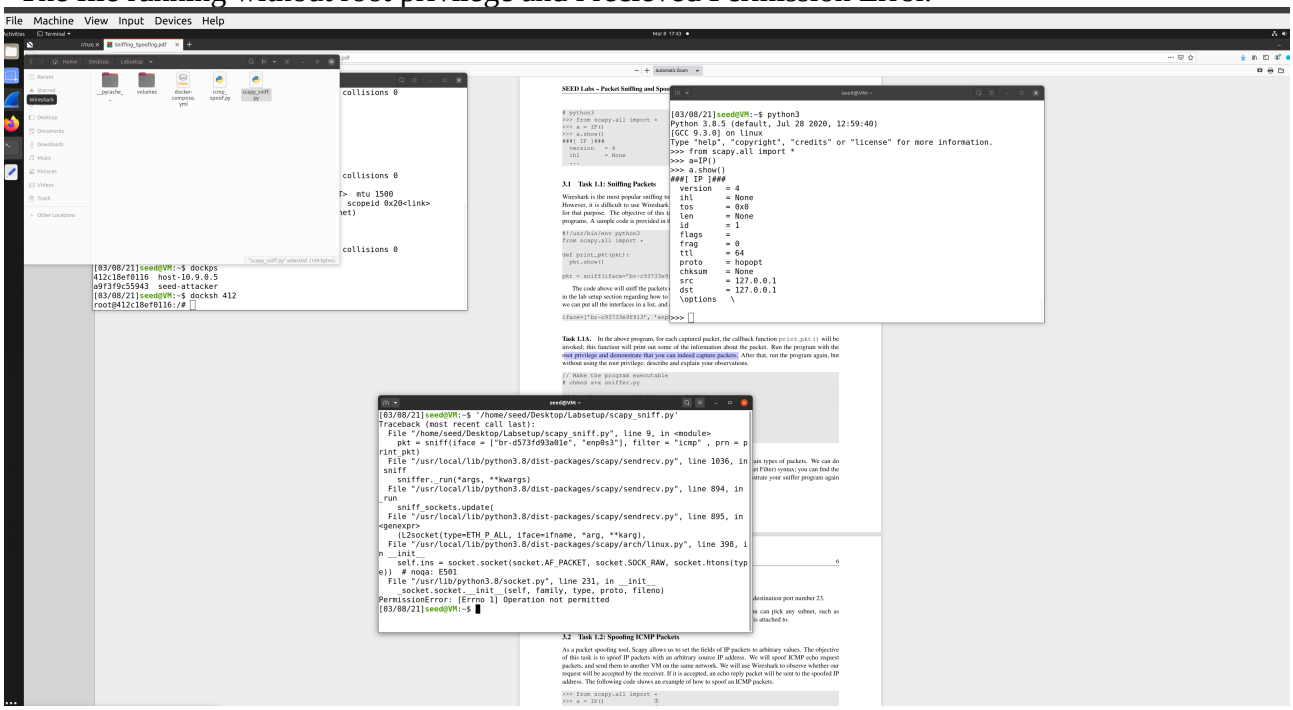


6
Name: Guy Cohen
ID: 301198099

The file running with root privilege and all ICMP packets are sniffed.(ping 8.8.8.8)



The file running without root privilege and i recieved Permission Error.



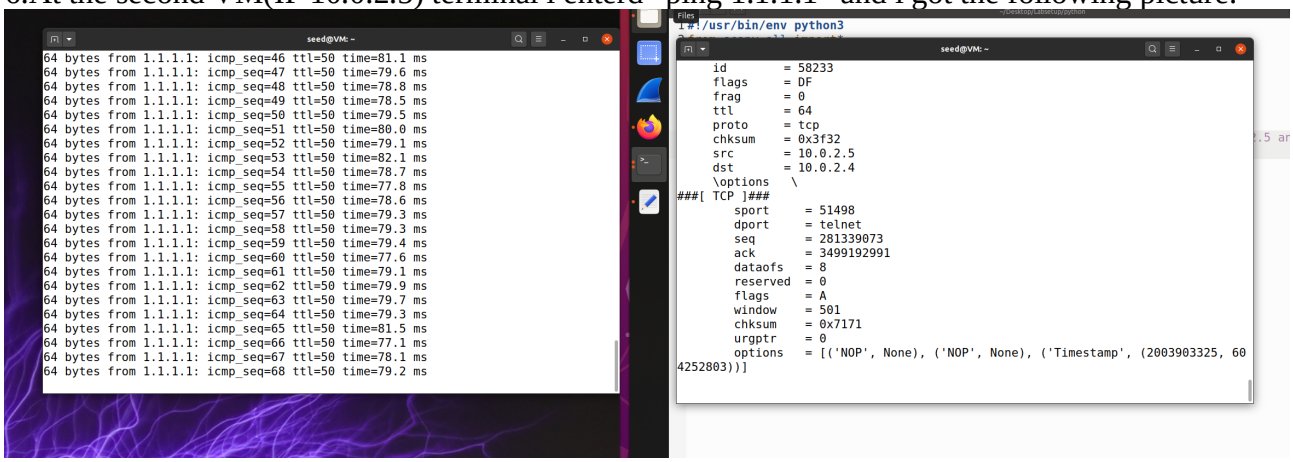
The above picture(without the picture that show the scapy file run without root privilege) show the capture only the ICMP packet. On the picture we can see that the program acts as a sniffer and captures the ping request sent by another machine on the same network.

7
Name: Guy Cohen
ID: 301198099

Capture any TCP packet that comes from a particular IP and with a destination port number 23.
1.I create another VM.
2.In terminal i typed “ifconfig” to get the IP(10.0.2.5).
3.At the first VM i write a python scapy file (i searched BPF manual filter to get the tcp and port 23 filter)

