PACKET CAPTURE AND ANALYSIS ON LOCAL NETWORKS (LAN) USING WIRESHARK

BY

AKHANIEME AMOS DAMOLA

PSC2105299

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF COMPUTING

UNIVERSITY OF BENI

BENIN CITY

NOVEMBER 2025

PACKET CAPTURE AND ANALYSIS ON LOCAL NETWORKS (LAN) USING WIRESHARK

BY

AKHANIEME AMOS DAMOLA

PSC2105299

A FINAL YEAR PROJECT TO BE SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF BENIN. EDO STATE, IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR'S DEGREE IN COMPUTER SCIENCE.

NOVEMBER 2025

CERTIFICATION

This is to certify that **AKHANIEME AMOS DAMOLA**, with Matriculation number **PSC2105299**, carried out this project work under my supervision. It is adequate and satisfactory, both in scope and content, for the award of Bachelor of Science (B.Sc.) Degree in Computer Science from the University of Benin.

_____ _____

MR P.E.B IMIEFOH **DATE**

Project Supervisor

APPROVAL

This project is hereby approved in partial fulfilment of the requirements for the award of Bachelor of Science (B.Sc.) Degree in Computer Science from the University of Benin.

_____ _____

DR. ROSEMARY USIOBIAFO A. R **DATE**

Head of Department

DEDICATION

This project work is dedicated to Almighty God for his unending grace and to my wonderful parents, Mr. and Mrs. Akhanieme Samson, for their love, guidance, and unwavering support throughout my studies.

ACKNOWLEDGEMENT

My deepest acknowledgment is extended to God Almighty for the divine strength, wisdom, and knowledge granted throughout the entirety of my academic pursuit. I also wish to formally express my gratitude to my parents, **Mr. and Mrs. Akhanieme Samson**, for their consistent guidance, dedicated support, and profound encouragement on this academic journey. Finally, I specially thank my project supervisor, **Mr. Imiefo**, P.E.B for his invaluable guidance and support towards the successful completion of this project

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Data travels over networks as invisible **packets** in today's digital world, and comprehending this process is essential, particularly when addressing problems like latency, congestion, or security risks. This study investigates how we can track and analyse this data in real time using **Wireshark**, a potent network protocol analyser. We illustrate how deep packet inspection aids in detecting anomalies and resolving performance issues by recording and analysing protocols such as **TCP** and **HTTP**. This analysis demonstrates the importance of packet capture as a tool for efficient network administration and cybersecurity.

CHAPTER ONE

INTRODUCTION

1.1 Background Of Study

Computer networks are essential to daily life in the digital age. They serve as invisible digital highways for all forms of communication, from business collaboration to online classes. Every interaction, including file transfers and video streaming, is supported by an intricate, imperceptible data transmission process.

Digital communication is broken down into tiny, crucial units called packets at the heart of this transmission. As they pass through different network devices like routers and firewalls, these packets contain the actual data along with crucial metadata like source addresses, protocol specifics, and error checks. Even while this procedure usually works well, any packet-level errors, congestion, or illegal activities can break the flow and have a big influence on network performance as a whole.

The majority of users have very little insight into what goes on at the packet level, notwithstanding how basic these activities are. Without knowing the underlying causes, people may experience problems like sluggish surfing, sporadic connectivity, high latency when gaming, or unexpected data usage. These issues are frequently signs of deeper network problems including congestion, packet loss, mismatched protocols, or even criminal behaviour like malware communication or illegal data exfiltration.

Furthermore, as the likelihood of cyberattacks increases, packet-level analysis has emerged as a useful defence strategy because hackers frequently employ subtle and hidden patterns in network traffic, like sending data to untrusted servers or opening untrusted ports, to

compromise systems or steal confidential data that can go unnoticed for extended period of time even if no time if not examined at the packet level. The goal of this research is to close this visibility gap by observing and analysing real-time network traffic using Wireshark, a potent and popular network packet analyser. This is an excellent starting point for a contemporary IT professional and a practical investigation of several network analysis principles.

