# Task 5: Capture & Analyze Network Traffic Using Wireshark.

Intern Name: Jainam Priteshkumar Shah

Date: 11/08/2025

Internship Provider: Elevate Labs

## 1. Objective:

Capture live network packets and identify basic protocols and traffic types.

## 2. Tools Used:

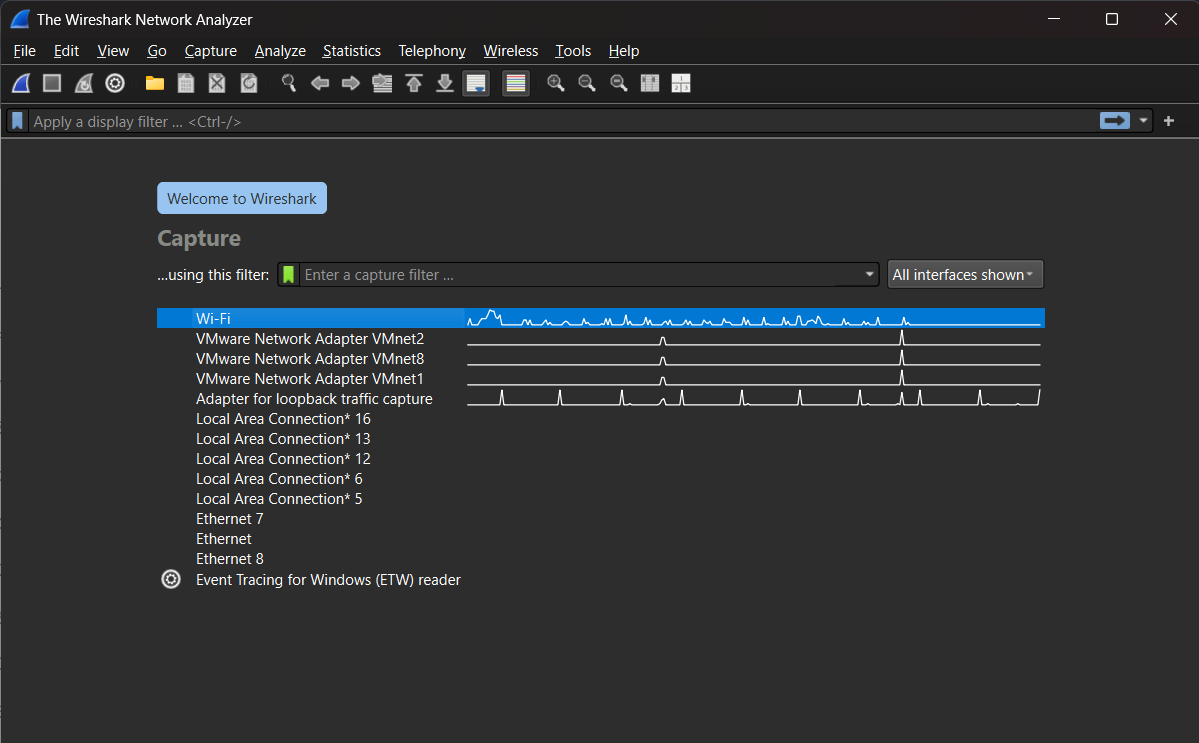
* **Wireshark:** A free and open-source packet analyzer used for network troubleshooting, analysis, software and communications protocol development, and education.

## 3. Procedural Steps:

This section details the step-by-step process to capture and analyze network packets.