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


def print_pkt(pkt):
        pkt.show()
pkt = sniff(iface = ["br-d573fd93a01e", "enp0s3"],
 filter = "tcp and (src 10.0.2.5 and port 23)"
 , prn = print_pkt)
```

4.At the first VM(IP:10.0.2.4) terminal i enter “sudo scapy_port23.py” to start sniffing puckets.
5.On the second machine i enterd “telnet 10.0.2.4”(the first VM IP).
6.At the second VM(IP 10.0.2.5) terminal i enterd “ping 1.1.1.1” and i got the following picture:



On the second VM (IP 10.0.2.5 (left pic)) we can see that on sending telnet packets, the sniffer program captures the packet.
at the picture we can see the src (The second VM) and the dst (First VM) and the protocol(tcp).

8
Name: Guy Cohen
ID: 301198099

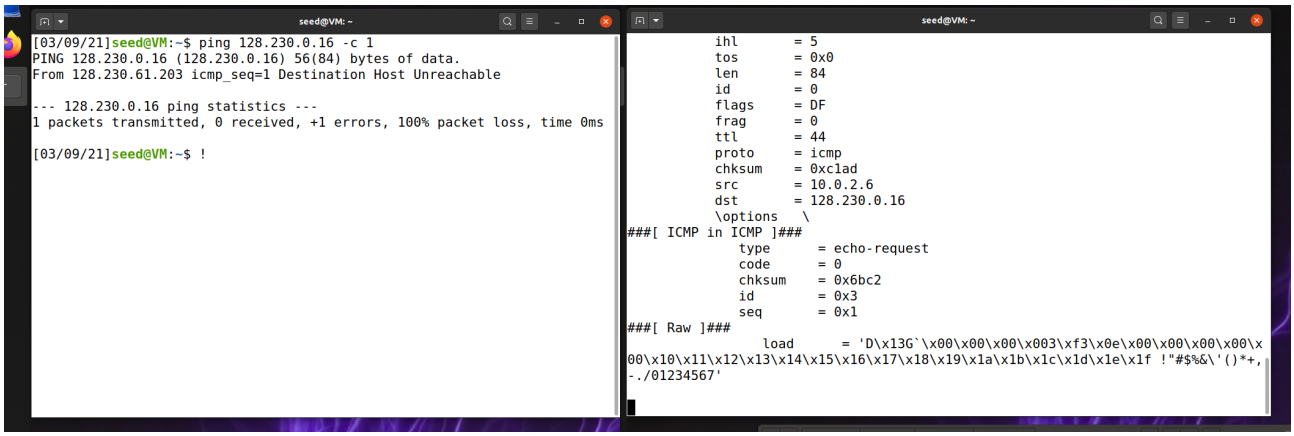
Capture packets comes from or to go to a particular subnet. You can pick any subnet, such as 128.230.0.0/16; you should not pick the subnet that your VM is attached to:
1. I create another VM(IP 10.0.2.6)
2.At the first VM(IP:10.0.2.4) terminal i enter “sudo scapy_subnet.py” to start sniffing puckets.

Open

scapy_subnet.py
~/Desktop/final work/Labsetup/python

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3
4 print("SNIFFING PACKETS...")
5
6 def print_pkt(pkt):
7         pkt.show()
8
9 pkt = sniff(filter="src net 128.230.0.0/16", prn=print_pkt)
```

3.On the third VM(IP:10.0.2.6) terminal i entered ping 128.230.0.16 -c 1 and get following picture.



On the first VM(IP 10.0.2.4) we can see that on sending ICMP packets to 128.230.0.16, the sniffer program capture the packet send out from 128.230.0.16.
at the picture we can see the src (The third VM) and the dst (128.230.0.16) and the protocol(ICMP).

