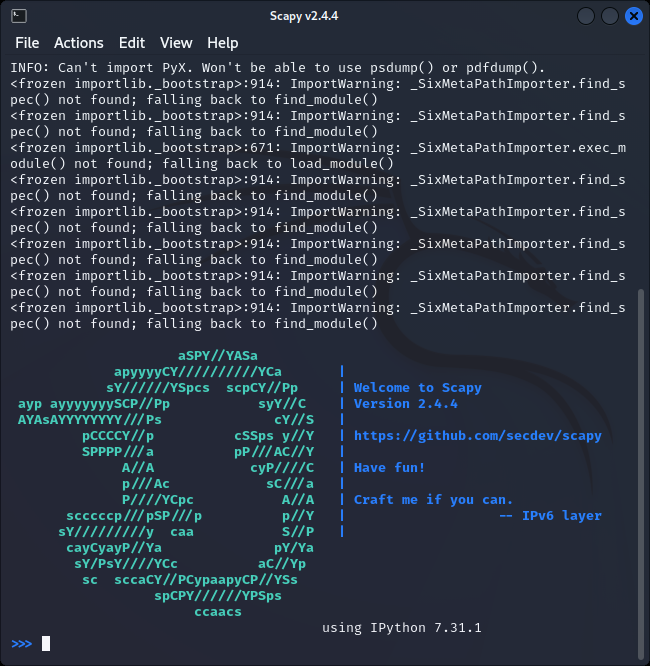
**LAB-04: ARP Cache Poisoning using Scapy**

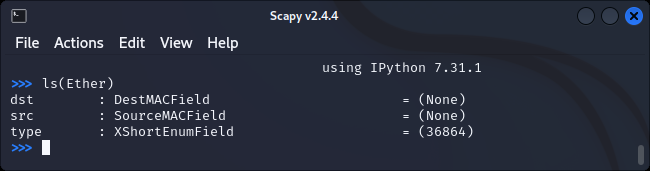
**3. Lab Task - 1**

**Part I: Introduction to Scapy**

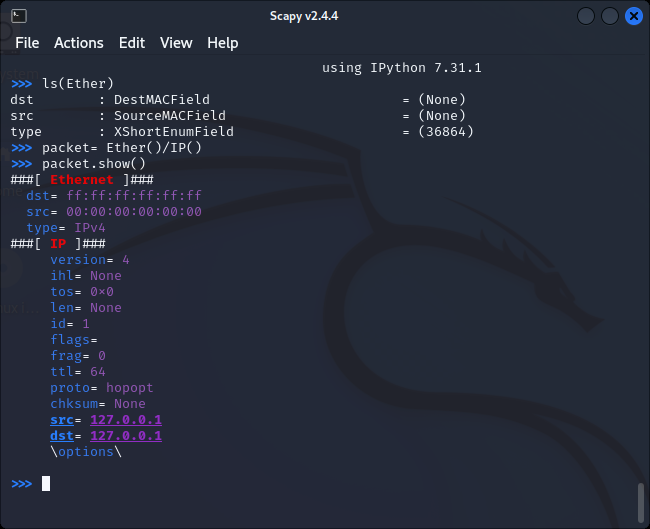
**Step 1:** Run scapy in Kali Linux by typing **scapy** in a terminal.