### 3.1. Installation and Setup

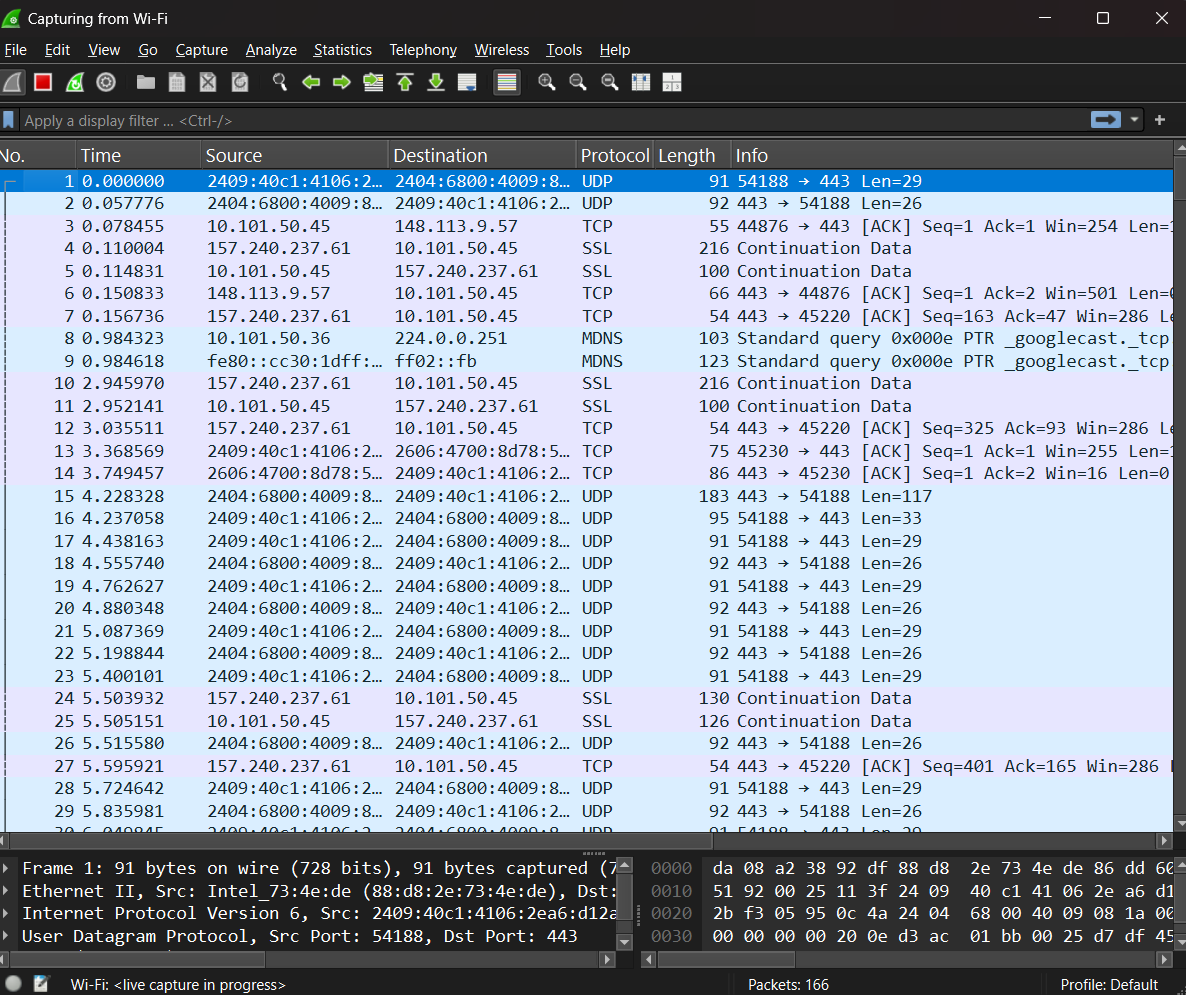
First, I downloaded and installed Wireshark from its official website. During the installation, I also installed the Npcap packet capture driver, which is required for Wireshark to capture live network data on Windows. After that, I launched Wireshark.



5.1 Wireshark Home Page

### 3.2. Starting the Packet Capture

Wireshark presented a list of available network interfaces on my computer. I identified my active network interface (Wi-Fi) by looking for a live traffic graph next to it. I double-clicked the interface name to start the capture process.