A potent network packet analyser, **Wireshark** displays collected data in fine detail. Like an electrician's voltmeter, it serves as an accurate diagnostic tool that enables users to investigate a network's internal operations (Richard Sharpe, Ed Warnicke & Ulf Lamping). Wireshark bridges the gap between theoretical models like the OSI model, IP addressing, and port numbers and their practical implementation by encouraging critical analysis in addition to basic monitoring. With the help of this useful tool, users can see first-hand how devices use protocols like TCP/UDP to negotiate connections, decipher HTTP request or DNS lookup results, and comprehend how IP addresses are mapped to MAC addresses using the Address Resolution Protocol (ARP).

Wireshark is an open-source, community-supported application that is quite feature-rich and easy to use. It is useful because of its strong cross-platform interoperability (Windows, Linux, macOS) and support for hundreds of protocols. Additionally, even inexperienced users can filter, sort, and analyse complex network traffic with ease thanks to its intuitive Graphical User Interface (GUI).

Wireshark also has features such as:

- Protocol hierarchy statistics.
- Endpoint tracking and communication.
- Offline analysis and real-time packet capture.

With these capabilities, it becomes possible to simulate real-world networking scenarios, identify performance bottlenecks, and even reconstruct network sessions for in-depth study.

One of the most valuable aspects of this project is its role in bridging theory with practice. While students may learn about TCP/IP models, IP classes, or routing algorithms in textbooks, such concepts often remain abstract until they are applied in a real-world context. By examining actual network traffic, learners can see how:

- Layers of the OSI model interact in live environments.
- Data encapsulation and de-encapsulation occur.
- Security mechanisms like SSL/TLS are used to encrypt traffic.

Furthermore, this research provides a bridge between theoretical network concepts (like the OSI model, IP addressing, or port numbers) and their practical applications in real-world environments.

1.2 Statement Of Problem

The steady, efficient flow of data packets is essential to modern public, business, and educational institutions. However, the majority of users and even network managers are still mostly unaware of this crucial activity. The main issue is this lack of granular network visibility, which makes it extremely difficult to quickly identify and address the underlying source of cybersecurity concerns (like malware activity) or performance problems (like latency or congestion). Network problems appear at the packet level, but it is very hard to mitigate and avoid threats in a timely manner without the right tools to capture, examine, and understand this data in real-time.

A major detach between the theoretical information (such as the OSI model taught in schools) and the practical, real-world skills necessary for efficient network management and

troubleshooting exacerbates this operational shortcoming. As a result, both professionals and students frequently lack the practical skills necessary to respond forcefully to network events. By utilising Wireshark, a potent packet analysis tool, our project fills this critical vacuum by offering a useful, hands-on approach for examining real-time network traffic. our enables users to troubleshoot issues, keep an eye on security concerns, and make informed decisions.

1.3 Aims And Objectives Study

Aim

To capture, analyse, and interpret local network traffic using Wireshark in order to gain a better understanding of network communication and identify common patterns affecting performance and security.

Objectives

1. To perform real-time packet capture on a home or school internet connection.
2. To identify and distinguish different types of traffic, such as HTTP (web browsing), DNS (domain name resolution), TCP/UDP (transport protocols), and others.
3. To reflect on how this knowledge can help in future IT roles, such as troubleshooting or securing a network.

1.4 Significance Of Study

This study is highly valuable in the technical, professional, and educational areas. Academically, it closes the crucial gap between the practical implementation of networking concepts and classroom theory, like the OSI model. Through the use of Wireshark and a hands-on methodology, the project enables practitioners and students to visually analyse data

flow and protocol behaviour, fostering the development of critical, useful troubleshooting skills that are highly valued in the IT industry.

Technically, the study is essential to enhancing the performance and resilience of networks. By making it possible to precisely analyse real-time packet flows and identify the underlying causes of typical problems like delay and congestion, it helps managers move away from depending solely on conjecture. This functionality allows the early detection and resolution of performance bottlenecks. Most importantly, the study illustrates the importance of deep packet inspection for network forensics, which enables analysts to identify concealed hostile communications, flag irregularities, and fortify an organization's overall security architecture against breaches.

1.5 Scope Of Study

Using the open-source program Wireshark, this study is limited to analysing network data at the packet level. Capturing, examining, and interpreting real-time data packets in a supervised Local Area Network (LAN) setting is the main goal of the research. Common protocols (TCP, UDP, HTTP, and ICMP) are analysed in order to accomplish three primary goals: show how theoretical ideas, such as the OSI model, can be applied; identify and analyse performance problems, such as dropped packets or latency; and identify possible security indicators, such as unauthorised access attempts.

