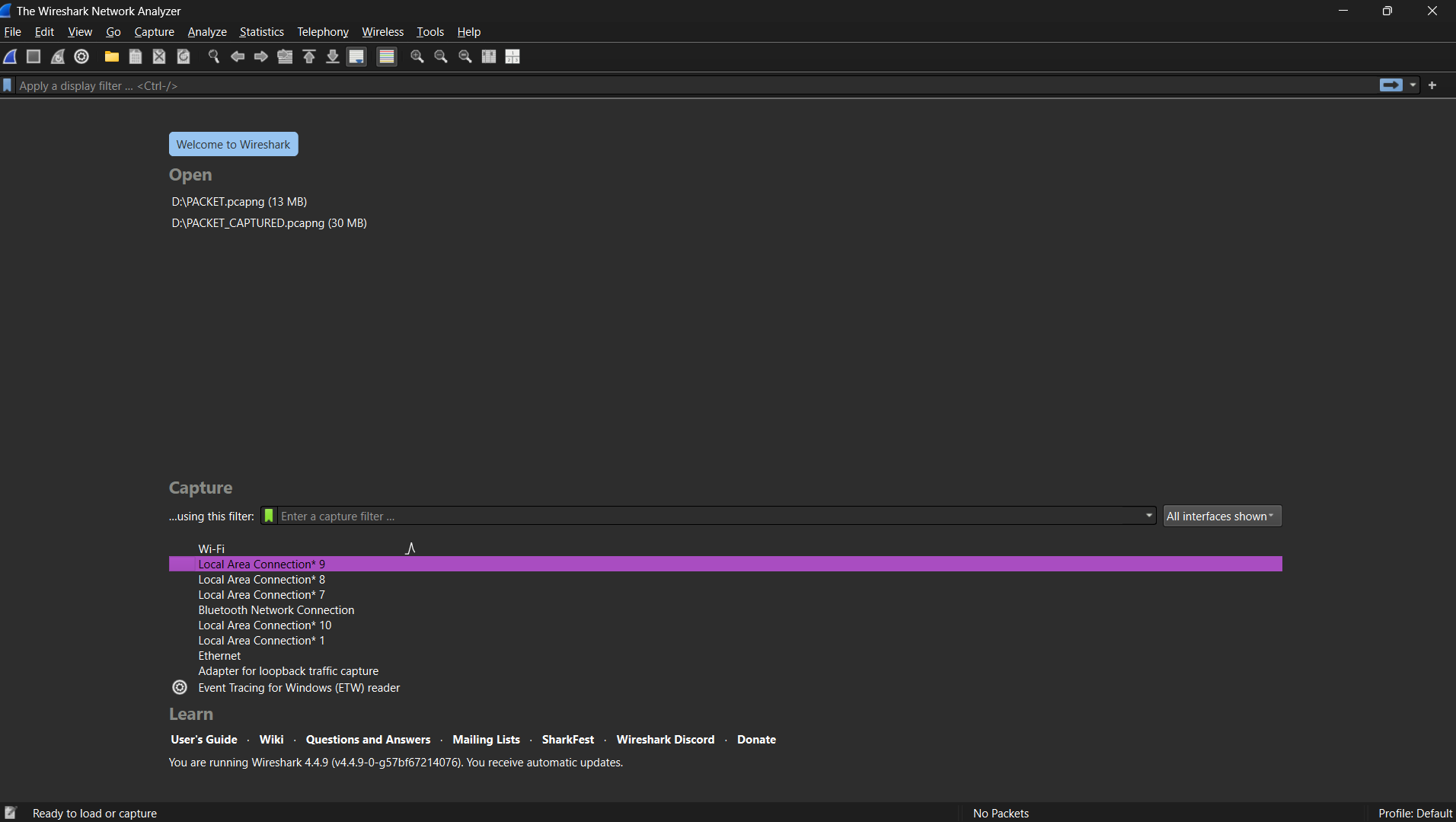
**Mini Project 1: Packet Analysis Tool**

**Objective:** Use a packet analysis tool (Wireshark) to monitor and analyze network traffic for potential security threats.