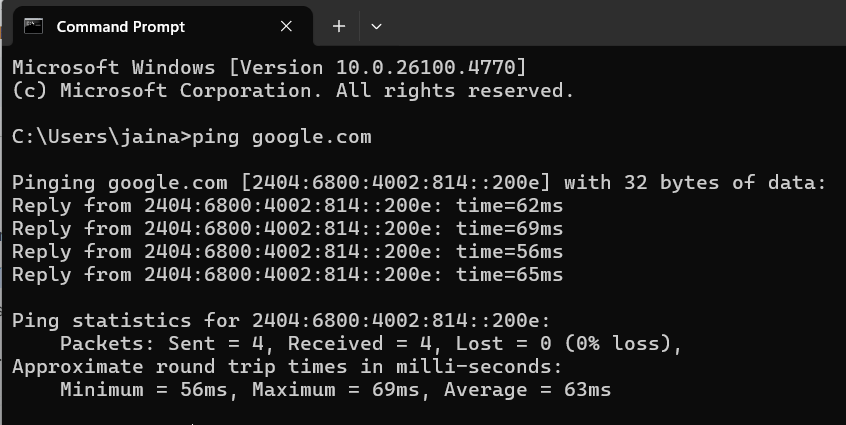


5.2 Capturing with Wi-Fi

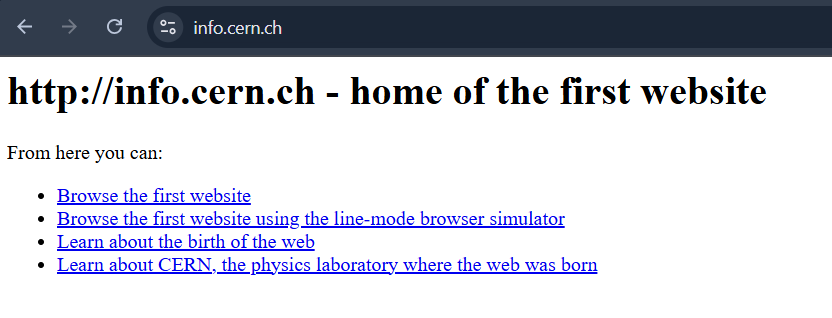
### 3.3. Generating Network Traffic

With the capture running, I generated some sample network traffic to analyze. To do this, I performed two simple actions:

1. Opened a Command Prompt and used the command ping google.com.
2. Opened a web browser and navigated to a non-encrypted website like http://info.cern.ch to ensure I could see unencrypted HTTP traffic.



