



**Lab Number: 14**

**Date: 2025/08/30**

## **Title: Introduction to Network Traffic Analysis using Wireshark**

---

### **Theory:**

**a. Wireshark:** Wireshark is powerful, open-source network protocol analyzer that allows users to capture and inspect the data traveling across a network in real-time. It provides detailed insights into packet-level information, making it an essential tool for network troubleshooting and performance monitoring. Wireshark helps identify issues such as network congestion, security vulnerabilities, misconfigurations, and protocol errors. By allowing users to visualize traffic patterns and analyze the behavior of network protocols, it aids in diagnosing problems efficiently and ensuring optimal network performance. Its ability to capture and decode a wide range of protocols makes it invaluable in both network administration and cybersecurity.

### **b. Key Concepts of Wireshark**

**Packets:** The smallest unit of data transmitted over a network, which Wireshark captures for analysis.

**Protocols:** Rules governing data transmission, such as TCP, UDP, HTTP, etc., that Wireshark can decode and display

**Capture Filters:** Filters used to specify which types of packets should be captured during network traffic analysis.

**Display Filters:** Filters applied after capturing traffic to narrow down and view specific packets based on criteria like IP addresses or protocols.

**Packet Details:** A breakdown of each captured packet, showing the different layers (e.g., Ethernet, IP, TCP/UDP) of the network protocol stack.

**Real-time Monitoring:** Allows users to capture and view network traffic in real-time for immediate troubleshooting and analysis.

### **Interface of Wireshark**

#### **Main Toolbar**

Provides quick access to essential functions like starting/stopping captures, opening and saving files, and setting preferences. It streamlines navigation within Wireshark.

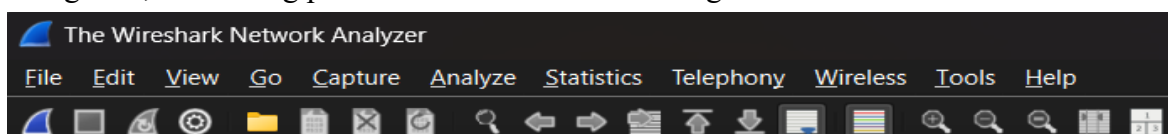


fig: Main toolbar of Wireshark

### Packet List Pane

Displays all captured packets in a list format, showing key information like packet number, timestamp, source/destination IP, protocol, and length.

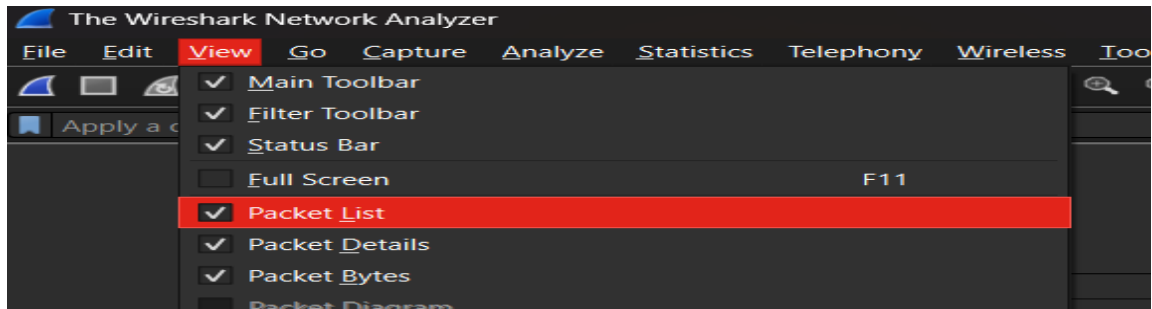


Fig: Packet List Pane of Wireshark

### Packet Details Pane

Shows a detailed breakdown of the selected packet, displaying information layer-by-layer (e.g., Ethernet, IP, TCP) to help analyze the packet structure.

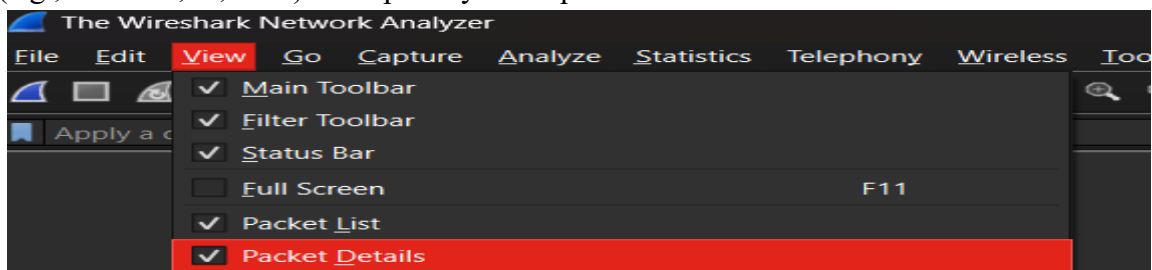


Fig: Packet Details of Wireshark

### Packet Bytes Pane

Displays the raw data of the selected packet in hexadecimal and ASCII formats, allowing for a deeper look into the packet's actual content.