**Step 2:** To see what are the fields of a layer, use the **ls()** function.

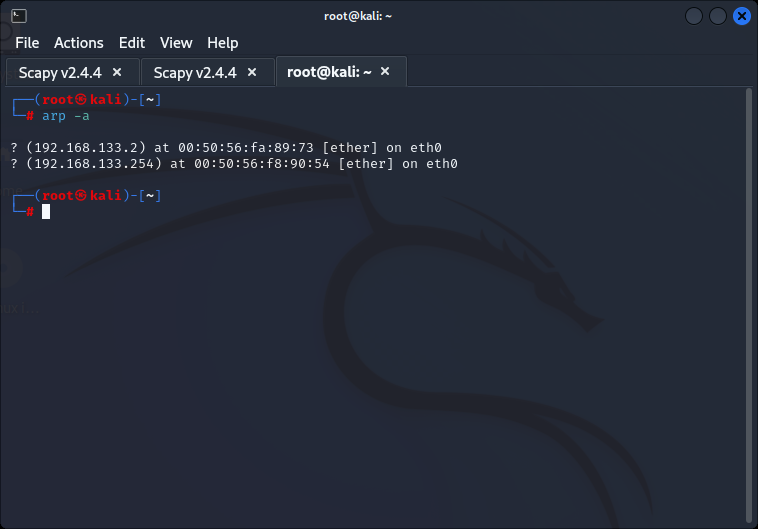


**Step 3:** To create a packet, append two layers together using the **“/” operator**.

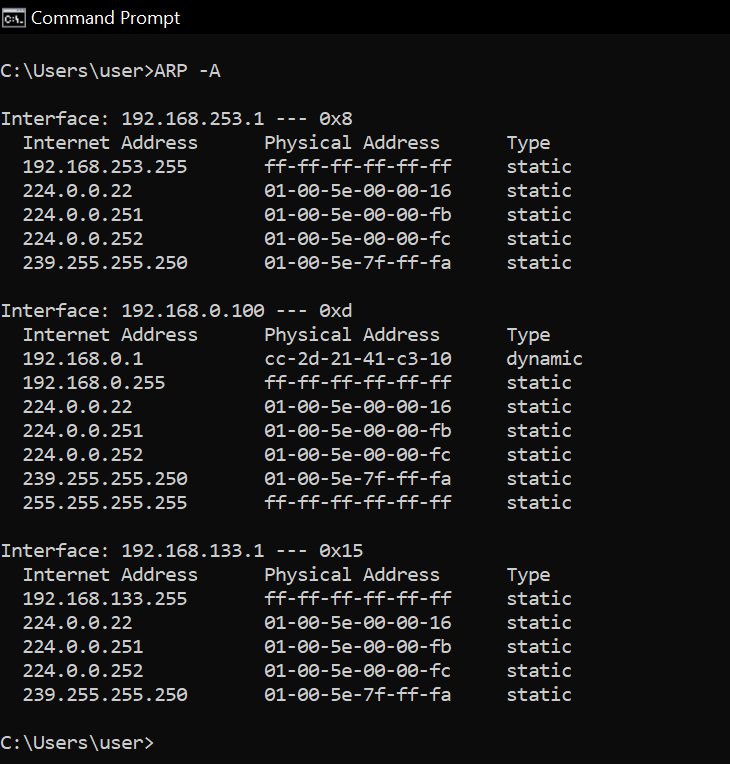


**Step 4:** To see the ARP table in both Windows and Linux, use the command **arp -a**.

In linux:



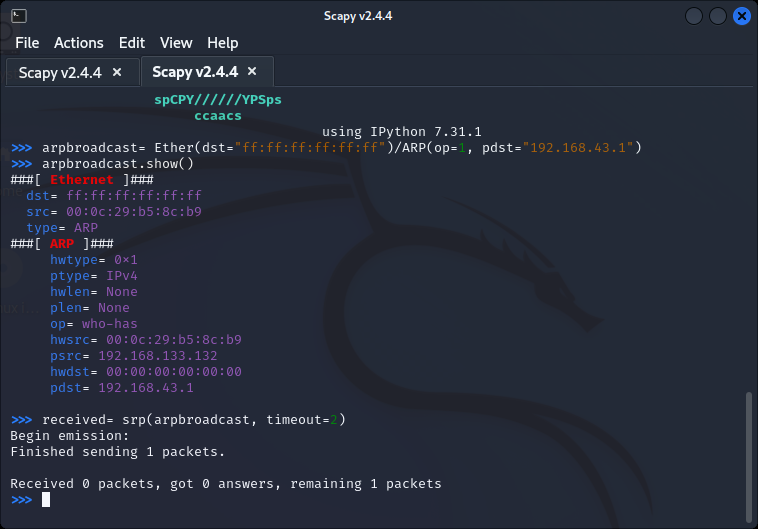
In windows:



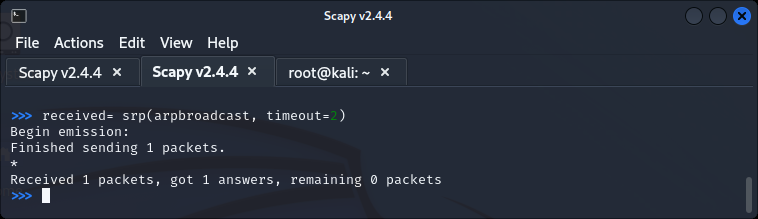
**Part II: Finding out the MAC address of the target and the Gateway**

**Step 1:** On a Kali Linux terminal run scapy.

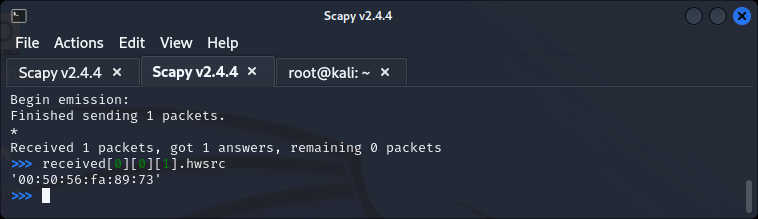
**Step 2:** ARP broadcast packet in Scapy by using the following commands:



**Step 3:** Use **srp()** to send the packet and then receive the response packet:

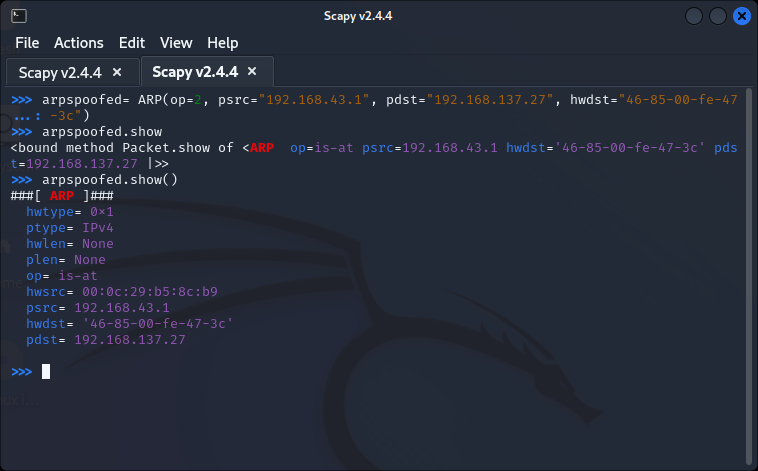


**Step 4:** Here is the MAC address:

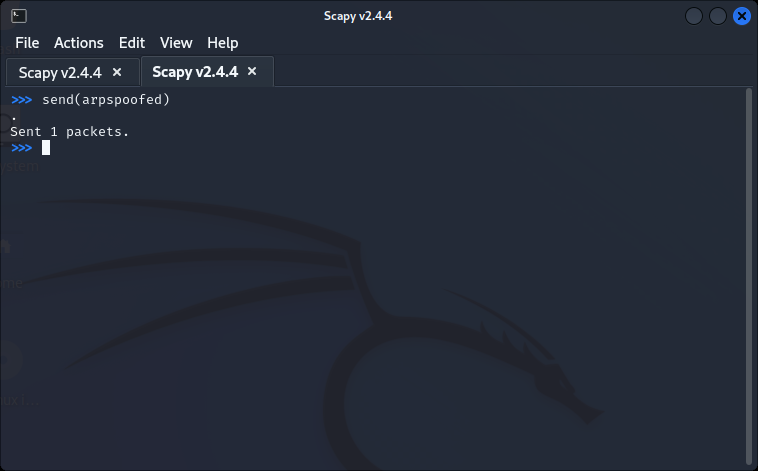


**Part III: Sending false ARP response packets to both the target and the gateway.**

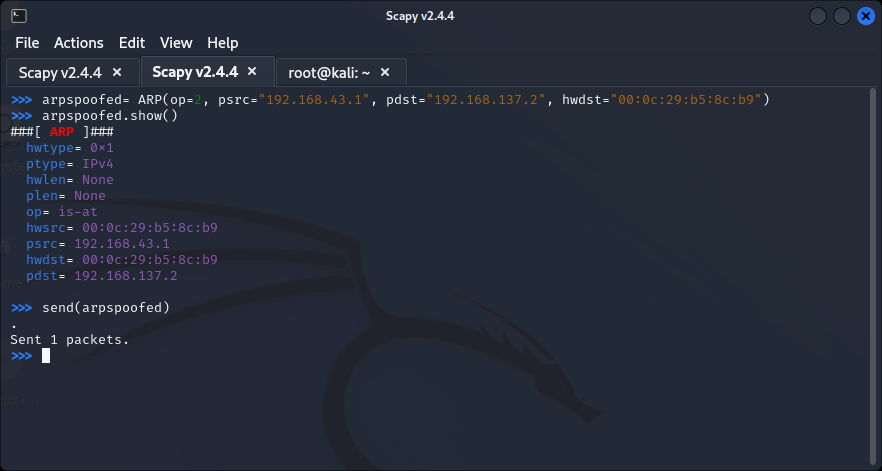
**Step 1:** Spoofed packet for 192.168.43.65:



**Step 2:** Send the spoofed packet, the response packet is not needed so use send():



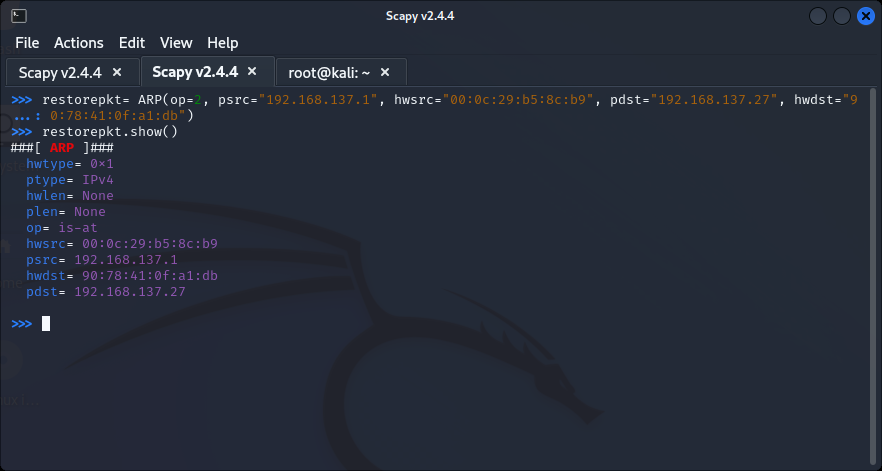
**Step 3:** Similarly, craft a packet for the gateway (192.165.43.1, 84:fd:d1:14:a6:9f ) by spoofing the psrc as “192.168.43.65”.