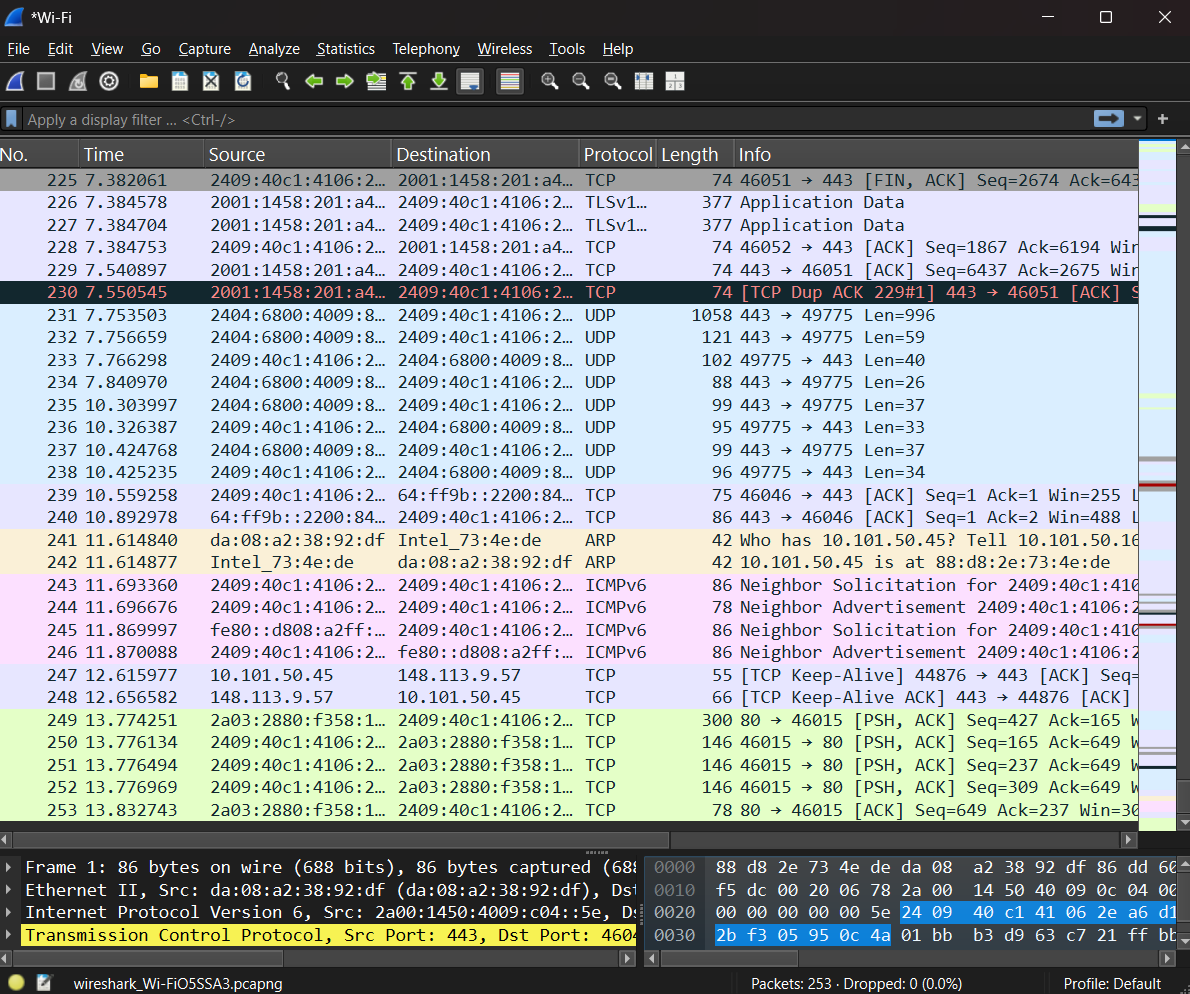
5.3 Ping to google.com



5.4 Non-encrypted Website (http://info.cern.ch)

### 3.4. Stopping the Capture and Initial Observation

After about a minute of generating traffic, I stopped the packet capture by clicking the red square "Stop" button in the Wireshark toolbar. The screen was immediately filled with hundreds of captured packets of various protocols and colors.