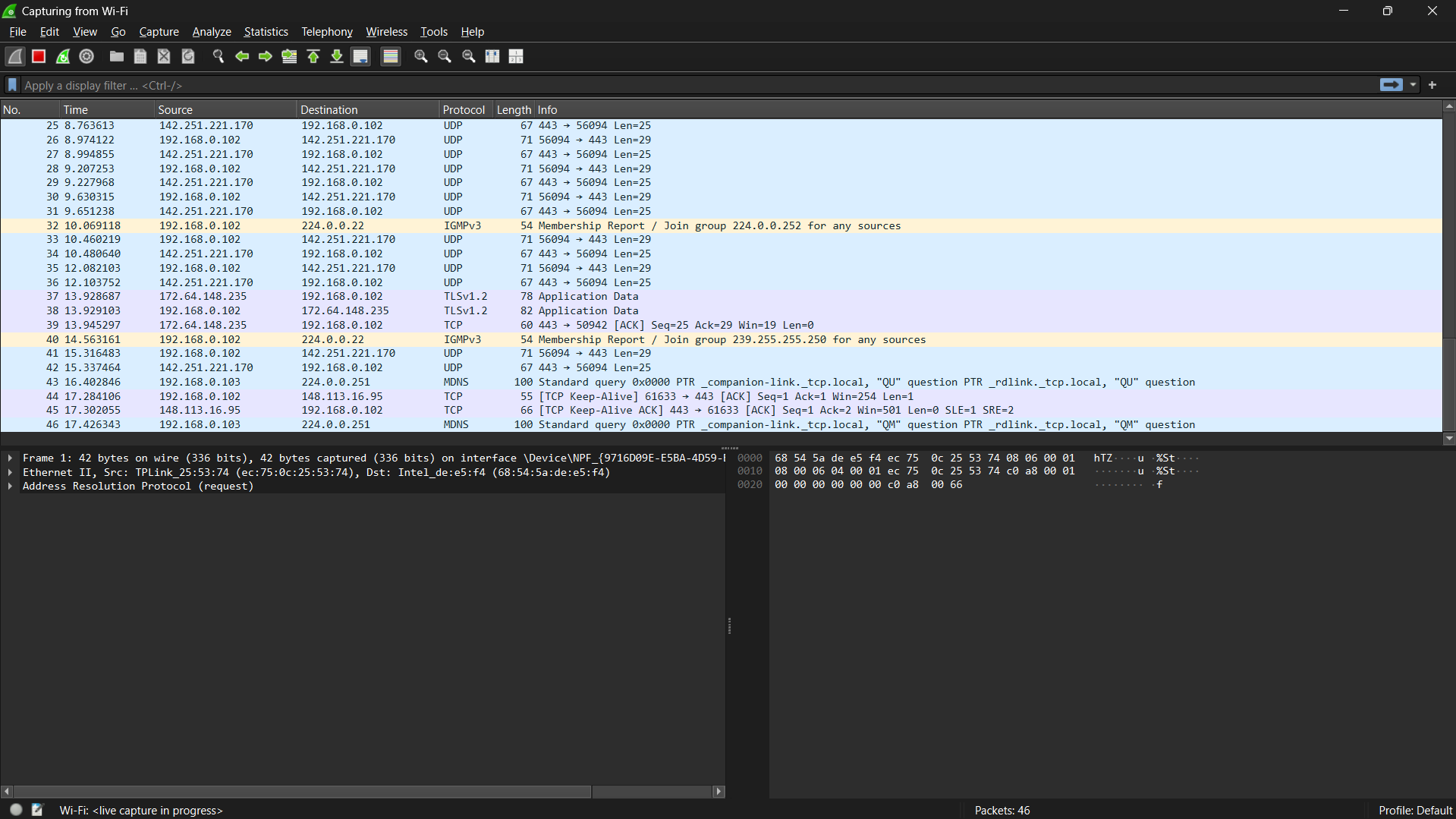
**Task 1: Packet Capture (Using Wireshark)**

Wireshark is a powerful tool that allows you to capture live network data and save it for analysis.