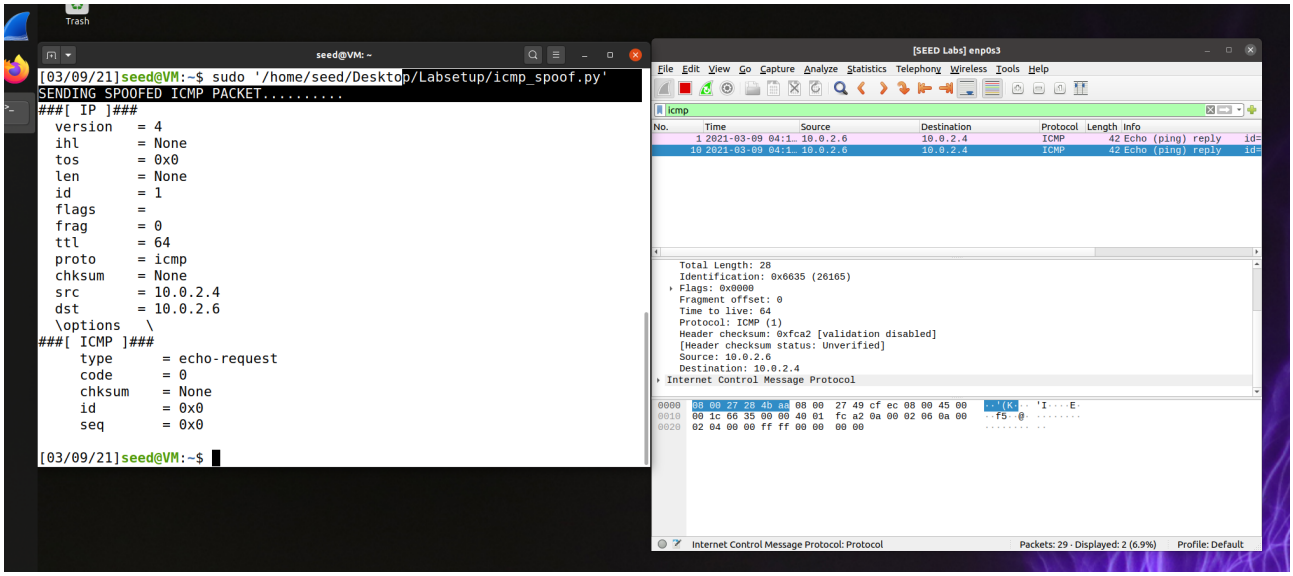
Name: Guy Cohen
ID: 301198099

3.2 Task 1.2: Spoofing ICMP Packets

Open — icmp_spoof.py — ~/Desktop/final work/Labsetup/python

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3
4 print("SENDING SPOOFED ICMP PACKET.....")
5 a = IP()
6 a.src="10.0.2.4"
7 a.dst="10.0.2.6"
8 b = ICMP()
9 pkt = a/b
10 pkt.show()
11 send(pkt,verbose=0)
```

On the first VM(IP: 10.0.2.4) i launched Wireshark app and selected the right interface(enp0s3) and turn on the start capturing packets and i entered icmp at the filter to capture only icmp packets. After Wireshark launched i run the icmp_spoof.py program with root privilege.
The following picture show “somthing strang”, Wireshark show that the package sent from the third VM(IP 10.0.2.6) to the first VM(IP 10.0.2.4) something that was not supposed to happen.
The source supposed to be first VM(IP 10.0.2.4) and the destination supposed to be the third VM(IP 10.0.2.6).



10
Name: Guy Cohen
ID: 301198099

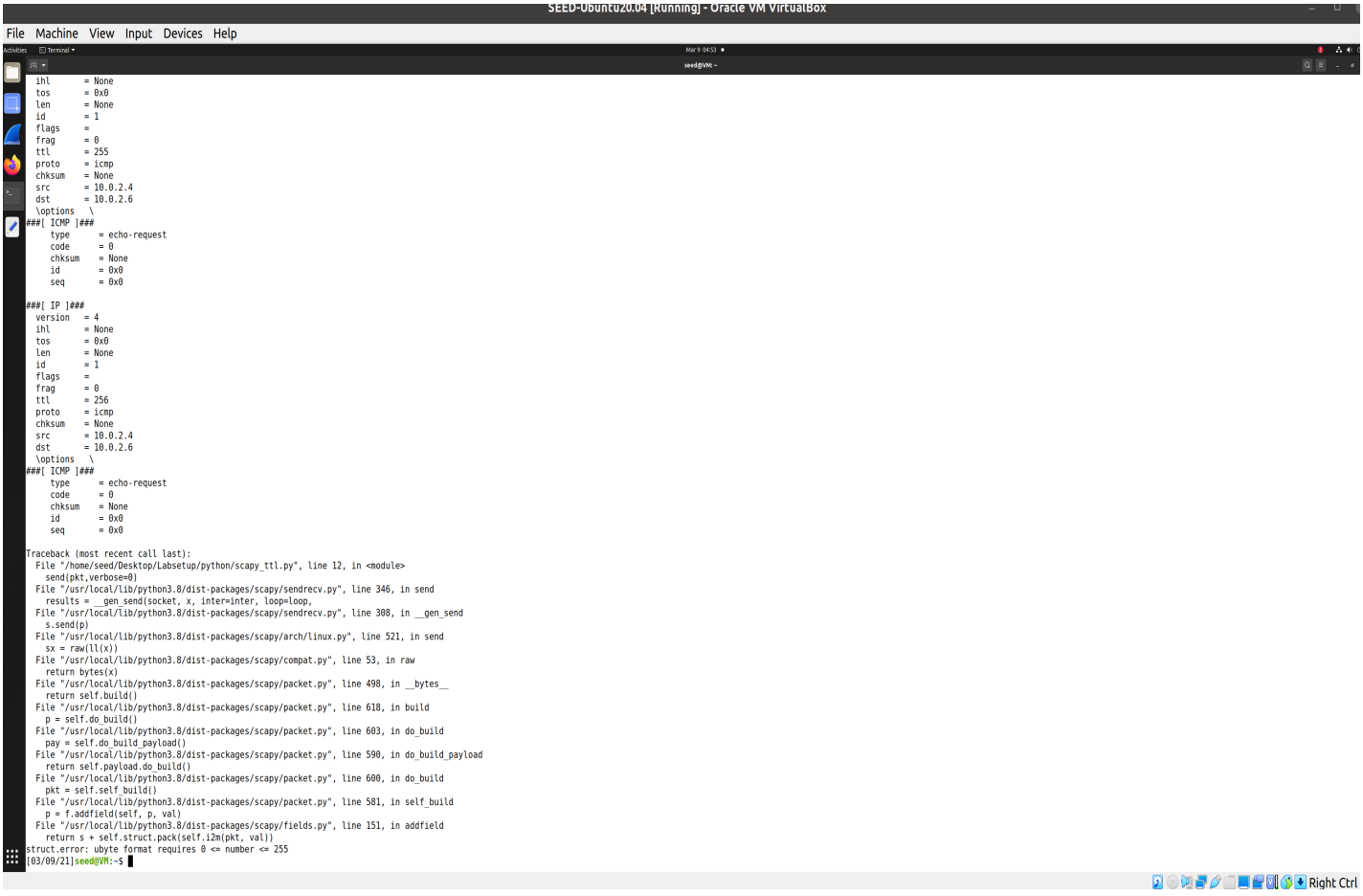
3.3 Task 1.3: Traceroute

At this task we use scapy to estimate the distance in terms of number of routers, between my VM and selected destination.
I build scapy program (scappy_ttl.py) that run from ttl 1 until the program broke.

Open — scapy_ttl.py — ~/Desktop/final work/Labsetup

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

a = IP()
a.src="10.0.2.4"
a.dst="10.0.2.6"
a.ttl = 1
b = ICMP()
while 1:
        pkt = a/b
        pkt.show()
        send(pkt,verbose=0)
        a.ttl+=1
```

The first picture show the code of the program and the second picture show that the value of the ttl is between 1 to 255.
the second picture show that when ttl=256 the program broken.



Name: Guy Cohen
ID: 301198099

3.4 Task 1.4: Sniffing and-then Spoofing

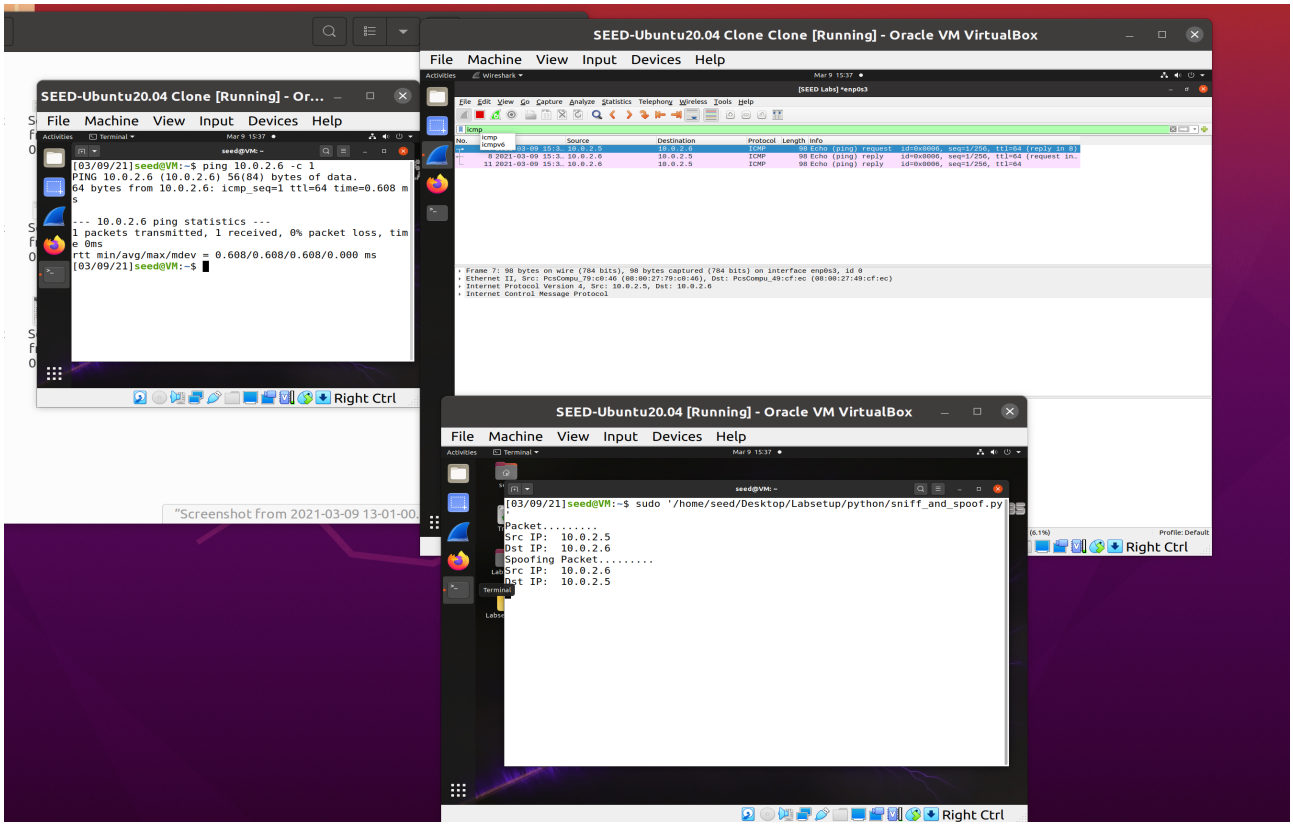
At this part i used three VM’s ubuntu-seed.
VM1 (IP 10.0.2.4)
VM2 (IP 10.0.2.5)
VM3 (IP 10.0.2.6)
At the first Vm(10.0.2.4) i run scapy file (sniff_and_spoof.py).