5.5 Captured Data

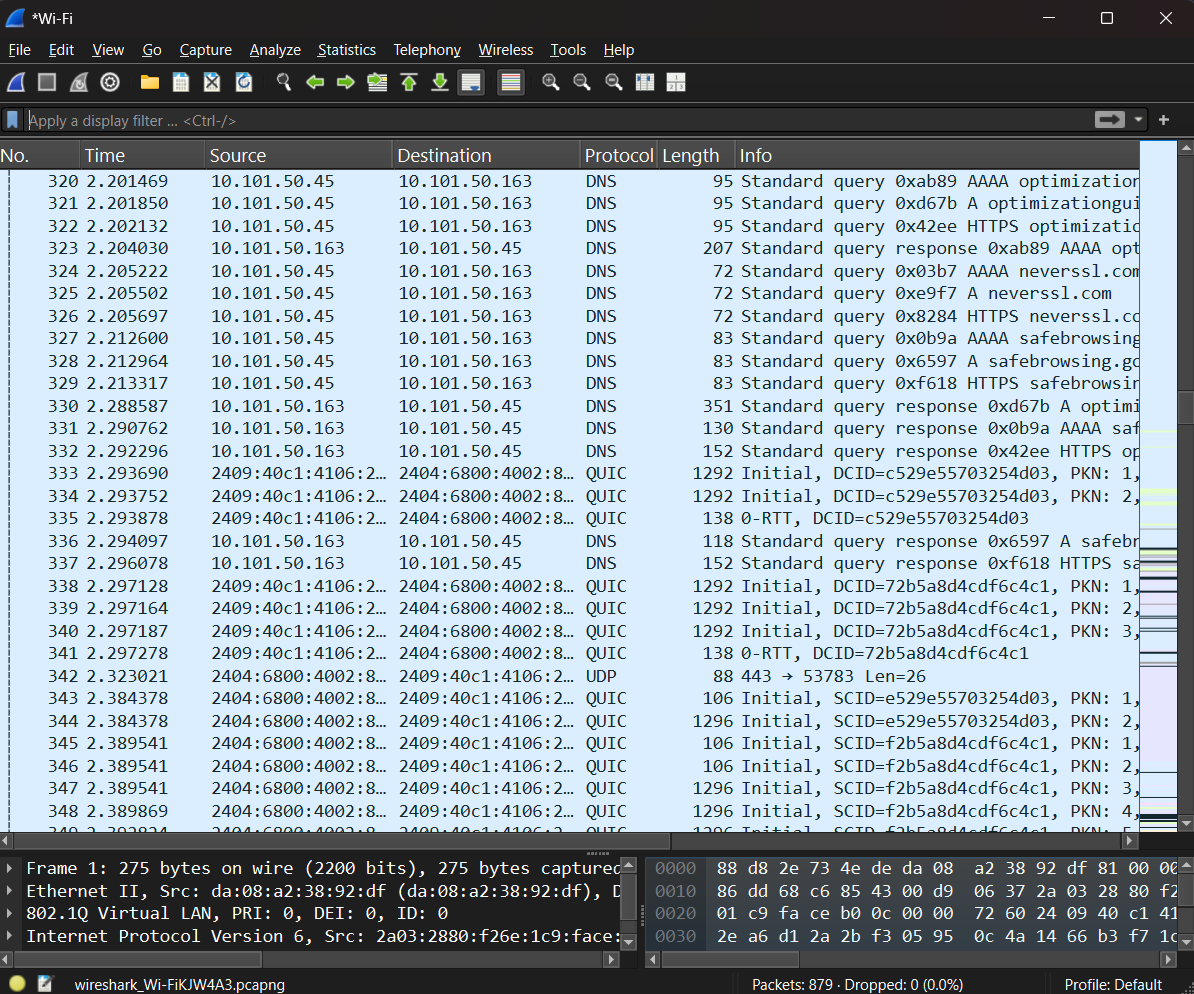
### 3.5. Filtering and Analyzing Protocols

To make sense of the data, I used Wireshark's display filter bar to isolate specific protocols.

#### Protocol 1: DNS (Domain Name System)

I wanted to see how my computer found the IP address for google.com.

* **Filter:** I typed dns in the filter bar and pressed Enter.
* **Analysis:** I observed a "Standard query" packet from my computer's IP to the DNS server, asking "who is google.com?". This was immediately followed by a "Standard query response" from the DNS server providing the IP address for google.com. This showed me the fundamental process of domain name resolution.