1. **Open Wireshark:** Launch Wireshark from the applications menu in your Kali Linux VM. You may need to run it with sudo wireshark from the terminal.
2. **Select an Interface:** You will see a list of network interfaces (e.g., eth0, ens33). Double-click the one that is active and showing traffic.



1. **Start Capturing:** The capture will begin immediately. You'll see packets scrolling in real-time.

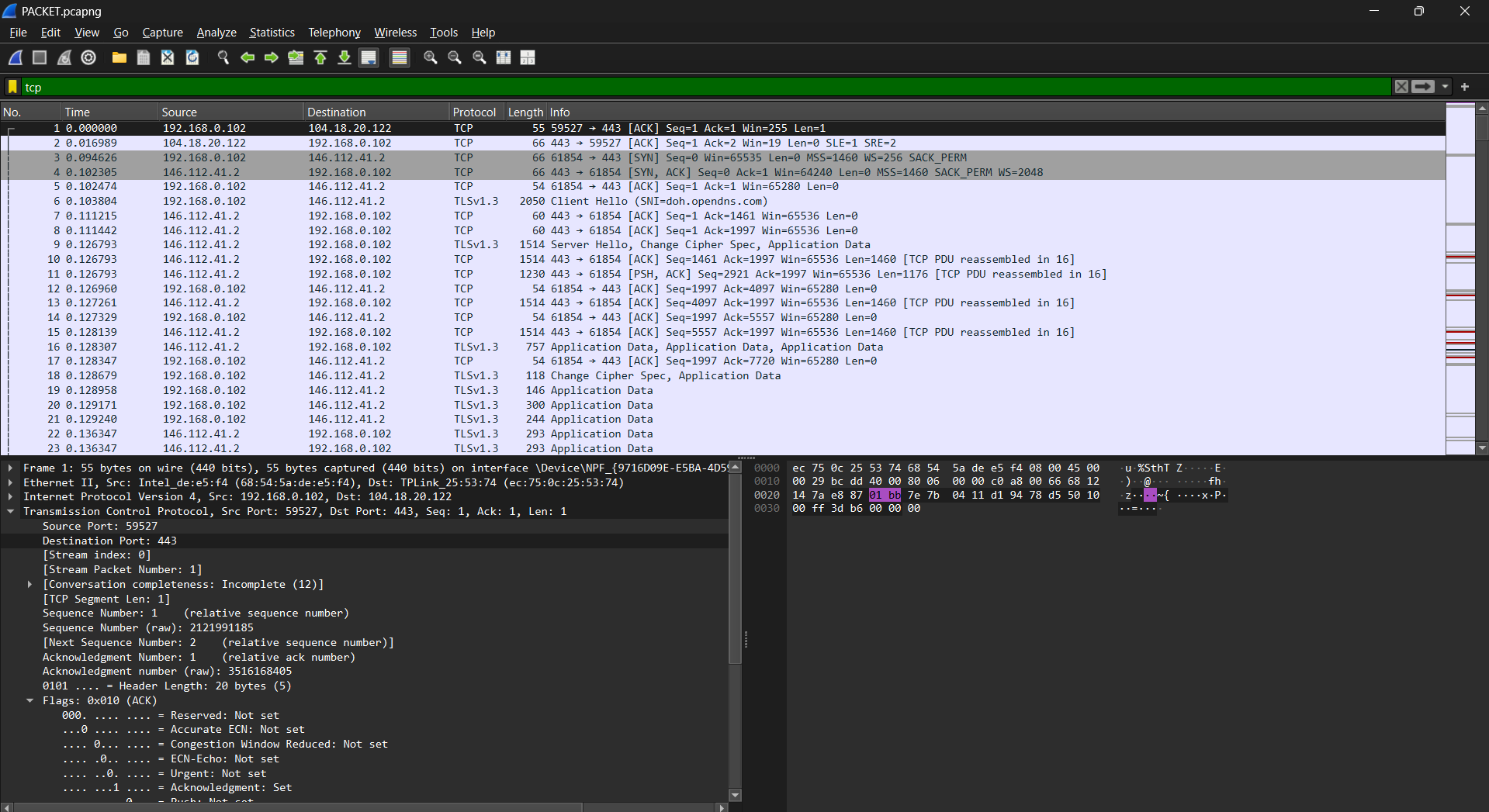


1. **Generate Traffic:** While the capture is running, perform some network activity, like browsing a website or logging into a service on your computer.
2. **Stop Capturing:** Click the red square "Stop" button in the toolbar when you have collected enough data.
3. **Save the File:** Go to **File > Save As...** to save your captured packets as a .pcapng file for your records.