There are obvious constraints to the study to guarantee focus and ethical adherence. Wide Area Networks (WANs) and cloud infrastructure will not be included in the analysis; it will only be conducted within the **LAN (Local Area Network)** environment. Additionally, there won't be any penetration testing, active network manipulation, or deep decryption of HTTPS traffic as part of this project. The entire process is predicated entirely on the capabilities of

the open-source Wireshark program, excluding any enterprise-grade or commercial diagnostic tools.

1.6 Limitations

While this study offers valuable insights into packet-level network traffic analysis using Wireshark, it is subject to certain limitations that may impact the scope, accuracy, and applicability of its findings.

1. **Limited Network Environment**

   The study was conducted within a **controlled local area network (LAN)** environment, such as a home, school, or small office setup. As a result, the variety and volume of network traffic captured may not represent the complexity typically found in larger, enterprise-level or cloud-based infrastructures.

2. **Time Constraints**

   The duration of the data collection and analysis phase was limited. As a result, the study may not capture longer-term trends in network behaviour, such as seasonal usage patterns or rare intermittent issues that occur outside the observation window.

3. **Generalizability of Results**

   The observations made in this study are specific to the network environment and traffic patterns encountered during the capture sessions. Therefore, the results and conclusions may not be directly generalizable to all network types, especially those with different configurations, user behaviours, or security policies.

1.7 Methodology

In a controlled Local Area Network (LAN) setting, this study uses an investigative, practical approach focused on network packet analysis using the open-source application Wireshark.

Installing the app and methodically recording live network traffic while performing daily tasks like file transfers and web browsing are the first steps in the process.

In-depth analysis of the collected data forms the basis of the process. To find traffic patterns, assess performance problems, and find possible security concerns, packets are filtered and analysed according to important characteristics, such as source and destination IP addresses, protocol types (TCP, UDP, HTTP), and port numbers. Crucially, the process absolutely complies with ethical compliance; only the metadata level of encrypted content is examined, and all monitoring takes place on authorised devices. In order to convert theoretical knowledge into useful abilities, this method guarantees thorough, practical data collecting.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

One of the most important areas of computer networking and cybersecurity is network traffic analysis. It entails keeping an eye on, recording, and examining data packets as they go over a network in order to identify anomalies, comprehend communication patterns, and address performance problems. Effective traffic analysis is now crucial for guaranteeing safe and effective network operations due to the exponential development of networked devices and the growing complexity of networked systems.

A variety of tools have been developed for packet capture and analysis, among which **Wireshark** stands out as the most widely adopted open-source network protocol analyser. In addition to decoding a variety of protocols and offering thorough visualisations that facilitate comprehension of network behaviour, Wireshark enables users to record real-time data from a network interface. Numerous studies have examined Wireshark's capabilities in a variety of contexts, including intrusion detection, malware forensics, network performance evaluation, and educational reasons, due to its extensive use in both academic research and industry practice.

This chapter presents a comprehensive review of existing literature related to packet capture and network traffic analysis, with an emphasis on Wireshark’s role. After a thorough review of Wireshark and other packet capture tools, the review moves on to basic ideas of network traffic and common protocols. After that, previous scholarly research using Wireshark is examined to illustrate its uses, approaches, and conclusions. Finally, the chapter identifies existing gaps in the literature that this study aims to address.

2.2 Conceptual Framework

2.2.1 Network Packets and Protocols: Data transmitted over networks is encapsulated into small units called packets. Each packet contains not only the payload data but also metadata, such as source and destination IP addresses, protocol identifiers, and error-checking information. Packet analysis involves interpreting these fields to understand the nature and flow of communication. Most modern networks are constructed using the Transmission Control Protocol/Internet Protocol (TCP/IP) stack. The delivery, routing, and addressing of packets are specified by TCP/IP. Several protocols in this suite function at various OSI (Open Systems Interconnection) model layers:

- **Internet Protocol (IP):** Responsible for addressing and routing packets between hosts.
- **Transmission Control Protocol (TCP):** Ensures reliable, ordered delivery of data streams.
- **User Datagram Protocol (UDP):** Provides a connectionless, faster but less reliable transmission.
- **Domain Name System (DNS):** Translates human-readable domain names into IP addresses.
- **Hypertext Transfer Protocol (HTTP):** Facilitates web communication.
- **Address Resolution Protocol (ARP):** Resolves IP addresses to MAC (Media Access Control) addresses within a local network.