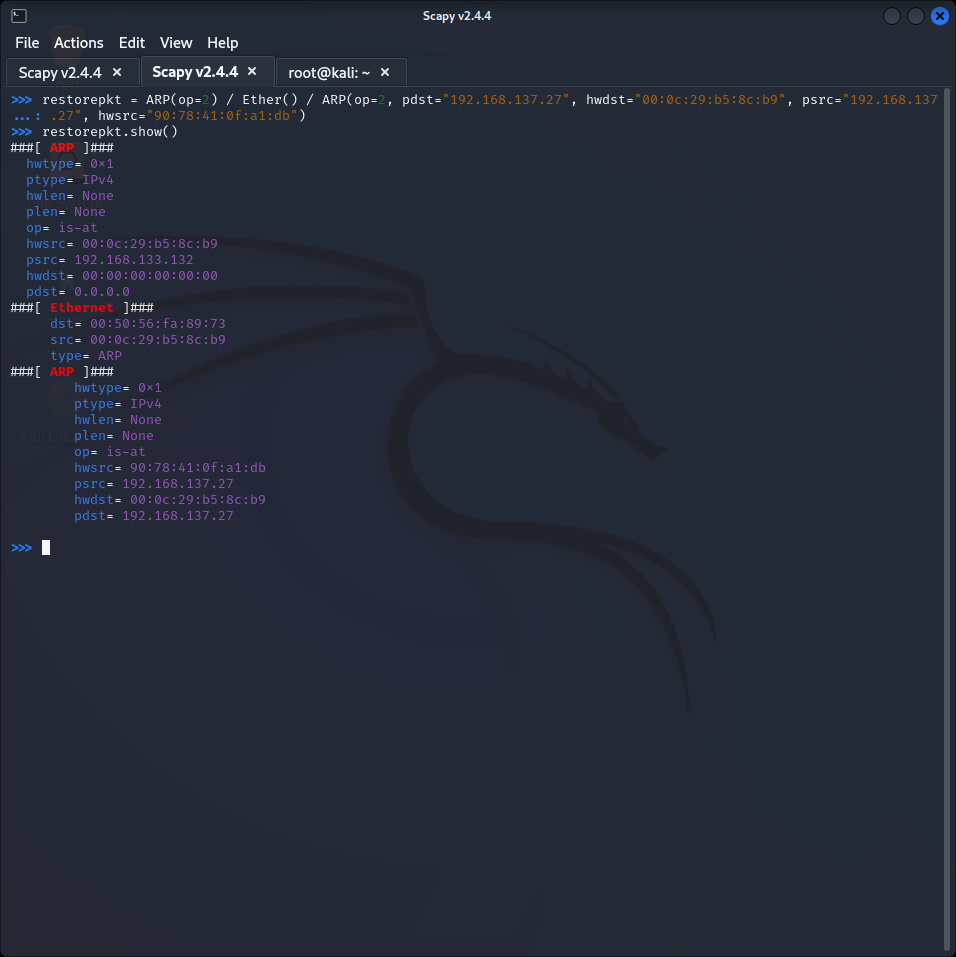
**Part III: Once the attack is done. Remember to restore the ARP tables of the machines.**

To restore the ARP tables, craft the packets that originally should have been used.