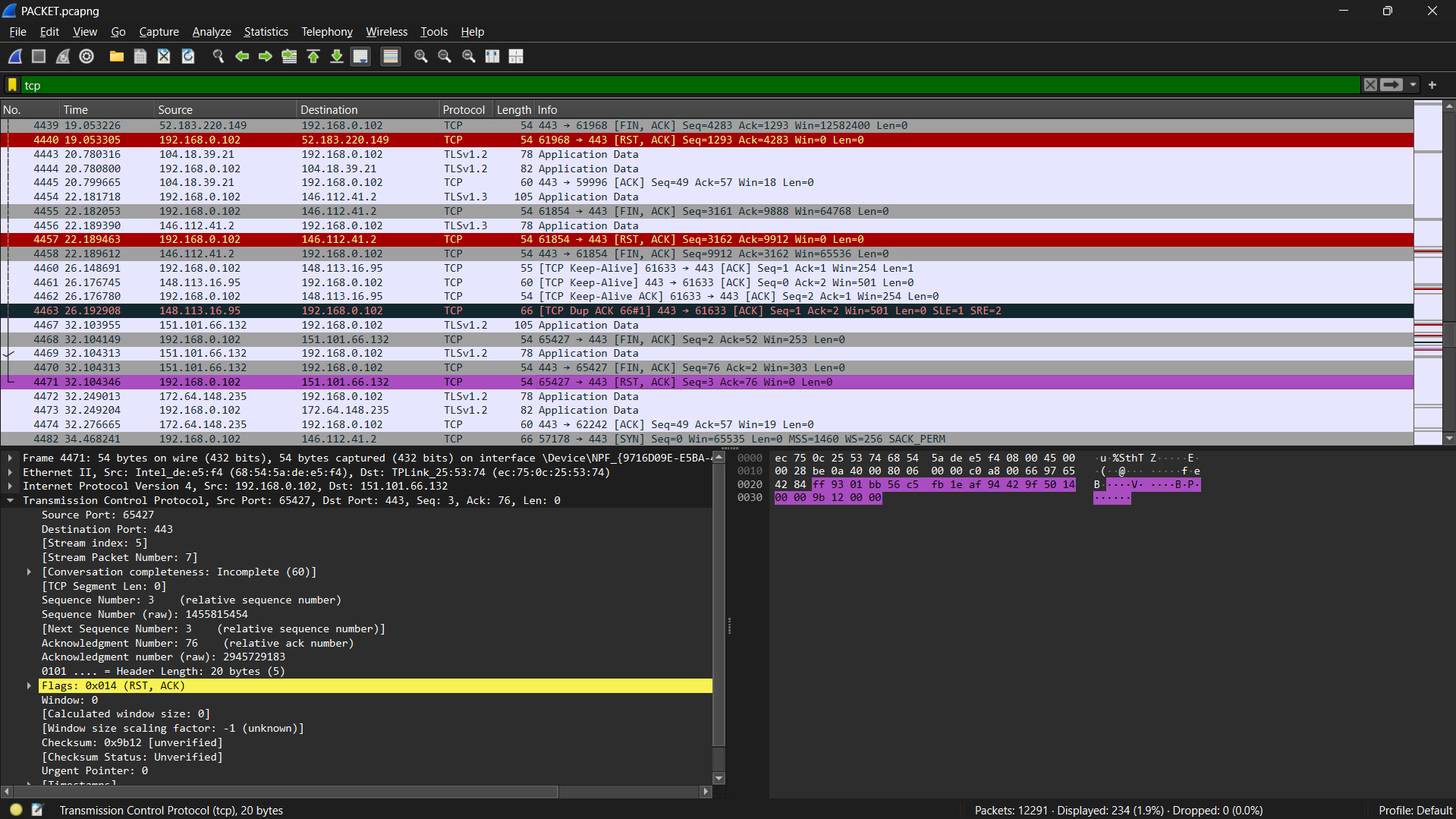
**Task 2: Packet Analysis (Using Wireshark)**

This analysis is based on the screenshots you provided, which serve as a perfect example of what to look for. The key to analysis in Wireshark is using **display filters** to isolate interesting traffic.

