Elevate Labs
Cybersecurity Internship Report

Task 5: Capture and Analyse Network Traffic using Wireshark

Submitted by: Namita Rana

Role: Cybersecurity Intern

Company: Elevate Labs

Date of Submission: 11 August 2025

Objective

The primary objective of this report is to conduct a comprehensive study of network communication by capturing and analysing live traffic using the Wireshark protocol analyser. This involves systematically recording network packets traversing a selected interface, applying protocol-specific filters, and examining the data at multiple layers of the OSI model.

Specific goals include:

1. **Capturing Live Network Traffic**
    - Utilizing Wireshark to intercept and record real-time network packets from a specific network interface (e.g., Wi-Fi or Ethernet).
2. **Identifying and Classifying Protocols**
    - Detecting commonly used protocols, including TCP, UDP, ICMP, DNS, and ARP, within the captured dataset and understanding the role and functionality of each.
3. **Analysing Packet Structures**
    - Examining protocol headers and payloads to understand packet composition.
    - Identifying key fields in Ethernet, IP, and transport layer headers, along with their significance in network operations.
4. **Understanding Real-World Network Behaviour**
    - Observing how multiple protocols operate simultaneously in a typical communication session.
    - Recognizing protocol dependencies, such as DNS queries over UDP and TCP-based application traffic.
5. **Developing Network Troubleshooting Skills**
    - Using Wireshark's filtering, sorting, and decoding features to diagnose potential communication issues.
    - Understanding how packet-level analysis can assist in detecting anomalies, performance bottlenecks, and security threats.

By achieving these objectives, the exercise enhances both theoretical understanding and hands-on proficiency in network protocol analysis, a fundamental skill for cybersecurity, network administration, and IT troubleshooting.

Acknowledgement

I would like to extend my heartfelt gratitude to **Elevate Labs**, whose guidance, encouragement, and valuable feedback played a crucial role in the successful completion of this report on **Network Traffic Capture and Analysis using Wireshark**.

They provided the infrastructure, technical resources, and learning environment necessary to carry out this practical task efficiently. Access to reliable internet connectivity, computing resources, and academic references proved essential in successfully performing live packet captures and subsequent analysis.

I am also grateful to the **Wireshark development team and the open-source community**, whose dedication and ongoing contributions have made this powerful protocol analyser freely available to students, researchers, and professionals around the world. Without their work, this project would not have been possible.

Lastly, I extend my appreciation to my **family** for their understanding, patience, and encouragement during the time spent on research, experimentation, and documentation of this project.

— Namita Rana

Privacy and Ethical Considerations

This network traffic capture and analysis exercise was carried out with strict adherence to privacy, ethical, and legal standards. Wireshark, while an invaluable tool for learning and troubleshooting, is capable of intercepting sensitive information such as login credentials, personal messages, and other confidential data if used on unsecured or unauthorized networks. Recognizing this, every precaution was taken to ensure that the tool was used responsibly and within approved boundaries.

All packet captures were conducted exclusively on a personal or institution-approved network, with explicit permission from the network owner. No attempts were made to monitor, intercept, or capture traffic from unauthorized networks or third-party systems. Where applicable, individuals connected to the network were informed about the capture activity to maintain transparency and trust.

Throughout the capture process, no personally identifiable information (PII) or sensitive credentials were intentionally collected, stored, or shared. Any packet data that could reveal private user activity was either anonymized or excluded entirely from this report. The sole focus of the analysis was to study protocol behaviour, traffic flow, and packet structure for educational purposes.

The exercise was conducted in compliance with relevant cybersecurity, network monitoring, and data protection laws. At no point were any activities undertaken that could be considered malicious, invasive, or in violation of legal standards. Furthermore, all captured data was stored securely during the analysis phase and will be deleted upon completion of this project to prevent unauthorized access or misuse.

By following these measures, this work maintained a balance between achieving its learning objectives and upholding the ethical responsibility to respect the privacy and confidentiality of network communications.

Table of Contents

Chapter 1 – Introduction to Wireshark

1.1 Overview of Wireshark

Wireshark is a powerful and widely adopted open-source network protocol analyser designed for capturing, analysing, and troubleshooting network traffic. It allows network professionals, cybersecurity analysts, and researchers to examine packets in detail, offering insight into how data travels across a network. By breaking down packets into protocol-specific components, Wireshark enables a deep understanding of communication processes, security vulnerabilities, and performance issues.