Open *sniff_and_spoof.py ~/Desktop/final work/Labsetup/python

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

def spoof_pkt(pkt):
    newseq = 0
    if ICMP in pkt:
        print("Packet.........")
        print("Src IP: ", pkt[IP].src)
        print("Dst IP: ", pkt[IP].dst)

        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

        print ("Spoofing Packet.........")
        print ("Src IP: ", newpkt[IP].src)
        print ("Dst IP: ", newpkt[IP].dst)

        send(newpkt,verbose=0)

pkt = sniff(filter='icmp and src host 10.0.2.5',prn=spoof_pkt)
```

The purpose of the program is just like its name to sniff and spoof packets that sent from VM3(10.0.2.6) to VM2 (10.0.2.5).

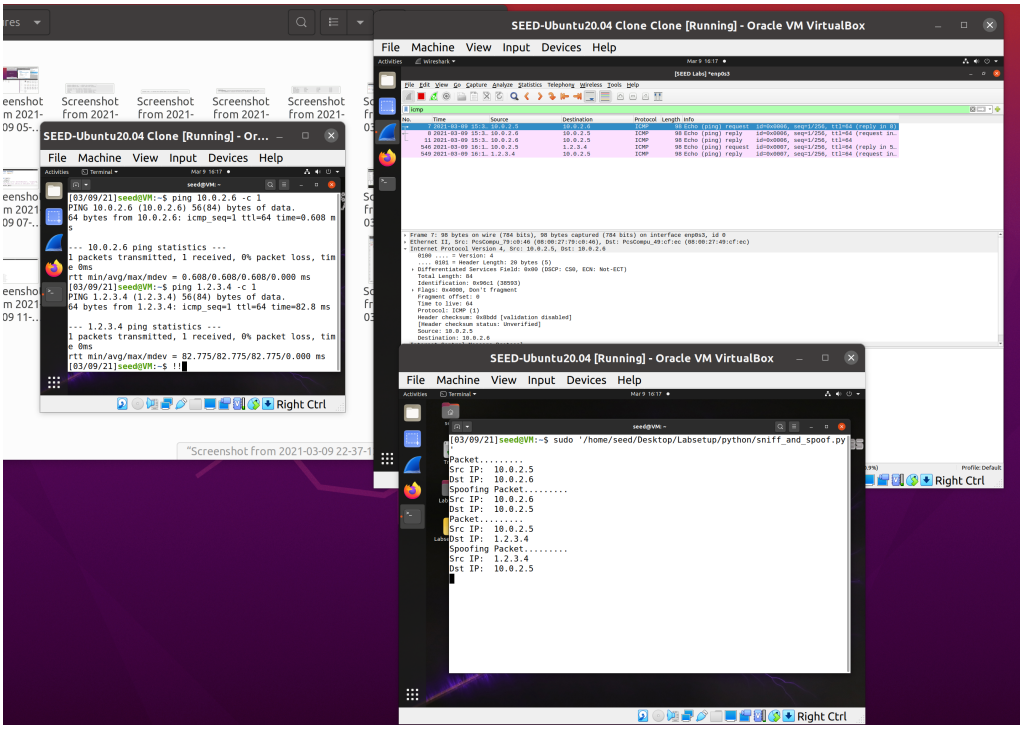
12
Name: Guy Cohen
ID: 301198099



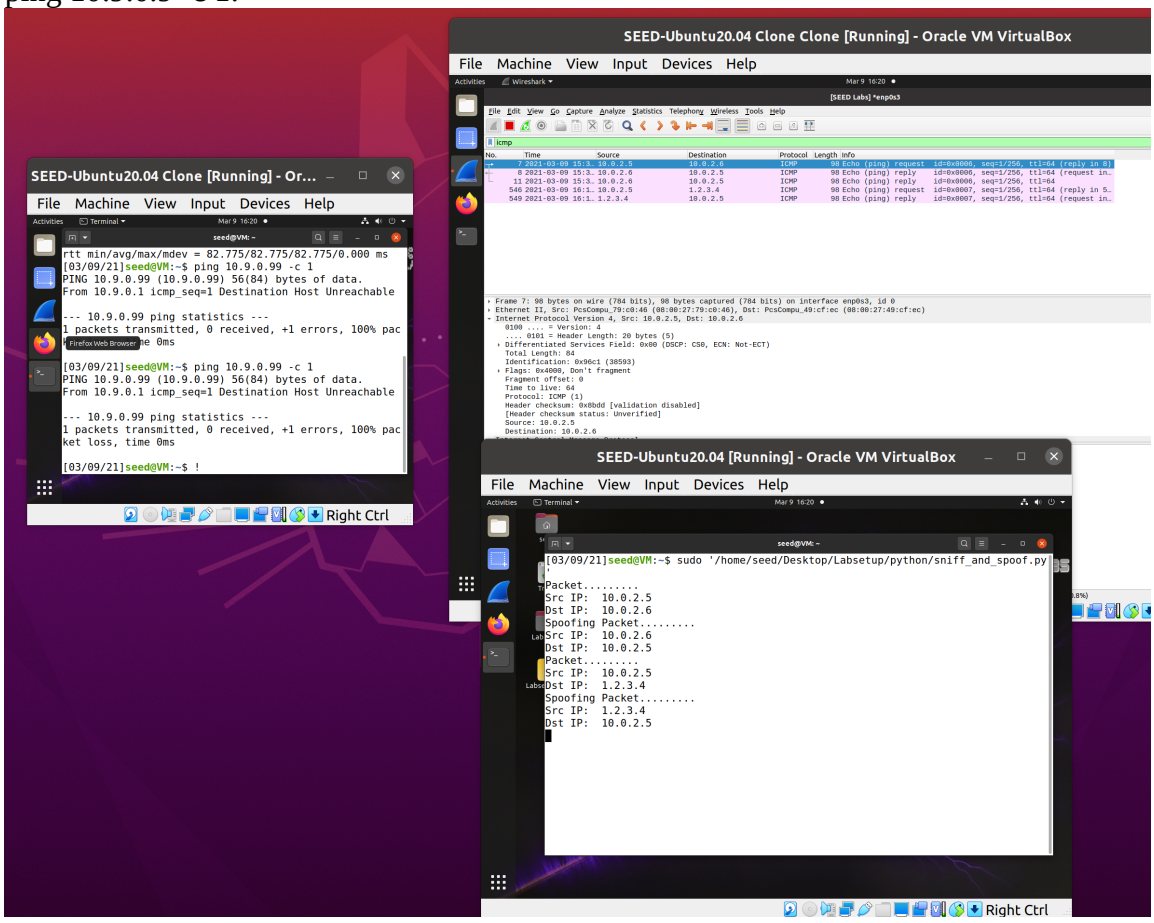
we can see that when i type at VM3(10.0.2.5) terminal ping 1.0.2.6 -c 1 to send 1 packet the program that run on VM1 (10.0.2.4) sniff and spoof the traffic between VM2(10.0.2.5) to VM3(10.0.2.6).
At VM(10.0.2.4) i run the scapy program, at VM2(10.0.2.5) i start wire shark with icmp filter.
At VM3(10.0.2.6) i opened terminal and send the pings.
The purpose of the scapy program is to sniff and spoof the trafic between VM2 to VM3.
We can see at the picture above that Wireshark at VM3 and the scapy program at VM1 captured the request from VM2 and the replay from VM3.
The pictures below shown the the scapy program at VM1 and Wireshark from VM3 when i ping {1.2.3.4}, {10.9.0.99}, {8.8.8.8} at VM2.

13
Name: Guy Cohen
ID: 301198099

Ping 1.2.3.4 -c 1:

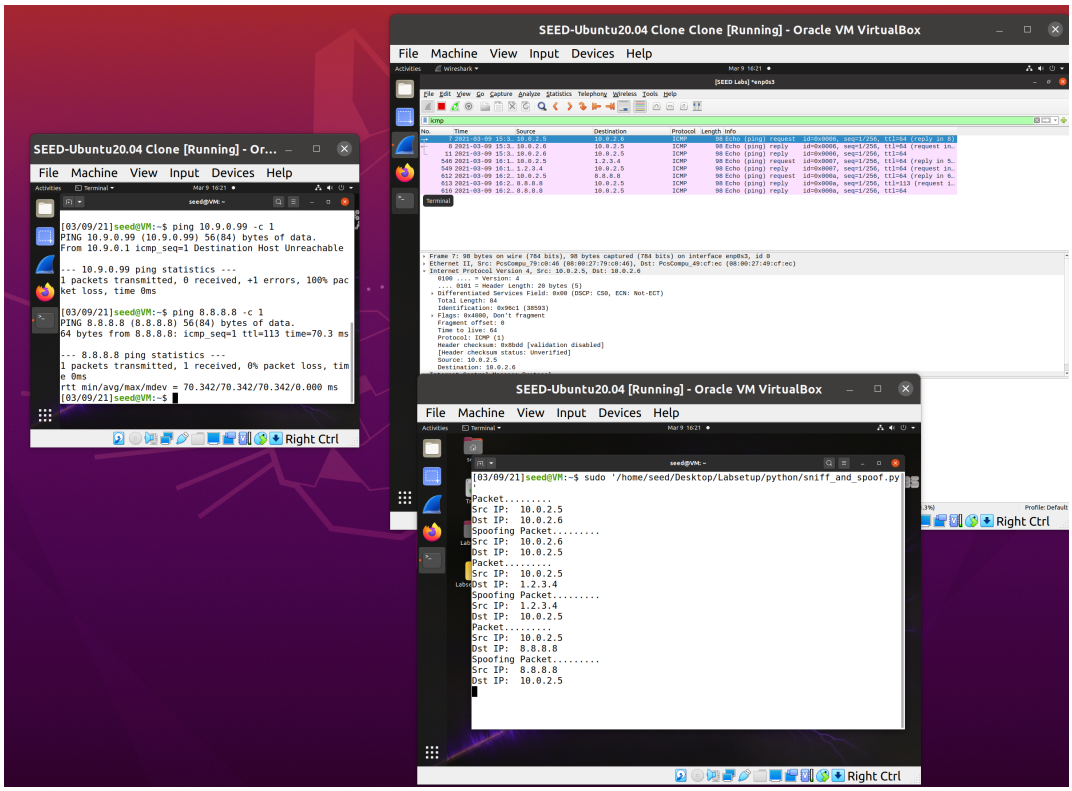


ping 10.9.0.9 -c 1:



14
Name: Guy Cohen
ID: 301198099

ping 8.8.8.8 -c 1:



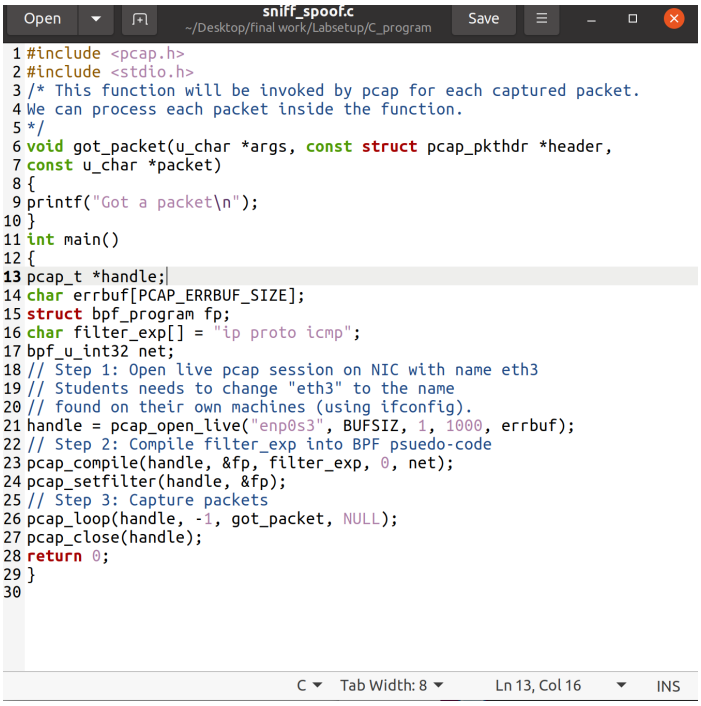
Conclusion:
ping 10.9.0.99 (Non-existing host on the lan) - not sent any packet(100 % packet loss) and the victim did not recieve any answers, it means that the spoof not succeeded.
Ping 1.2.3.4 (Non-existing host) – the victim got answers that is mean that the spoof succeeded. The attacker (VM1) successfuly capture the trafic communication between VM2 and VM3 and the victim recieved response from host that not realy exist.