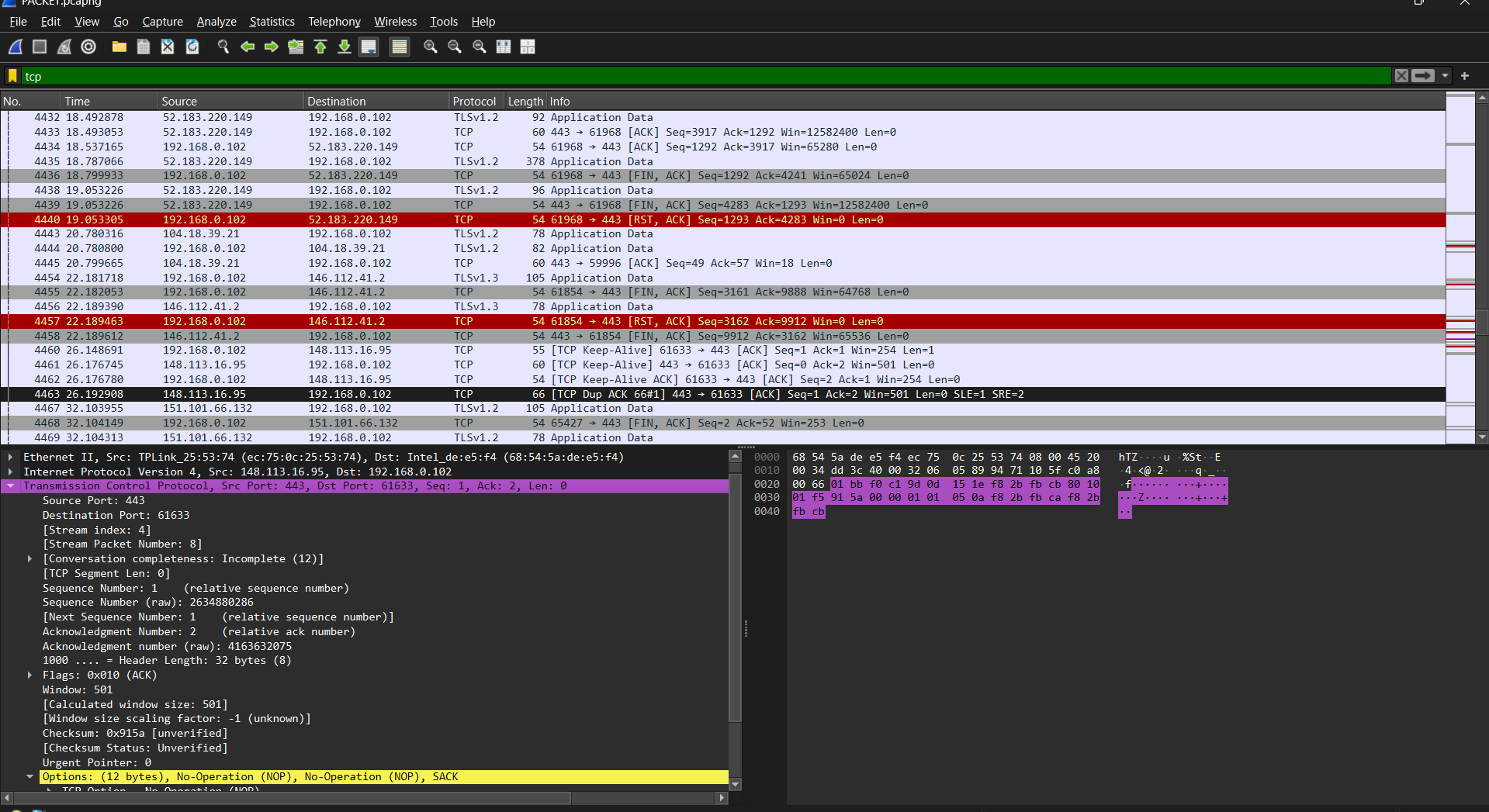
* **Analysis of Findings (Based on Your Screenshots):**
  + **Finding 1: Secure Connection Establishment (TCP & TLS Handshake)**
    - **How to Find:** Use the filter tcp.flags.syn == 1 && tcp.flags.ack == 0 to find the start of connections. Observe the packets that follow.
    - **Observation:** The capture shows a standard TCP 3-Way Handshake (SYN, SYN-ACK, ACK) followed by a TLSv1.3 handshake (Client Hello, Server Hello).
    - **Conclusion:** This is normal behavior for establishing a secure, encrypted connection. The use of TLS is a positive security control, but it prevents an analyst from inspecting the packet's payload for threats.