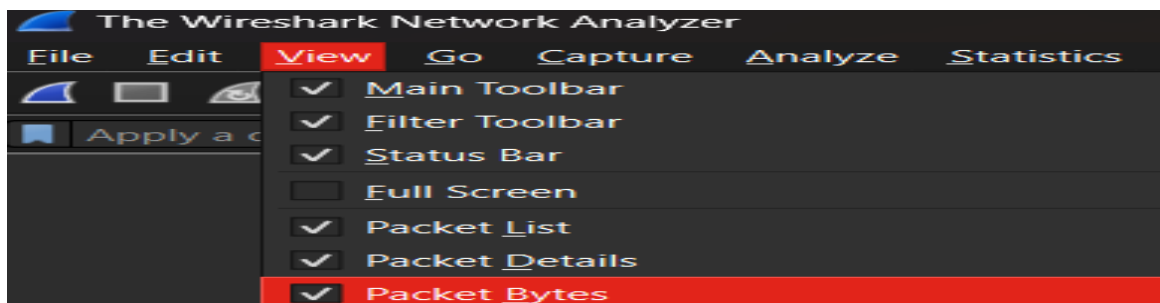


Fig: Packet Bytes Pane

## Basic Network Capture and Analysis

### Selecting a Network Interface

Step 1: Open Wireshark and click the “Capture” menu.

Step 2: Choose the correct network interface from the list (e.g., Ethernet or Wi-Fi).

Step 3: Look for the interface with active traffic (represented by a graph with fluctuating data).

Step 4: Choose the required interface by clicking on it.

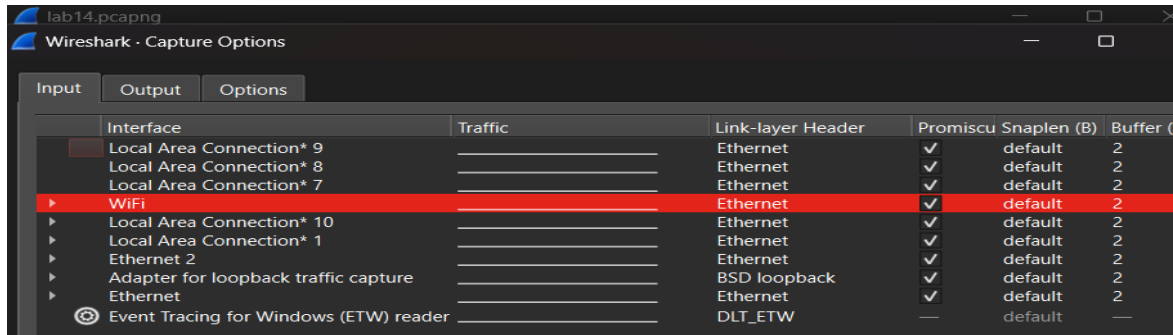


Fig: Choosing the Required Interface

## Starting Packet Capture

Step 1: Click the blue “Shark Fin” icon (Start Capture) on the toolbar to begin capturing.

Step 2: Open a web browser and navigate to “example.com” to generate traffic.

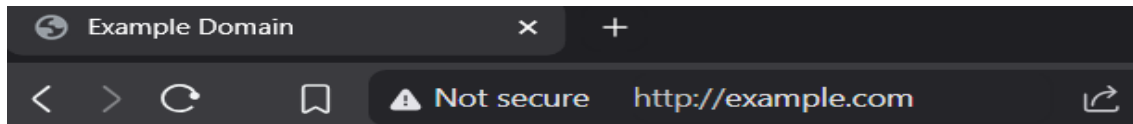


Fig: Navigate example.com in web browser

Step 4: With the use of filter bar packets related to example.com can be captured also type ip.addr = <ip address of src>

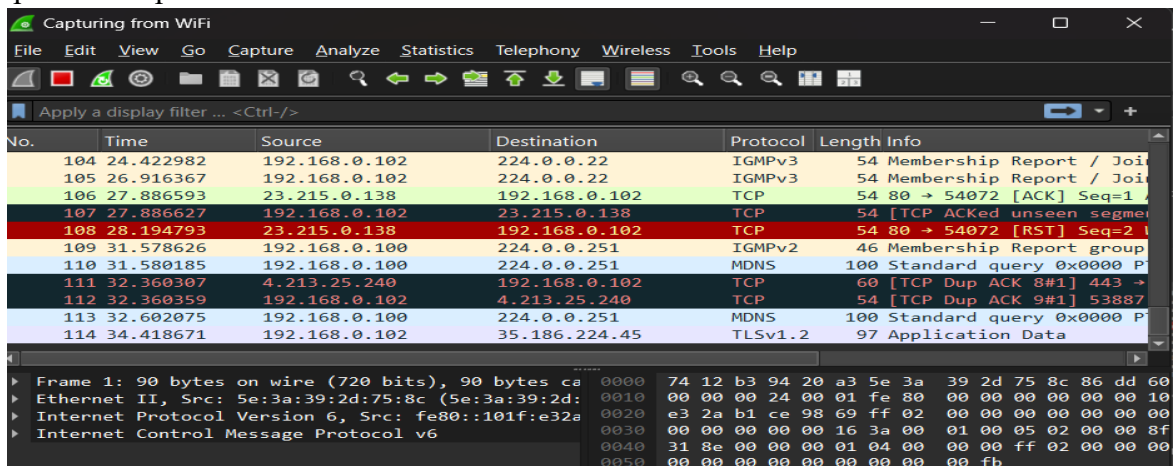


Fig: Capturing Packet

## Stopping and Saving Captures

Step 1: Click the red “Square” (Stop capture) on the toolbar when you have enough data.