At this part the spoof was succeeded.
Ping 8.8.8.8(google) – this ping describe existing host on the internet and we can see at last the picture above that the communication is captured succesfully by the attacker that is mean that the spoof succeeded.(the picture above shown the dst and the src ip).

15
Name: Guy Cohen
ID: 301198099

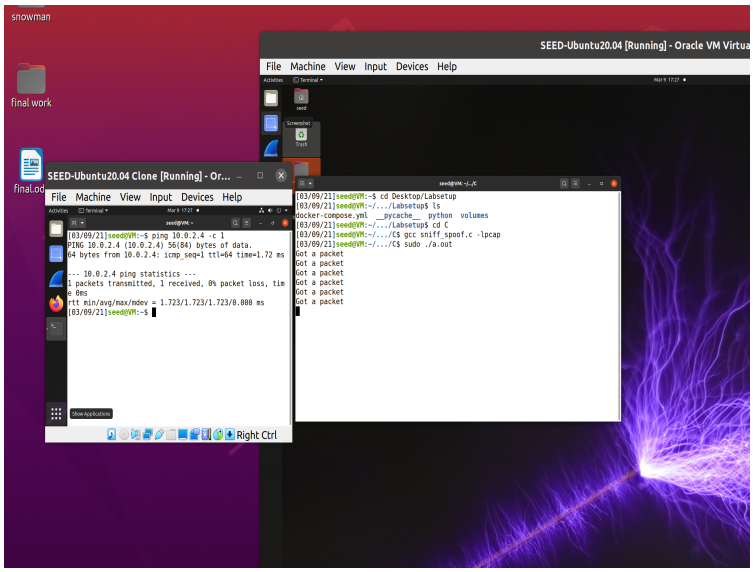
4.Lab Task Set 2: Writing Programs to Sniff and Spoof Packets

I took the C program from 4 task 2.1 and change the name to the VM1(IP: 10.0.2.4) name(ifconfig at VM terminal show me the name) to “enp0s3 and run the code (compile with gcc by command “gcc sniff_spoof.c -lpcap).
At the second VM (VM2 – IP: 10.0.2.5) terminal i typed ping 10.0.2.4 -c 1 to send 1 packet and wait to see if VM1 get the packet.(look at the pictures below).

The C program with name change to enp0s3
.



The C program run on VM1 and the Packet sent from VM2



4.1 Task 2.1: Writing Packet Sniffing Program

Q1

1.Setting up device – setting up the device and decide which interface(Wireshark,X11,eth0,enp0s3...) to start capturing with.
2.Initialze sniffing - sniffer initialize PCAP and tell it to sniff a particular device to create an enviroment for sniffing called as session.
Sniffing can ocuur in multiple devices or multiple sniffing on a single device and this part managed the 2 options.
3.Traffic filtering – at this part we define :rules” like which packets are to be sniffed and analyzed. We need write our rquirements in filter string and then compile it to apply the “rule”.
4.Execution – There are 2 option:
a.packet is sniffed and then analyzed.
b.enter into a loop that waits for x number of packets to be sniffed before we go for analyzing.
We need to configure which option we want, and we can display the result immediately or save them in a file for future record.
5.Ending Session – End for session when we are done with the requirements of sniffing or we have enough data to analyze.

16
Name: Guy Cohen
ID: 301198099

1.pcap_loookupdev: Finds a capture device to sniff on.

2.pcap_lookupnet: Returns the network number and mask for the capture device.