Initially developed under the name *Ethereal* in 1998 by Gerald Combs, Wireshark was rebranded in 2006 due to trademark issues. Since then, it has grown into an essential tool for both industry and academia, supported by a strong community of contributors and continuously updated to recognize and decode thousands of network protocols.

Wireshark supports live capture from various interfaces, including Ethernet, Wi-Fi, Bluetooth, and more. It also offers the ability to open and analyse previously recorded capture files, making it useful for retrospective investigations. This versatility has made Wireshark the standard tool for network troubleshooting, protocol development, and educational purposes in networking courses worldwide.

1.2 Features and Capabilities

Wireshark offers an extensive set of features that make it a preferred choice for packet analysis:

- **Live Packet Capture** – Captures network packets in real time from multiple interfaces and displays them in a structured, readable format.
- **Offline Analysis** – Opens saved capture files for in-depth review and historical analysis.
- **Protocol Decoding** – Supports over 1,000 protocols, decoding each packet to display headers, payloads, and control information.
- **Filtering and Search** – Offers powerful capture and display filters, enabling users to focus on specific traffic (e.g., only TCP packets or DNS queries).
- **Colour Coding** – Uses customizable colour rules to visually distinguish different types of traffic for faster identification.
- **Export Options** – Allows exporting data in multiple formats such as plain text, CSV, or XML for reporting or further analysis.
- **Cross-Platform Support** – Available for Windows, macOS, and Linux, ensuring accessibility for a wide user base.

1.3 Role in Network Analysis

Wireshark plays a crucial role in network analysis, providing visibility into the flow of data across a network and enabling both proactive and reactive monitoring. Common uses include:

- **Troubleshooting Network Issues** – Identifying packet loss, retransmissions, or delays to diagnose performance bottlenecks.
- **Security Analysis** – Detecting suspicious activities such as scanning attempts, unauthorized access, or malicious payloads.
- **Protocol Development and Testing** – Assisting developers in validating custom protocols or checking compliance with existing standards.
- **Educational Use** – Serving as a hands-on teaching tool to demonstrate network concepts, protocol operations, and communication flows.

In the context of this task, Wireshark was used to capture and analyse traffic for multiple protocols, specifically **TCP, UDP, ICMP, DNS, and ARP**. Each protocol's behaviour, packet structure, and communication patterns were studied in detail to understand their operational characteristics within the network. This foundational understanding will inform the subsequent chapters of the report.

Chapter 2 – Tools Used

This chapter details the software and hardware resources employed for capturing and analysing network traffic. All tools were chosen for their reliability, compatibility, and ability to provide accurate packet-level inspection without introducing unnecessary complexity into the testing environment.

2.1 Software Requirements

1. **Wireshark**
    - **Purpose:** Served as the primary network protocol analyzer for packet capture and detailed inspection.
    - **Functions Utilized:**
        - Real-time capture from the selected network interface.
        - Application of display filters to isolate TCP, UDP, ICMP, DNS, and ARP traffic.
        - Detailed breakdown of protocol headers and payloads.
        - Saving and exporting captured packets in .pcapng format for archival and external review.
2. **Operating System**
    - **Platform:** A stable desktop operating system capable of running Wireshark without compatibility issues.
    - **Role:** Provided the execution environment, maintained driver compatibility, and ensured smooth interaction between Wireshark and the network hardware.
3. **Supporting Utilities**
    - **Command-line tools:** Simple utilities (e.g., ping, nslookup, arp) to generate specific traffic for capture.
    - **Web browser:** Used to access web resources, triggering DNS lookups and HTTP/HTTPS connections for analysis.

2.2 Hardware Requirements

1. **Network Interface Card (NIC)**
    - **Type:** Either an Ethernet adapter or a wireless network adapter, depending on the network setup.
    - **Mode of Operation:** Configured to allow the capture of all packets visible to the interface, enabling comprehensive monitoring.

2. **Host System**
    - **Specifications:**
        - Moderate processing capability to handle live packet capture without performance degradation.
        - Adequate RAM (at least 4 GB recommended) for smooth operation of Wireshark.
        - Sufficient storage space to save large packet capture files for later review.
3. **Network Connection**
    - **Type:** Wired Ethernet or wireless LAN with active data exchange during the capture period.
    - **Requirement:** Stable and consistent connectivity to ensure continuous data capture.

