**Task 1.1:**

We were asked to run scapy once using sudo and once without sudo.

After using sudo, we managed to capture the pings sent to google using scapy, both requests and replies.

As is seen below.

Graphical user interface, text

Description automatically generated

However, when we tried using scapy without sudo, we got an error.

And the reason for is, that scapy uses raw sockets in order to listen to the trafic, and in order to do that, you need admin privlileges which you get when using sudo.

So when we used sudo, we could capture ICMP packets without getting an error.

However, the second we tried using scapy without the admin privlieges, we got an error because we didn’t have the proper privlieges.

As is seen in the picture below.

Graphical user interface, text, application

Description automatically generated

**Task 1.1B:**

Here we were asked to use the BPF filter in order to capture:

Only ICMP packets – as seen in the picture below, after running scapy and using netcat to reach the google website without using ping, we didn’t capture any packets with scapy.

Text

Description automatically generated

However, once we used ping, we managed to capture the icmp requests and replies with scapy.

A picture containing graphical user interface

Description automatically generated

Any TCP packet from a source IP address – as seen below, after changing the filter to look for IP 10.0.4.2 at destination port 23, we managed to catch both packets as they were sent to Ynet.co.il (one marked and the other is above it in gray).

Graphical user interface, application

Description automatically generated

packets coming from or to go a particular subnet – We decided to try and catch packets sent to the subnet 128.230.0.0, sent to host 128.230.0.10.

As seen in the picture, scapy managed to catch the packets to the host at the wanted subnet.

Text

Description automatically generated

**Task 1.2:**

We sent a packet from our vm (ip address: 10.0.2.4) faking a source ip of 10.9.0.5 to destination 10.9.0.1 and we can see in the picture below that the packet was received and the replay exiting from the host container (whose ip is 10.9.0.5) and entering the attacker container (whose ip is 10.9.0.1).

Text

Description automatically generated

**Task 1.3:**

We were asked to create traceroute.

We ended up answering this manually because we couldn’t fugure out how to capture the ICMP reply. However, also manually it works.

Graphical user interface, application

Description automatically generated

**Task 2.1A:**

**Q1 –** The first function called is pcap\_open\_live which is used to open a new session on a given interface in promiscuous mode.

Then we use pcap\_compile to convert the filter string to something the BPF can understand and use.

After that, we actually set the filter on the socket (on the interface through which we are listening for packets).

We then use pcap\_loop to listen to the socket for a set amount of time(depending on the input given to the function).

Pcap\_loop will call a function the we gave it, that will process the captured packets.

And at the end, pcap\_close will close the session we created.

**Q2 –** we need root privileges in order to run the sniffer program because the sniffer program uses raw sockets in order to capture packets and using raw sockets requires root privileges.

If we try and run the sniffer without root privileges, the program will fail at the pcap\_open\_live function because it won’t be able to o pen a raw socket.

**Q3 –**

**Task 2.1B:**