3.pcap_open_live: Starts sniffing on the capture device.

4.pcap_datalink: Returns the kind of device we are capturing on.

5.pcap_compile: Compile the filter expression stored in a regular stringin order to set the filter.

6.pcap_setfilter: Sets the compiled filter.

7.pcap_loop: Process packets from the capture. We can either sniff one packet at a time with pcap_next or use a “loop sniff” with pcap_loop.

8.pcap_free_code: Free up allocated memory.

9.pcap_close: Closes the sniffing session.

Q2

We need root privilege for run sniff_spoof.c because sniff_spoof.c will need to access a network device which non-root user cannot do because sniff_spoof.c need user authorization.

Q3

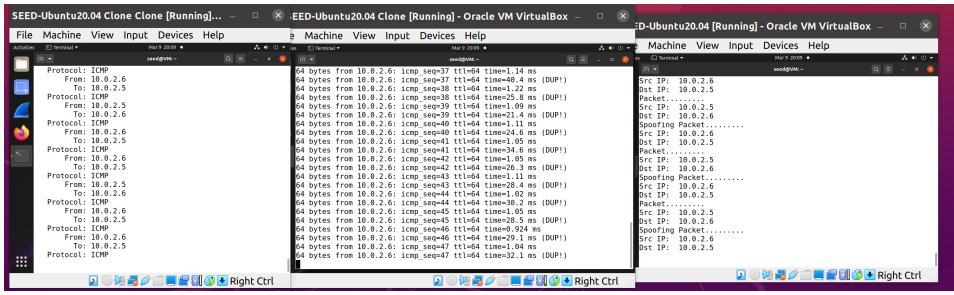
I used seed-ubuntu 20.04 on VirtualBox so i need to turn of the clone machine and get setting/Network/advanced and set the promiscuous Mode “Allow all”.

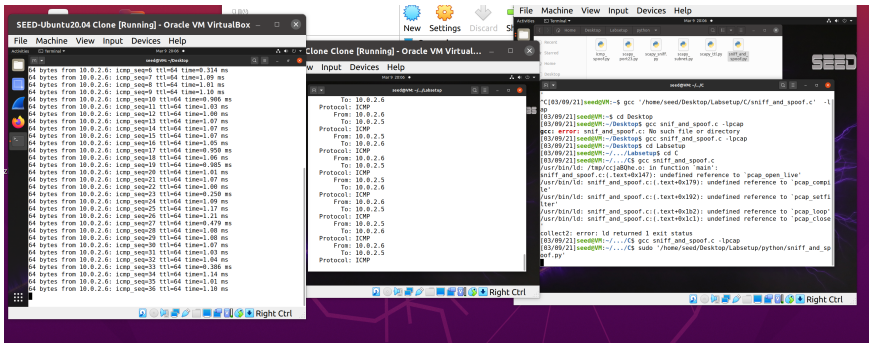
This action allows for a network sniffer to pass all trafic from network.

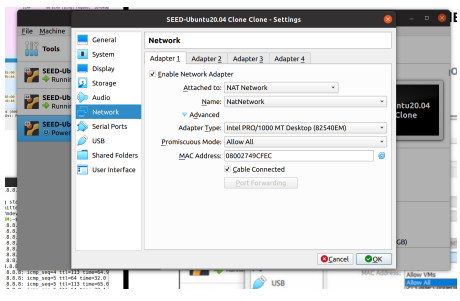
The first picture with “Allow all”

The second picture with “Deny”

the left terminal at deny can’t sniff packets.







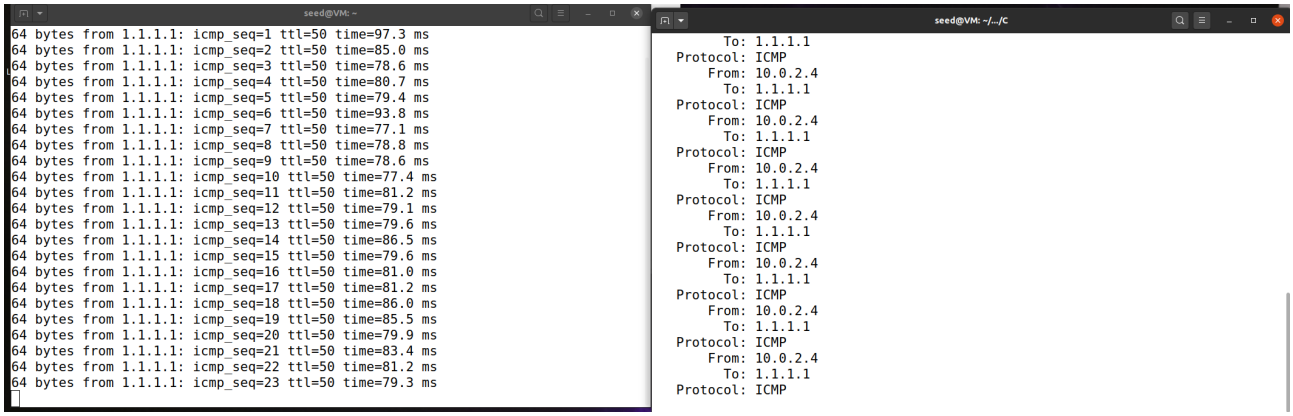
17
Name: Guy Cohen
ID: 301198099

Task 2.1B: Writing Filters.

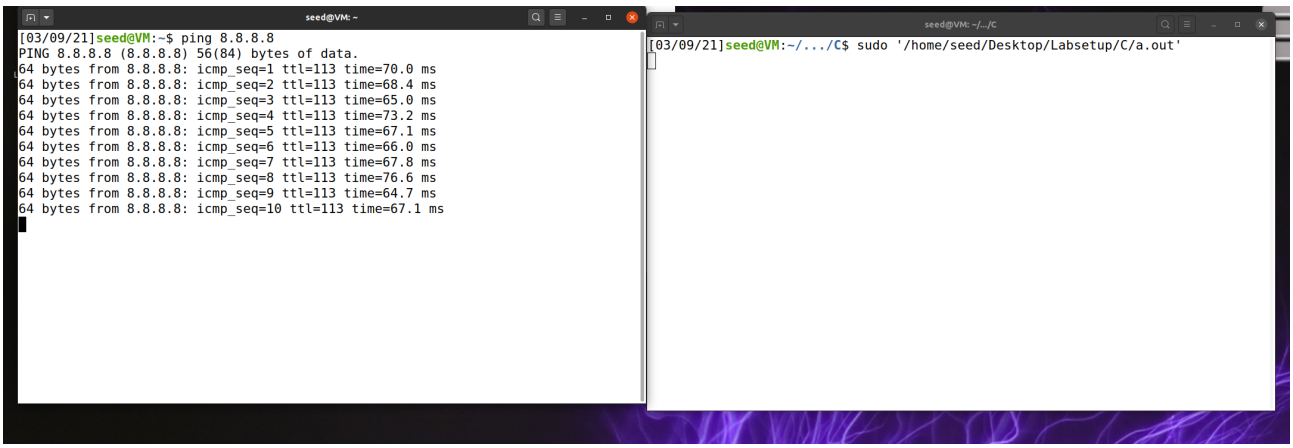
```
char filter_exp[] = "icmp and (src host 10.0.2.4 and dst host 1.1.1.1)";
```

I modifying the string “filter_exp” to capture only icmp packets between my VM1(IP 10.0.2.4) to host 1.1.1.1 and we can see the result in both picture below.

Ping 1.1.1.1 – spoofed succesfully



ping 8.8.8.8 not pass the filter.



We can set the filter in a way that will work in both direction:

char filter_exp[] = “icmp and (src host 10.0.2.4 and dst host 1.1.1.1) or (src host 1.1.1.1 and dst host 10.0.2.4)”.

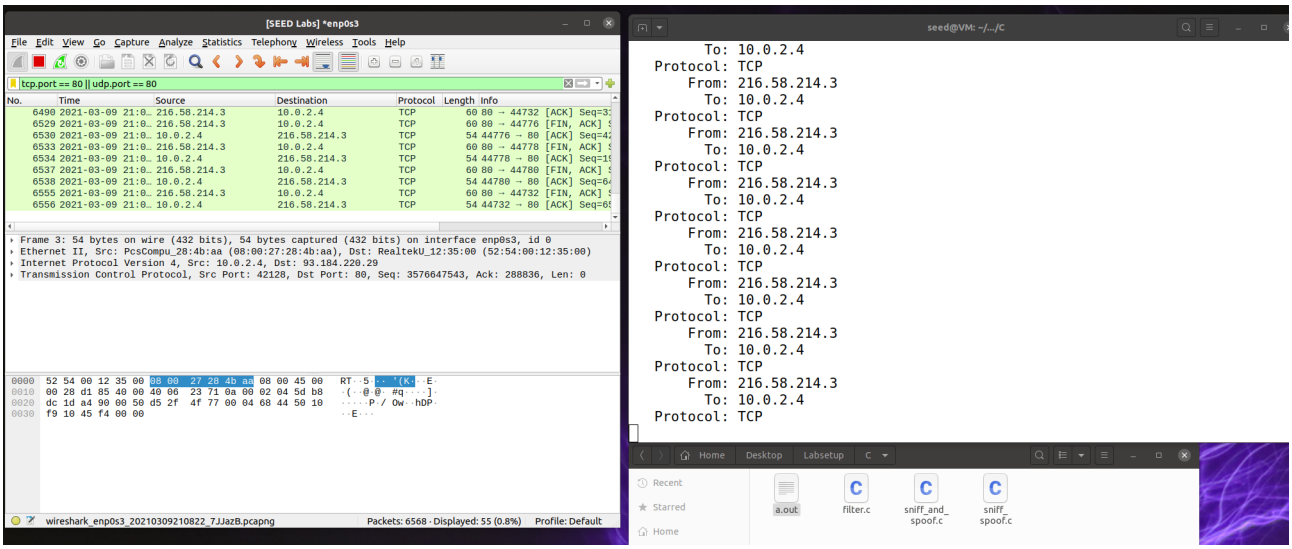
Important:(I used the internet to find out how to write the filter).

18
Name: Guy Cohen
ID: 301198099

Capture the TCP packets with a destination port number in the range from 10 to 100:

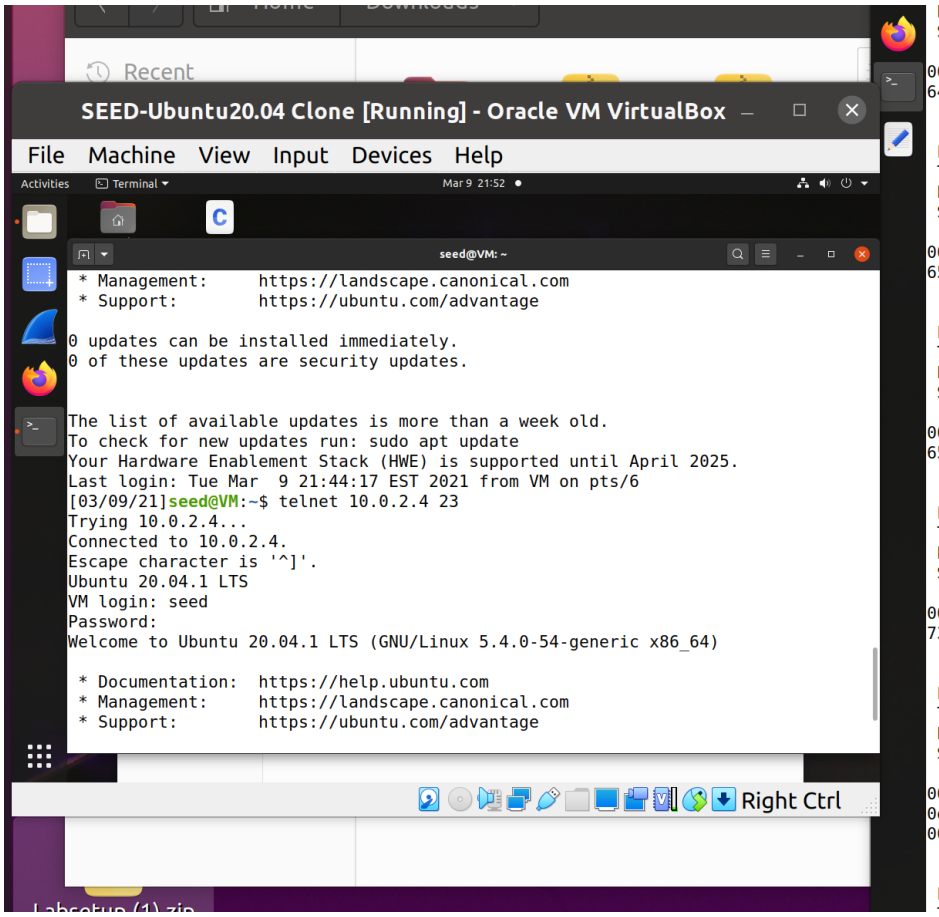
char filter_exp[] = “tcp src portrange 10-100”