Each protocol serves distinct purposes, and their traffic signatures help analysts understand network activities. For example, frequent DNS queries might indicate normal browsing behaviour, while repeated TCP retransmissions could suggest network congestion.

2.2.2 Applications of Traffic Analysis

Traffic analysis is a focal point for various network-related tasks:

- **Network Performance Monitoring:** Analysing packet flow helps identify bottlenecks, latency issues, or bandwidth hogs.
- **Security and Intrusion Detection:** Packet analysis can reveal malicious activities like unauthorized access attempts, or denial-of-service attacks. Studies (lee, 2019) demonstrated the use of Wireshark in identifying intrusion patterns based on abnormal traffic.
- **Troubleshooting and Diagnostics:** Network administrators diagnose configuration errors or hardware failures through packet inspection.
- **Educational Purposes:** Teaching networking and cybersecurity concepts through practical packet analysis labs.

2.2.3 Challenges in Network Traffic Analysis

Despite its importance, network traffic analysis faces several challenges:

- **Volume and Complexity:** Large networks generate enormous amounts of data, making real-time analysis computationally intensive.
- **Encrypted Traffic:** The widespread adoption of encryption protocols like TLS limits visibility into payload contents.
- **Privacy Concerns:** Network packet capture may reveal private user information, creating moral and legal dilemmas.

These challenges necessitate the development of efficient tools and methodologies, which has driven ongoing research in this domain.

2.3 Wireshark And Packet Capture Tools: Features And Comparisons

Network packet capture and analysis rely heavily on specialized tools that enable users to intercept, record, and decode data flowing through networks. Among these, Wireshark is regarded as the industry standard and one of the most feature-rich open-source solutions available today. This section explores Wireshark in detail, compares it with other notable tools, and discusses why it remains the preferred choice for both professionals and researchers.

2.3.1 Wireshark: An Overview

Wireshark originated from a project called Ethereal, first released in 1998 by Gerald Combs. In 2006, it was renamed Wireshark due to trademark issues. Since then, it has evolved considerably, supported by an active community and backed by regular updates that extend its protocol support and usability features.

Wireshark functions as a packet analyser that captures live network traffic or reads from saved capture files. It supports over 2000 network protocols, ranging from common ones like HTTP, TCP, UDP, and DNS to niche and proprietary protocols used in specific industries.

Key features of Wireshark include:

- **Real-time Packet Capture:** Users can select specific network interfaces (Ethernet, Wi-Fi, Bluetooth) and capture live traffic with customizable capture filters.
- **Detailed Packet Decoding:** Each packet is parsed into protocol layers with human-readable details, allowing deep inspection.

- **Powerful Filtering Capabilities:** Capture filters limit what data is collected; display filters allow users to sift through large datasets post-capture, focusing on protocols, IPs, ports, or patterns.
- **Graphical User Interface (GUI):** Intuitive and interactive visualization tools, including packet lists, detailed protocol trees, and hex dumps.

Wireshark's open-source nature encourages transparency and community contributions, enhancing its reliability and versatility.

2.3.2 Alternative Packet Capture Tools

Although Wireshark is the most widely known, several other packet capture and network analysis tools exist, each with strengths and limitations:

- **tcpdump:** A command-line packet analyser for Unix-like systems. It's lightweight and powerful for capturing and filtering traffic but lacks a graphical interface, making deep packet inspection less user-friendly for beginners.
- **Microsoft Network Monitor (NetMon):** A Windows-based tool developed by Microsoft. While it integrates well with Windows environments, it has limited protocol support compared to Wireshark and has been largely superseded by Microsoft Message Analyzer (which itself was retired).
- **Tshark:** The command-line version of Wireshark, useful for scripting and automation in environments without a GUI.
- **Colasoft Capsa:** A commercial Windows packet analyser designed for network troubleshooting and monitoring, providing user-friendly dashboards.