****

* + **Finding 2: Connection Maintenance (TCP Keep-Alive)**
    - **How to Find:** Use the display filter tcp.analysis.keep\_alive.
    - **Observation:** Packets labeled [TCP Keep-Alive] and [TCP Keep-Alive ACK] are visible. These are sent during periods of inactivity to keep a session open.
    - **Conclusion:** This is normal behavior for applications requiring a persistent connection and is not inherently suspicious.

****

* + **Finding 3: Anomaly - Abrupt Connection Termination (TCP Reset)**
    - **How to Find:** Use the display filter tcp.flags.reset == 1.
    - **Observation:** A packet with the [RST, ACK] flags was observed. A TCP Reset (RST) flag is used to abruptly and immediately terminate a connection.
    - **Conclusion:** This is an anomaly. While not always malicious, it indicates a non-standard event, such as an application crash or a firewall blocking the connection, and warrants further investigation

.****

**Combining Filters: You can combine filters using and or or .**

**CODE:** ip.addr == 192.168.0.102 and tcp.port == 443 and tls

**These filters are great for finding network problems or suspicious behavior.**

* **Filter for TCP Retransmissions:**

**CODE** : tcp.analysis.retransmission

**This shows packets that had to be sent more than once, which is a strong indicator of packet loss or network congestion.**

* **Filter for HTTP errors:**

**CODE :** http.response.code >= 400

**This instantly shows you all HTTP requests that resulted in an error, such as a 404 Not Found or a 403 Forbidden. It's excellent for debugging web application issues.**

* **Filter out traffic to or from an IP:**

**CODE :** !(ip.addr == 1.1.1.1)

**The ! operator means "not". This filter is perfect for removing noisy traffic from a known, trusted source (like a monitoring server) so you can focus on other activity.**

**Task 3: Reporting for Project 1**

* **Executive Summary:** "A network traffic analysis was performed using Wireshark. The analysis revealed examples of standard secure communication patterns, connection maintenance techniques, and a connection anomaly in the form of a TCP Reset."
* **Methodology:** "Live network traffic was captured and analyzed using the Wireshark graphical packet analyzer. Display filters were used to isolate and inspect specific communication patterns."
* **Recommendations:** For anomalies like a TCP Reset, recommend "further investigation of host-based logs to determine the cause of the abrupt termination."
* **Findings:**

- Normal Activity Observed:

- DNS requests to Google DNS (8.8.8.8).

- HTTPS traffic to common websites (Google, YouTube).

- Suspicious Activity (Example, if any):

- Multiple repeated TCP handshake failures (possible scanning attempt).

- Unknown IP communication outside common regions.

- Screenshots Attached:

5. Conclusion

The packet analysis tool was successfully implemented using Wireshark. The tool captured live network traffic, and analysis showed normal browsing traffic with no major threats detected during the test. This project demonstrates how packet analysis can identify suspicious activity and help in monitoring network security.

6. References

- Wireshark Documentation – https://www.wireshark.org/docs/

- Nmap Documentation – https://nmap.org/book/

- Scapy Documentation – https://scapy.readthedocs.io