**Step 1:** From the gateway to the target:



**Step 2:** From the target to the gateway:

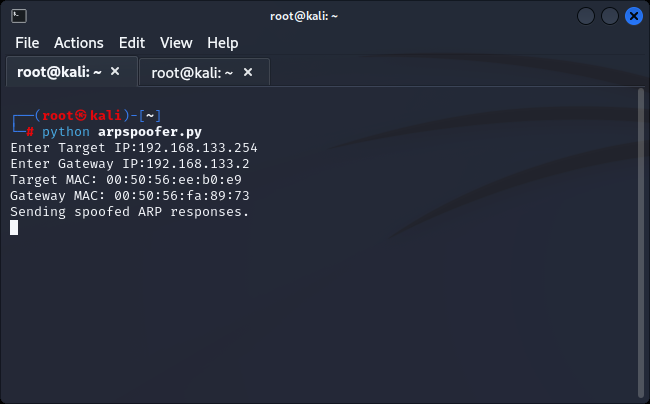


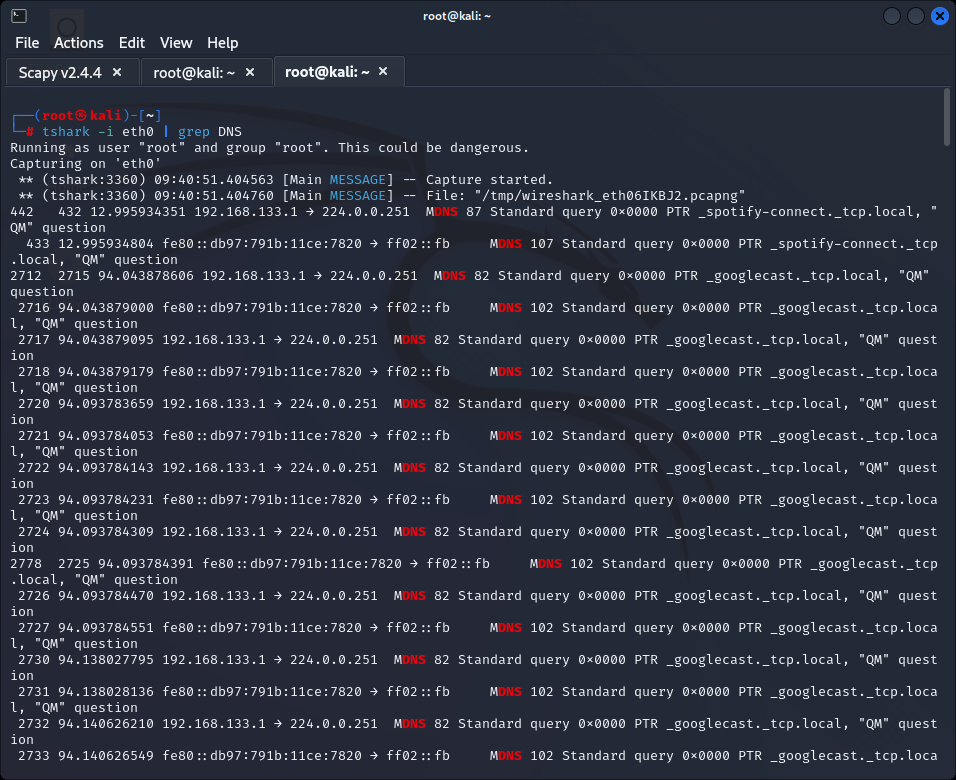
**Part IV: Automate the Whole Process Using Python Script**