Chapter 3 – Methodology

This chapter outlines the systematic approach adopted to perform the network traffic capture and analysis using Wireshark. The process was carefully designed to ensure accuracy, reproducibility, and compliance with privacy and ethical guidelines. Each stage of the workflow was documented to maintain traceability and transparency.

3.1 Installation of Wireshark

1. **Acquisition of Software**
    - The latest stable release of Wireshark was downloaded directly from its official website (*https://www.wireshark.org/*) to ensure software authenticity and avoid tampered or malicious versions.
2. **Installation Procedure**
    - The installer package was executed with administrative privileges to allow installation of both the Wireshark application and the required packet-capturing driver (**Npcap**).
    - The default installation settings were retained, ensuring full functionality for live traffic capture.
3. **Post-Installation Verification**
    - Upon completion, Wireshark was launched, and a verification check was performed to confirm:
        - Successful installation of Npcap.
        - Availability of all active network interfaces.
        - Correct version number matching the downloaded release.

3.2 Capturing Live Network Traffic

1. **Interface Selection**
    - The active **Wi-Fi interface** was selected from the list of available network adapters in Wireshark.
2. **Capture Configuration**
    - Promiscuous mode was enabled to capture all packets visible to the wireless adapter.
    - In environments that supported it, monitor mode was considered for capturing raw 802.11 frames for enhanced analysis.
3. **Execution of Capture Session**

- The capture session was started during normal wireless network activity.
- Data was collected only from the controlled network to ensure that only authorized traffic was analysed.

4. **Ethical Considerations**
   - Only a controlled and non-sensitive network was monitored.
   - Traffic from unknown external users or unrelated devices was excluded from analysis to maintain privacy compliance.

3.3 Generating Traffic for Analysis

1. **Command-Line Generated Traffic**
   - **ICMP:** ping command was used to send echo requests to known, reachable hosts.
   - **DNS:** nslookup was used to perform DNS queries and capture request-response exchanges.
   - **ARP:** The arp -a command was executed to reveal address resolution activity on the network.
2. **Browser-Based Traffic**
   - Multiple websites were accessed to generate HTTP/HTTPS (TCP-based) and potential UDP-based traffic.
   - This also triggered DNS lookups and occasional redirects, enriching the captured protocol variety.
3. **Timing and Repetition**
   - The above activities were repeated at intervals to simulate typical user behaviour and to capture packets under different network loads.

3.4 Filtering and Analysing Captured Packets

1. **Applying Display Filters**
   - Wireshark's display filters (tcp, udp, icmp, dns, arp) were applied one at a time to isolate each protocol.
2. **Packet-Level Inspection**
   - For each protocol, multiple packets were examined in detail to identify:
     - **Frame details** (frame number, time, size in bytes).
     - **Ethernet headers** (MAC addresses and type).

- **Network layer details** (source/destination IPs, protocol type).
- **Transport/application layer details** depending on the protocol.

3. **Correlation and Context Analysis**
    - Packets were reviewed in sequence to observe relationships between request and response patterns (e.g., TCP handshake, DNS query/response).
    - Notable anomalies, such as retransmissions or unusually large packets, were flagged for reporting in the *Observations and Findings* chapter.

3.5 Saving and Exporting Capture Files

1. **Saving Entire Captures**
    - Complete session data was saved in .pcapng format for archival and possible re-analysis.
2. **Exporting Filtered Data**
    - Protocol-specific filtered captures were exported separately for easier reference in the report.
3. **Data Handling**
    - All files were stored in a secure, access-controlled location.
    - Only necessary, anonymized packet data was included in the final documentation to protect sensitive information.

Chapter 4 – Protocols Identified

During the Wireshark capture session on the wireless network, multiple network protocols were detected and analysed. This chapter presents an overview of five key protocols observed in the captured data. Each protocol plays a distinct role in network communication, from data transport to address resolution. The details provided focus on technical functionality and general characteristics without exposing any private or sensitive information from the capture.

4.1 TCP (Transmission Control Protocol)

TCP is a **connection-oriented transport layer protocol** that ensures reliable data delivery between devices. In the Wi-Fi capture:

- TCP was primarily observed in **HTTP(S)** web requests and other application communications.
- Packets displayed the typical **three-way handshake** process for session establishment, followed by data transfer and termination sequences.
- Features such as sequencing, acknowledgments, and error checking were evident in the packet headers.
- Common ports identified in the traffic included **port 80 (HTTP)** and **port 443 (HTTPS)**, though no payload content was inspected to maintain privacy.