```
2   struct bpf_program fp;
3   char filter_exp[] = "tcp src portrange 10-100";
4   bpf_u_int32 net;
```



Task 2.1C: Sniffing Passwords:

19
Name: Guy Cohen
ID: 301198099

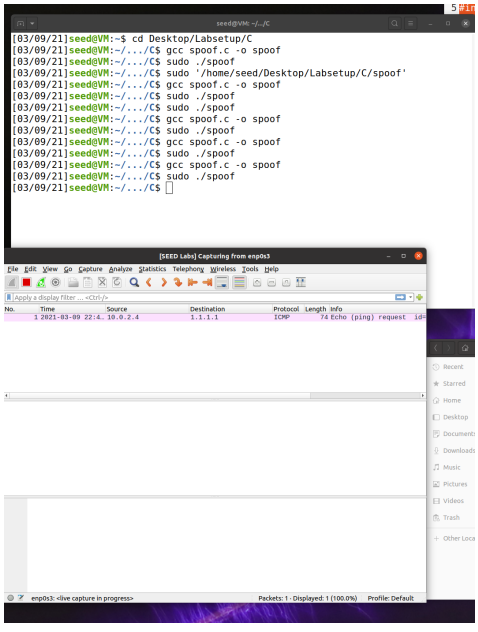