**Step 1:** Create this python script:



**Step 2:** Run the script:

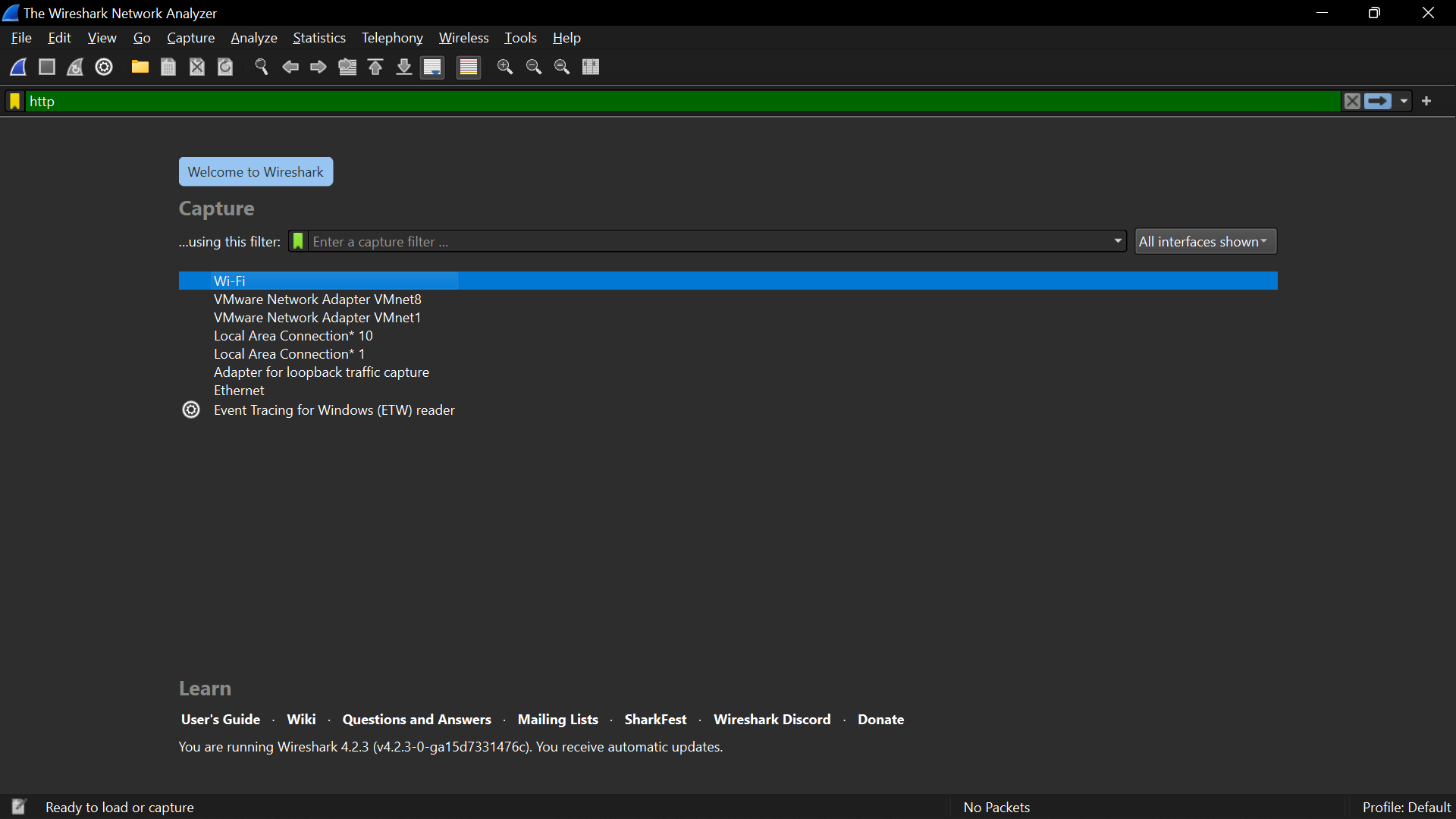


You will also notice that the MITM is also working, if you does **tshark -i eth0| grep DNS**, on the attacker's machine, one can see the DNS requests of the target going through.  


**4. Lab Task – 2: Packet sniffing with Wireshark**

13. Select your Network Interface Card if it is not already selected.

14. Take a screenshot.



15. Close *all* other programs you currently have open except your word processing program (Microsoft Word, LibreOffice Writer®, etc.).

16. Click Start.

17. Let it run for 10 seconds.

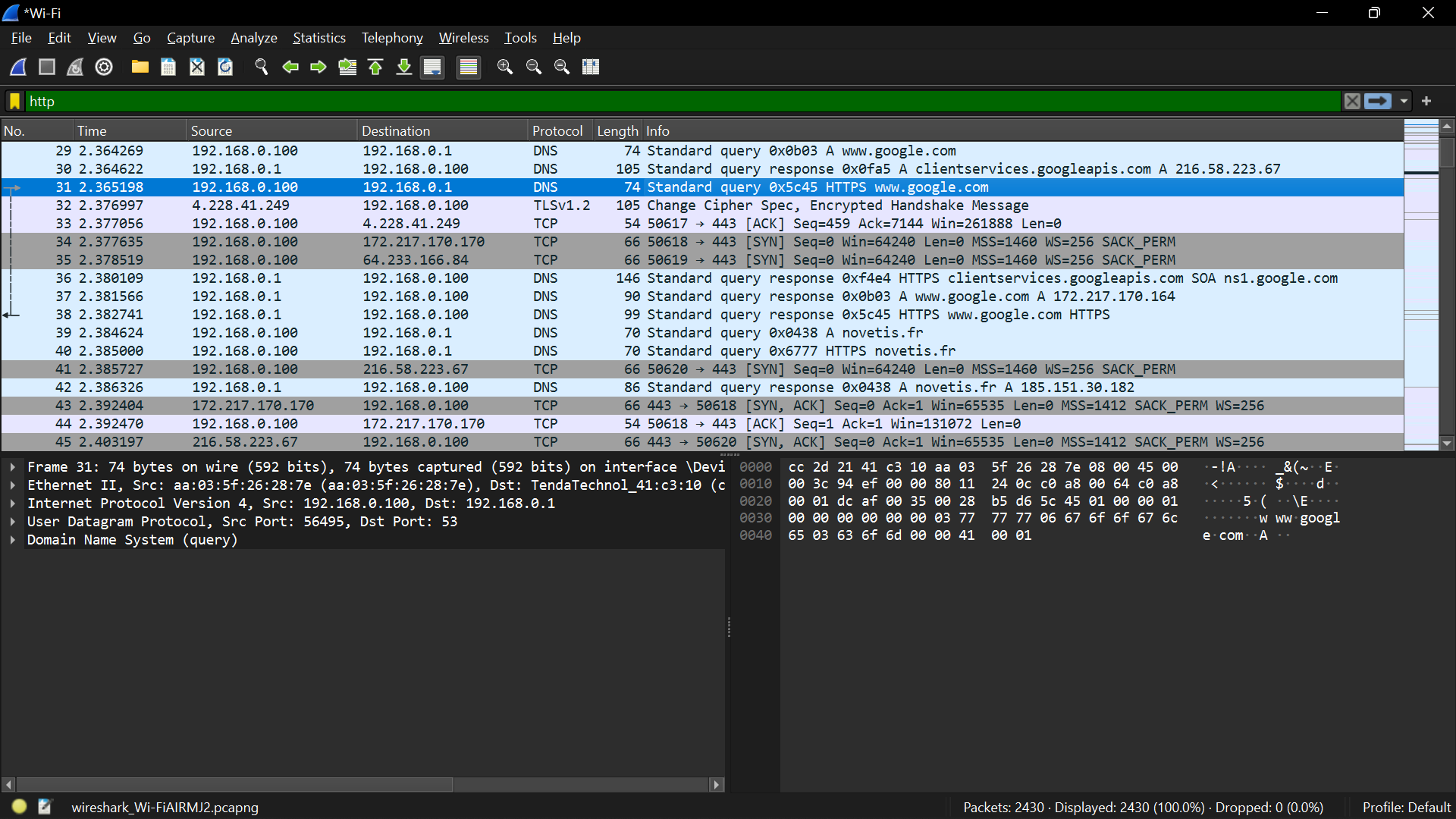
18. While you are waiting open a web browser and go to www.google.com.