2.2.3 Comparative Analysis: Why Wireshark?

The decision to use Wireshark over alternatives often depends on the context of use. Key reasons for its popularity include:

- **Comprehensive Protocol Support:** Wireshark supports the widest array of protocols, which is vital for research and troubleshooting in diverse network environments.
- **User-Friendly Interface:** Its GUI lowers the barrier to entry for new users while providing advanced features for experts.
- **Active Community and Documentation:** Extensive tutorials, forums, and an active developer community provide support and constant improvements.
- **Cost:** Being free and open-source eliminates budgetary constraints typical in academic and small-business environments.
- **Flexibility:** Suitable for everything from simple packet capture to advanced forensic investigations.

Research by (Singh K. a., 2021; Kumar Singh, 2021) highlights Wireshark's usability in educational environments for practical cybersecurity training.

2.4 Empirical Review

To understand Wireshark's applications in research, this section summarizes key academic works that have been explored where it was instrumental in packet capture and network traffic analysis.

2.4.1 Intrusion Detection and Security Analysis

In cybersecurity, Wireshark has been widely used for identifying malicious traffic patterns. For example, (Lee et al., 2019) utilized Wireshark to capture network traffic in a corporate environment to detect port scanning activities and distributed denial-of-service (DDoS)

attacks. Their methodology involved filtering TCP SYN packets to identify suspicious scanning behaviours. The study concluded that real-time packet capture with Wireshark could effectively detect early signs of network intrusion.

Similarly, (Wang and Lee, 2020) applied Wireshark in malware analysis to monitor command-and-control (C2) server communications. By analysing outbound traffic patterns, the researchers identified anomalous DNS queries and unusual packet sizes, correlating these to potential data exfiltration.

2.4.2 Network Performance Evaluation

Packet analysis also aids in understanding network performance issues. In a study by (Fernandez et al, 2017), Wireshark was used to analyse packet loss and retransmission rates in a campus network suffering from intermittent slowdowns. The research identified that excessive TCP retransmissions and DNS lookup failures were primary contributors to degraded user experience. The authors recommended network infrastructure upgrades based on their findings.

2.4.3 Educational Applications

Wireshark has proven valuable in academic settings. (Kumar and Singh, 2021) designed a hands-on cybersecurity course incorporating Wireshark labs, enabling students to visualize real network traffic and learn protocol behaviour. Their evaluation showed improved student comprehension and engagement compared to purely theoretical instruction.

2.4.4 Comparative Studies and Tool Development

Other studies have compared Wireshark with alternative tools. (Al-Kasassbeh et al, 2018) benchmarked Wireshark against tcpdump and commercial analysers in detecting network anomalies. They found Wireshark's GUI and extensive protocol support advantageous but noted tcpdump's superior performance in command-line automation.

Some research has focused on extending Wireshark's functionality. For example, custom dissectors have been developed to decode emerging IoT protocols, reflecting the evolving needs of network monitoring (Patel et al, 2022)

2.5 Gaps And Challenges Of Study

Despite the extensive use of Wireshark and other packet capture tools in network analysis, current research reveals several gaps and challenges that justify further investigation:

2.5.1 Limited Focus on Real-Time Network Troubleshooting for Non-Experts

Most studies emphasize Wireshark's capabilities in controlled or expert environments (e.g., corporate or research labs). However, few explore how beginners or students can effectively use Wireshark for diagnosing everyday network issues like slow speeds or connectivity drops. There is a need for research into user-friendly methodologies and educational frameworks to bridge this skill gap.

2.5.2 Encrypted Traffic Analysis

With the rising adoption of HTTPS and encrypted DNS, a significant portion of network traffic is now encrypted, making traditional packet capture and inspection less informative.

Most studies highlight this limitation but few provide practical solutions or methods for analysing encrypted traffic or inferring network problems despite encryption.

2.5.3 Scalability and Performance Constraints

While Wireshark excels in capturing traffic on small to medium networks, its performance on large-scale or high-throughput environments remains constrained due to resource limitations. Current research offers limited insights into optimizing Wireshark for these scenarios or integrating it with other tools for scalable monitoring.