4.2 UDP (User Datagram Protocol)

UDP is a **connectionless transport layer protocol** designed for fast, low-overhead communication. In the Wi-Fi capture:

- UDP was observed in **DNS queries**, certain application communications, and media-related traffic.
- Unlike TCP, there was no handshake; packets were sent without prior connection setup.
- Protocol headers revealed source and destination ports, along with length and checksum values.
- UDP traffic was more lightweight compared to TCP, reflecting its use in speed-prioritized transmissions.

4.3 ICMP (Internet Control Message Protocol)

ICMP operates at the **network layer** and is primarily used for diagnostic and error-reporting purposes. In the wireless capture:

- Echo requests (ping) and replies were identified, confirming device reachability within the network.
- ICMP packets were small in size, containing type, code, and checksum fields.
- Their occurrence was intentional, as ICMP traffic was generated specifically for controlled testing purposes.

4.4 DNS (Domain Name System)

DNS is an **application layer protocol** responsible for translating human-readable domain names into IP addresses. In the Wi-Fi session:

- DNS queries and responses were visible as UDP packets, generally on **port 53**.
- Queries requested the resolution of domain names into IPv4 addresses (A records).
- Responses provided the mapped IP addresses, enabling subsequent TCP or UDP connections.
- The captured DNS traffic was generated by controlled browsing activity.

4.5 ARP (Address Resolution Protocol)

ARP operates at the **link layer** and is used to map IP addresses to physical MAC addresses within a local network. In the wireless capture:

- ARP requests were sent to determine the MAC address corresponding to a given IP address.
- ARP replies returned the resolved MAC address to the requesting device.
- This exchange allowed devices on the Wi-Fi network to locate each other for direct communication.

Chapter 5 – Packet Structure Analysis

Understanding the structure of network packets is essential for effective analysis in Wireshark. During the Wi-Fi packet capture session, several layers of encapsulation were observed. This chapter examines the structure of packets at the data link, network, and transport layers, using only generalized and non-sensitive details from the recorded data.

5.1 Ethernet Frame Structure (Wi-Fi Adaptation)

Although Wi-Fi (IEEE 802.11) operates differently from traditional Ethernet, Wireshark often presents wireless traffic in an **Ethernet-like format** for ease of interpretation when operating in capture modes without raw 802.11 frames.

The standard Ethernet frame structure includes:

- **Destination MAC Address** – The hardware address of the receiving device.
- **Source MAC Address** – The hardware address of the sending device.
- **Ether Type / Length** – Identifies the protocol encapsulated in the payload (e.g., IPv4, IPv6).
- **Payload** – The encapsulated data (IP packet or other protocol data).
- **Frame Check Sequence (FCS)** – Used for error detection.

In monitor mode (if enabled), raw **802.11 frame details** such as management, control, and data frames would be visible, along with signal strength information.

5.2 IPv4/IPv6 Packet Structure

The network layer packets observed in the Wi-Fi capture primarily used **IPv4**, though IPv6 capability was also present in the environment.

IPv4 Header Fields included:

- **Version** – Set to 4 for IPv4 packets.
- **Header Length** – Specifies the size of the IP header.
- **Source IP Address** – The sender's IP (anonymized in this report).
- **Destination IP Address** – The intended receiver's IP (anonymized).
- **Time to Live (TTL)** – Limits the packet's lifespan to prevent endless routing loops.
- **Protocol** – Indicates the transport protocol (e.g., TCP, UDP, ICMP).
- **Header Checksum** – Ensures header integrity.

IPv6 Header Fields (less common in the captured data) included:

- **Version** – Set to 6 for IPv6 packets.
- **Source & Destination IPv6 Addresses** – 128-bit addresses, anonymized in this report.
- **Next Header** – Specifies the type of the next encapsulated protocol.
- **Hop Limit** – Similar to TTL in IPv4.

5.3 Transport Layer Headers

Two major transport protocols were analysed from the Wi-Fi traffic: **TCP** and **UDP**.

TCP Header Fields observed included:

- **Source & Destination Ports** – Identifying application endpoints.
- **Sequence & Acknowledgment Numbers** – Supporting ordered, reliable delivery.
- **Flags** – Such as SYN, ACK, FIN, and RST to manage session states.
- **Window Size** – Flow control mechanism for data transmission.