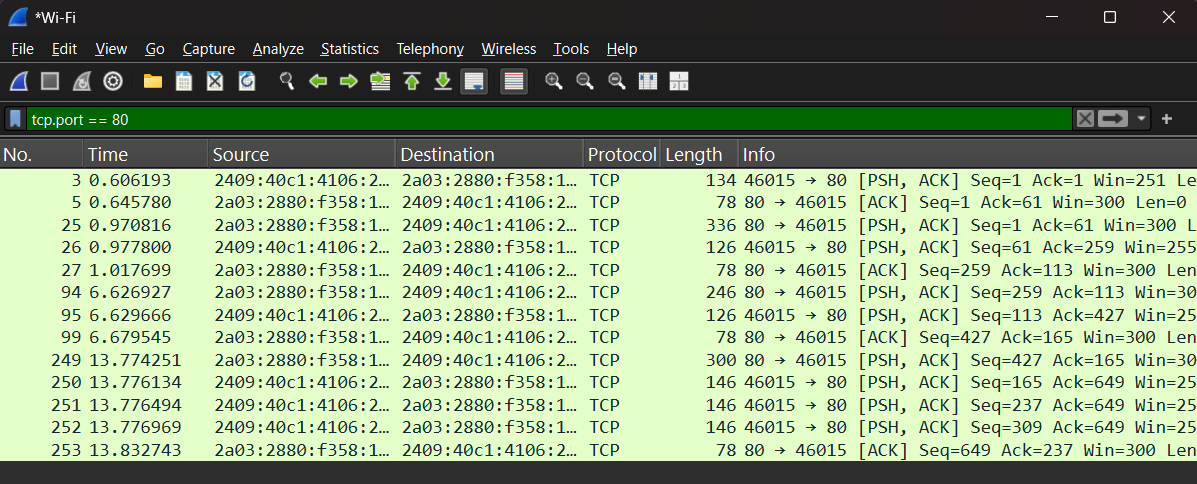


5.6 using “dns” filter

#### Protocol 2: TCP (Transmission Control Protocol)

I wanted to see the connection setup for my web browsing.

* **Filter:** I typed tcp.port == 80 to filter for traffic to the HTTP port.
* **Analysis:** I located the famous "three-way handshake." I could see the sequence clearly:
  1. [SYN] - My computer sent a Synchronize packet to the web server.
  2. [SYN, ACK] - The server replied with a Synchronize-Acknowledge packet.
  3. [ACK] - My computer sent a final Acknowledge packet.  
     This confirmed that a reliable connection was established before any data was sent.

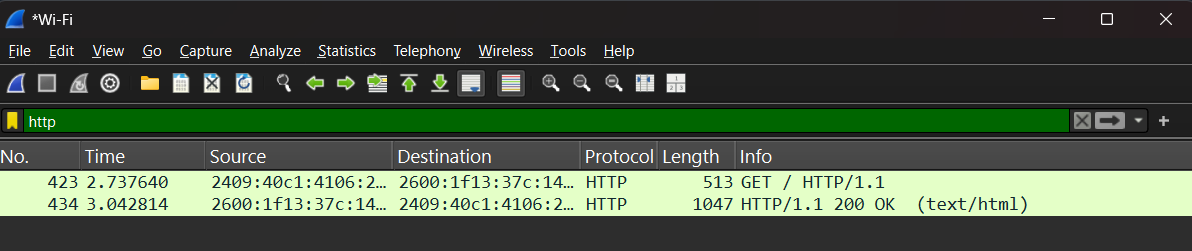


5.7 using “tcp.port==80” filter

#### Protocol 3: HTTP (Hypertext Transfer Protocol)

I wanted to see the unencrypted web traffic itself.

* **Filter:** I typed http in the filter bar.
* **Analysis:** I saw a packet with "GET / HTTP/1.1" in the info column. This was my browser requesting the main page of the website. Following this, I saw packets with "HTTP/1.1 200 OK," which contained the actual HTML code of the website. I could even see the plain text HTML in the packet details pane, which demonstrated why HTTPS is so important.



5.8 using “http” filter

### 3.6. Exporting the Capture File

As required by the deliverables, I saved the entire packet capture by going to **File > Save As...** and saving the file as Task-5.pcapng.