19. Return to your Wireshark window.

20. In the file menu click Capture and Stop (or use the keyboard shortcut—Ctrl+E).

21. Scroll up until you see a green and blue area. (These are the packets you captured when you requested Google’s main page.)

22. Take a screenshot.



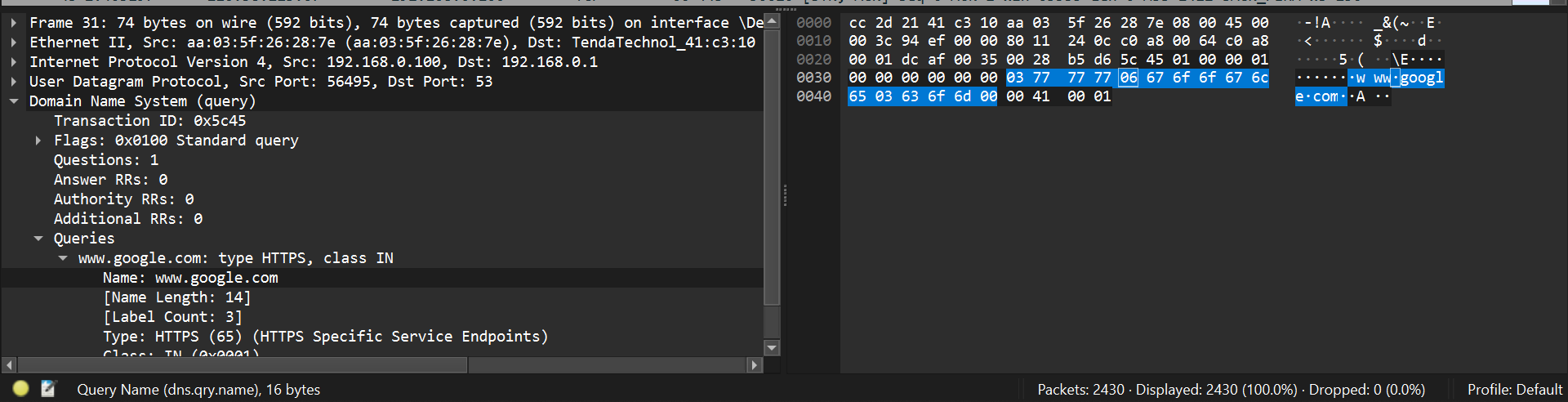
23. Scroll down until you see a line that has GET / HTTP/1.1. (You may have to try more than one until you get to the packet that shows “www.google.com” in the bottom pane.)