UDP Header Fields included:

- **Source & Destination Ports** – Mapping to application services.
- **Length** – Size of the UDP packet including header and data.
- **Checksum** – For error detection in the packet.

This layered breakdown highlights the encapsulation process in network communication over Wi-Fi:

- Data originates at the **application layer** (e.g., web browsing, DNS queries).
- It is then packaged into **transport layer segments** (TCP/UDP).
- The transport segments are wrapped into **IP packets**.
- Finally, the IP packets are encapsulated into **Ethernet-like or 802.11 frames** for transmission over the wireless medium.

Chapter 6 – Observations and Findings

This chapter summarizes the notable patterns, behaviours, and anomalies observed during the analysis of Wi-Fi network traffic captured with Wireshark. All findings have been sanitized to avoid revealing any personally identifiable information or sensitive system details.

6.1 General Traffic Patterns Observed

- The network displayed a **steady flow of mixed traffic** generated by common applications such as web browsing, DNS lookups, and background system processes.
- **Bidirectional communication** was consistently observed, with both inbound and outbound packets present for TCP and UDP connections.
- Short bursts of ICMP packets were identified during testing when connectivity checks were performed.
- Most traffic originated from trusted local devices connected to the same wireless access point, consistent with a secure private network environment.

6.2 Protocol Usage Frequency

Based on filtered counts in Wireshark:

- **TCP** was the most frequently observed protocol, primarily supporting HTTP(S) traffic and other reliable data transfers.
- **UDP** traffic was present in moderate volumes, largely from DNS queries and responses.
- **ICMP** usage was minimal and primarily linked to intentional test pings.
- **DNS** requests occurred periodically as applications and browsers resolved domain names.
- **ARP** exchanges were observed whenever devices refreshed their local address mappings.

6.3 Notable Anomalies or Unusual Packets

- No evidence of malicious or suspicious packets was found within the analyzed capture.
- Occasional **retransmissions and duplicate TCP acknowledgments** were detected, likely due to transient network latency or packet loss.

- Some broadcast ARP requests did not receive immediate replies, which may indicate inactive or temporarily disconnected devices on the network.
- Slight variations in packet timing and size were consistent with normal wireless link variability.

6.4 Correlation Between Protocols and Network Activity

- **TCP and DNS** frequently appeared in sequence, with DNS resolving hostnames prior to establishing TCP connections.
- **ARP** activity tended to precede first-time communications between devices, highlighting its role in mapping IP addresses to MAC addresses.
- **ICMP** packets correlated directly with manual ping tests and did not appear during normal background activity.
- **UDP** traffic, while less frequent than TCP, was critical in supporting lightweight and time-sensitive tasks such as DNS resolution.

These observations reinforce the typical structure of day-to-day network communication on a Wi-Fi connection. The protocol mix, traffic frequency, and minor retransmission patterns are consistent with expected performance in a secure and stable wireless environment.

Chapter 7 – Conclusion

7.1 Summary of Key Insights

This report documented the process of capturing, analysing, and interpreting Wi-Fi network traffic using Wireshark. The analysis covered multiple layers of network communication, from Ethernet-like (or 802.11) frames through IP packets and transport layer segments.
Key takeaways include:

- The network primarily carried **TCP-based traffic**, supplemented by **UDP**, **DNS**, **ARP**, and minimal **ICMP** communication.
- Protocol usage patterns matched typical secure Wi-Fi network behaviour, with DNS lookups preceding TCP sessions and ARP handling address resolution before data exchange.
- The packet structure analysis confirmed the proper encapsulation process across OSI layers, demonstrating Wireshark's effectiveness as a network diagnostic and research tool.
- No malicious or unauthorized traffic was detected, indicating a stable and secure network environment during the observation period.

7.2 Limitations of the Analysis

While the analysis yielded meaningful insights, certain constraints should be acknowledged:

- **Capture Scope** – The dataset was limited to packets captured during a specific session, and therefore does not represent all possible network conditions over time.
- **Network Environment** – The capture was performed on a single Wi-Fi segment; other network segments or devices outside the capture range were not analysed.
- **Protocol Depth** – While key protocols were examined, the analysis did not include in-depth application layer inspection beyond general observations, in order to maintain privacy.
- **Tool Limitations** – Wireshark relies on the capturing device's network interface and driver capabilities; certain Wi-Fi management frames or encrypted payloads may not have been visible without additional configurations such as monitor mode or decryption keys.