Step 2: Go to “File” and select “Save As” to save the captured data in your preferred location.

Step 3: Choose the appropriate format (e.g., .pcap or .pcapng) for saving the file.

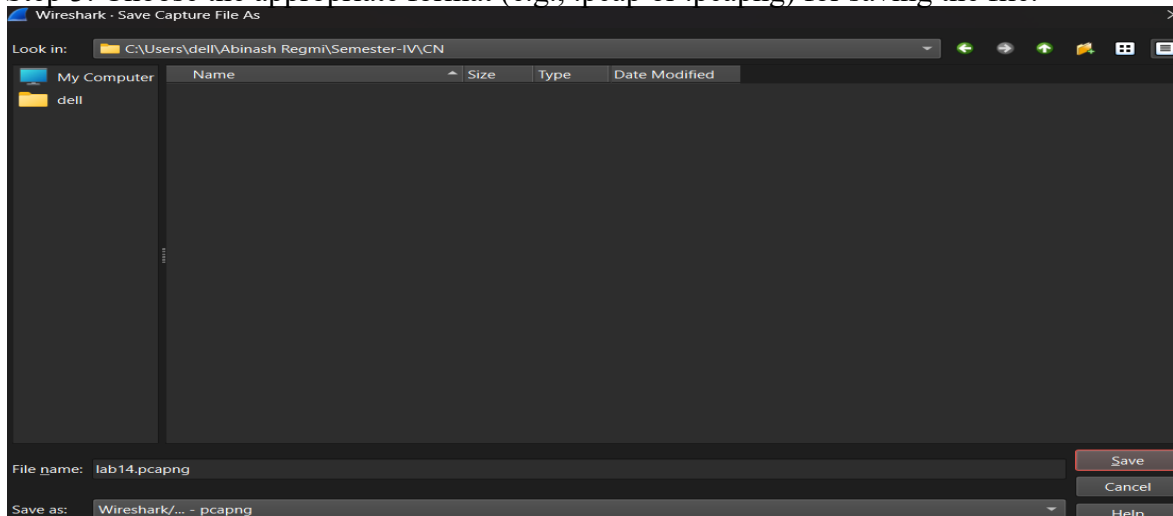


Fig: Saving a file

### Export the Captured Data

Step 1: After stopping the capture, click on “File” and choose “Export Specified Packets”.

Step 2: Apply any filters if you only want to export specific packets (e.g., by IP address or protocol).

Step 3: Save the exported file in the destined format for further analysis or sharing with others.

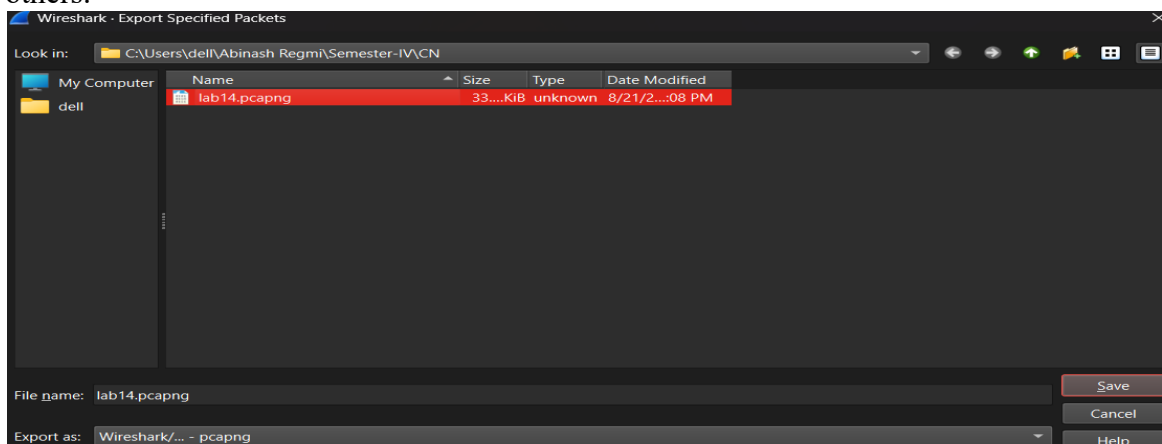


Fig: Exporting Capture Packet

### Conclusion

In this lab, we used Wireshark to capture, analyze, and export network packets, demonstrating its real-time traffic monitoring capabilities. By selecting a network interface and using example.co as a test site, we highlighted how Wireshark helps troubleshoot network issues and perform security audits. Its ability to decode protocols and visualize packet data makes it a vital tool for network administrators to ensure performance and security.