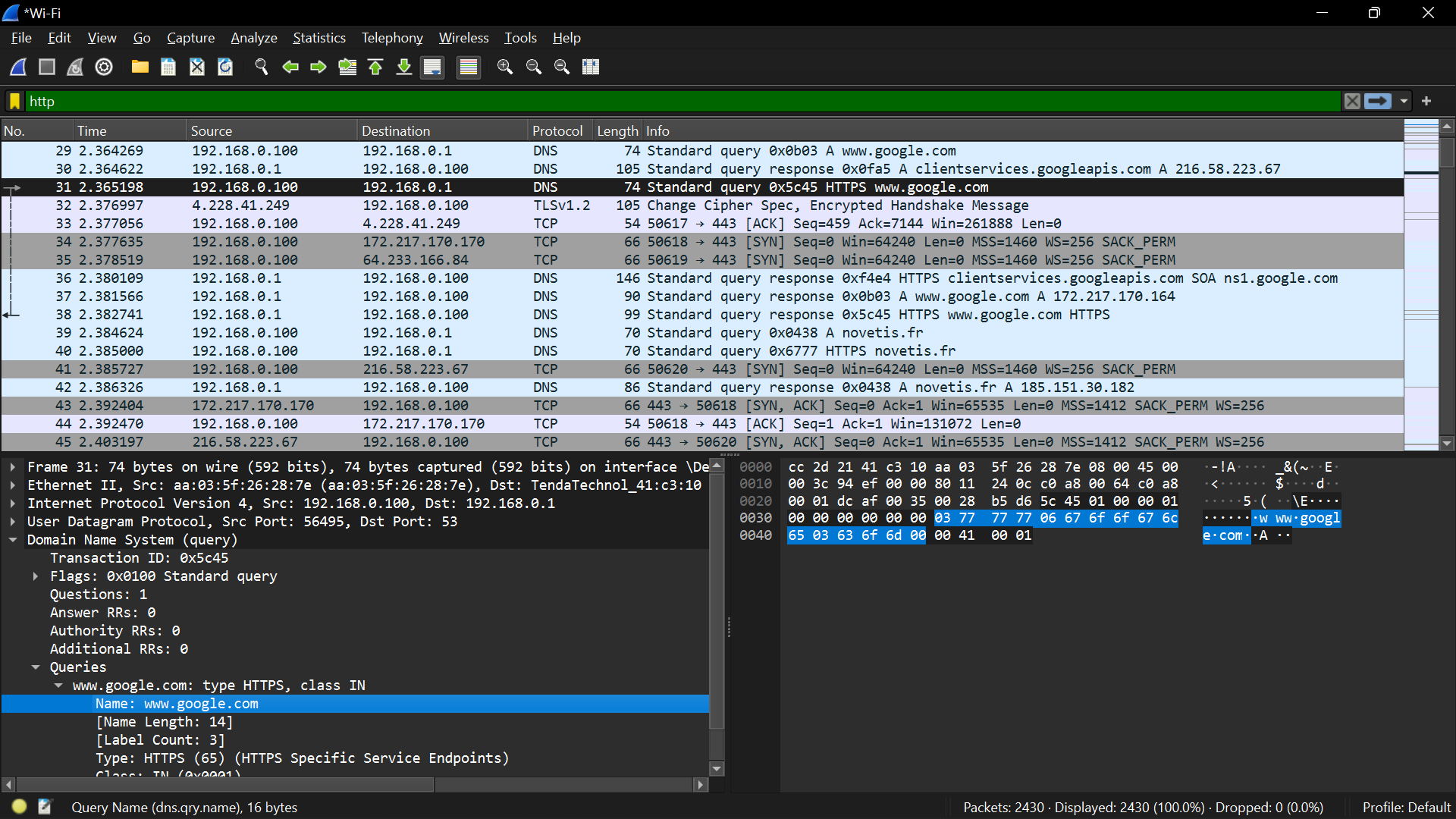
24. Select that row.

25. In the bottom pane, you will see a bunch of numbers to the left. (It’s the packets contents in hexadecimal.) Just to the right you will see the content of the packet in a column.

26. Select the text: www.google.com.

27. Take a screenshot.





**5. Lab Task – 3: THOUGHT QUESTIONS**

1. I believe my computer sends many packets instead of one big packet because breaking the data into smaller packets allows for more efficient transmission over the network. It helps with error handling, and smaller packets are routed more easily through the network, promoting fair resource sharing.
2. SYN (synchronize), ACK (acknowledge), FIN (finish), and GET have specific meanings in networking protocols. For instance, SYN initiates a connection, ACK acknowledges receipt, FIN signals the end of a connection, and GET is a request method used in HTTP.
3. I think packets have sequence numbers to aid in reordering and reassembling at the destination. They ensure that data arrives in the correct order, even if packets take different routes through the network.
4. My computer sends packets to the webserver to request and receive data in response to my web browsing activities. These packets contain information such as the request for a particular webpage or resource.
5. The different colors in Wireshark indicate various types of packets or their statuses. For example, green might represent TCP packets, while red might indicate errors or issues in the communication.
6. My computer might receive packets addressed to another computer due to network misconfigurations or errors. Routers might mistakenly forward packets to the wrong destination.
7. The number of packets sent/received in a single mouse click when visiting a website depends on various factors such as the complexity of the webpage, the number of resources it loads, and the underlying protocols used.
8. Yes, I can organize or filter traffic in Wireshark to make it easier to understand. I can filter packets based on protocols, IP addresses, ports, or other criteria.
9. Blocking all ICMP traffic could protect me by preventing certain types of network attacks, like ICMP-based ping floods. However, blocking ICMP entirely might affect legitimate network functionalities.
10. With Port 80 (HTTP) blocked, I could still access websites using other ports, such as Port 443 (HTTPS). Many websites today use HTTPS for secure communication.
11. Allowing incoming Port 443 (HTTPS) while blocking incoming Port 80 (HTTP) could be a security measure to enforce encrypted communication. Blocking Port 80 might be an attempt to avoid unencrypted communication.
12. I think malware could potentially rename itself to mimic legitimate processes or files, making it harder to detect through a firewall. This could work because firewalls often rely on recognizing known patterns or signatures to identify malicious activity. Renaming may help the malware evade detection temporarily.