```
Protocol: TCP
Src port: 55384'n  Dst port: 23
 Payload (1 bytes):
00000
64

                d  Packet number 2:
From: 10.0.2.5
To: 10.0.2.4
Protocol: TCP
Src port: 55384'n  Dst port: 23
 Payload (1 bytes):
00000
65

                e  Packet number 3:
From: 10.0.2.5
To: 10.0.2.4
Protocol: TCP
Src port: 55384'n  Dst port: 23
 Payload (1 bytes):
00000
65

                e  Packet number 4:
From: 10.0.2.5
To: 10.0.2.4
Protocol: TCP
Src port: 55384'n  Dst port: 23
 Payload (1 bytes):
00000
73

                s  Packet number 5:
From: 10.0.2.5
To: 10.0.2.4
Protocol: TCP
Src port: 55384'n  Dst port: 23
 Payload (2 bytes):
00000
0d
00

               ..  Packet number 6:
From: 10.0.2.5
```

At VM2(IP: 10.0.2.5) i open terminal and open telnet, after i set the password (pass: dees)

at VM1 the C program capture the password.

4.2 Task 2.2: Spoofing:



Name: Guy Cohen
ID: 301198099

packet spoofed

spoof.c

Task 2.2B: Spoof an ICMP Echo Request:

Q4:

We can check this problem with code above.

We can set that the total lenght over 2000 bytes but such a situation we will get an error because the len parameter on the sendto function call has not been modified, so the packets total lenght is overwritten to its original size when is sent.

So we can not set the IP packet lenghtfield to an arbitary value.

Q5:

NO: The local system must caculate a checksum for the sum value of the header's bits and store this value in the header checksum field to prevent problems with the next device that recieve the IP data because the next device verify that the Header Checksum matches the values seen in the rest of the header If the values do not agree, the data is assumed to have become corrupted and must be destroyed.

Q6:

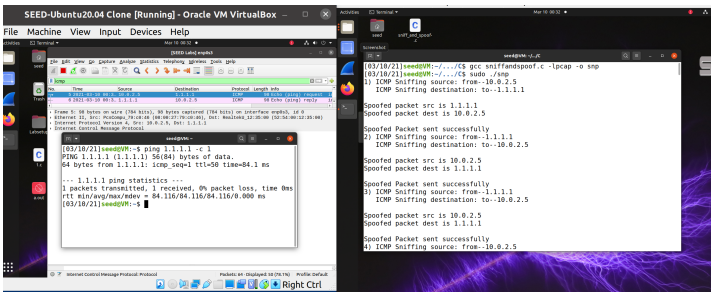
Without root we cannot bind on a port lower than 1024, because with raw sockets we can simulate a sever on any port and sniff/spoof packets.

4.3 Task 2.3: Sniff and then Spoof:

ping 1.1.1.1 on terminal VM2(IP: 10.0.2.5) and Wireshark open on icmp filter.

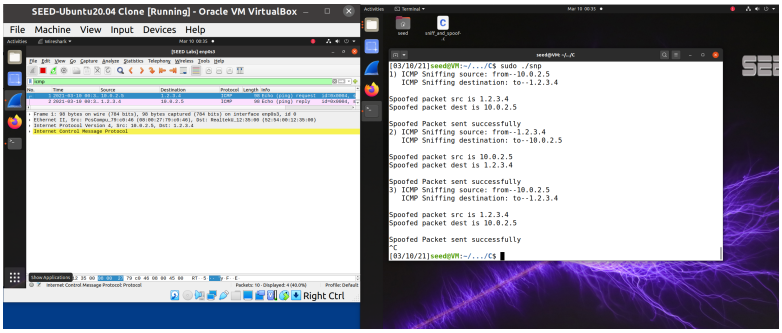
21
Name: Guy Cohen
ID: 301198099

sniff_and_spoof program reun on VM1(IP: 10.0.2.4)



ping 1.2.3.4(non-existing host)on terminal VM2(IP: 10.0.2.5) and Wiresharl open on icmp filter.

sniff_and_spoof program reun on VM1(IP: 10.0.2.4)



Conclusion: at the first picture we can see that when we sniff/spoof on existing host the packets was sent and when we sniff/spoof on non-existing host no packet was sent.

In both attempts the sniff and spoof succeeded we can see it at the terminal on VM1(left pic).