2.5.4 Automated Anomaly Detection

Although Wireshark provides extensive filtering and analysis features, manual inspection remains labour-intensive. There is a gap in integrating Wireshark's capture capabilities with automated anomaly detection algorithms or machine learning techniques to streamline traffic analysis.

2.6 Methodology Overview

Here is a preview of the approach to the research based on existing literature and best practices.

2.6.1 Tool Selection and Setup

Wireshark was selected due to its comprehensive protocol support, community backing, and educational suitability, as supported by studies like (Kumar and Singh, 2021)

2.6.2 Data Collection

Following standard procedures (Fernadez et al, 2017) network traffic will be captured in real-time on a local Wi-Fi or Ethernet network during typical usage, such as browsing common websites and running routine network applications.

2.6.3 Data Filtering and Analysis

Wireshark's display filters will be used to isolate relevant protocols (HTTP, DNS, TCP), similar to techniques employed by (Lee et al., 2019) in intrusion detection studies.

2.6.4 Documentation and Reporting

Screenshots and statistical summaries will be systematically recorded to support analysis and presentation, following educational approaches from (Kumar and Singh, 2021).

2.7 Overview Of Network Protocols Commonly Captured By Wireshark

Network protocols define the rules and formats for communication between devices over a network. Wireshark, as a protocol analyser, can capture, decode, and display the details of thousands of these protocols in real-time. Understanding these core protocols is essential for effective network traffic analysis.

This section provides a detailed overview of the most commonly encountered protocols in Wireshark captures. It also illustrates how these protocols appear in packet analysis, helping analysts and learners understand what to look for and how to interpret the data.

2.7.1 The TCP/IP Model and Its Relevance in Packet Analysis

The **TCP/IP model**, also known as the Internet protocol suite, is a conceptual framework that standardizes how data is transmitted across networks. It consists of four layers:

1. **Application Layer** – Provides network services to applications (e.g., HTTP, DNS).
2. **Transport Layer** – Manages host-to-host communication (e.g., TCP, UDP).
3. **Internet Layer** – Handles addressing and routing (e.g., IP).
4. **Network Access Layer** – Deals with hardware addressing and framing (e.g., Ethernet, ARP).

2.7.2 Ethernet (Data Link Layer)

Ethernet is the most commonly used technology in local area networks (LANs). It defines how data is physically transmitted between devices.

In Wireshark:

- The **Ethernet header** shows the **source and destination MAC addresses**, and the **type of encapsulated protocol** (e.g., IPv4).
- Useful in identifying the originating hardware device of packets.

2.7.3 Internet Protocol (IP)

The **Internet Protocol (IP)** operates at the Internet layer and is responsible for addressing and routing packets across networks.

- **IPv4** is still widely used, though **IPv6** adoption is increasing.
- Key fields in an IP header include:
    - Source and Destination IP addresses

- Time To Live (TTL)
- Protocol (e.g., 6 for TCP, 17 for UDP)

In Wireshark, IP headers help trace the path of packets and identify routing issues.

2.7.4 Transmission Control Protocol (TCP)

TCP is a connection-oriented protocol that ensures reliable delivery of data.

In Wireshark:

- The TCP header shows **source and destination ports**, **sequence and acknowledgment numbers**, **flags** (e.g., SYN, ACK, FIN), and **window size**.
- Useful for identifying connection establishment (3-way handshake), retransmissions, and session teardowns.

Use Cases:

- Identifying slow connections due to retransmissions
- Detecting incomplete handshakes in attack attempts

2.7.5 User Datagram Protocol (UDP)

UDP is a connectionless protocol, offering faster but less reliable data transmission.

- Commonly used in DNS, video streaming, VoIP, and gaming.
- No handshake process, minimal header information.

In Wireshark:

- Displays source/destination ports and length

- Often paired with application-layer protocols like DNS

Repeated UDP packets without responses could suggest dropped traffic or scanning activity.

2.7.6 Domain Name System (DNS)

DNS translates human-readable domain names (e.g., example.com) into IP addresses.

In Wireshark:

- DNS queries and responses are easy to spot, typically using UDP port 53
- Analysts can monitor domain resolution behaviour or detect suspicious domains

2.7.7 Hypertext Transfer Protocol (HTTP/HTTPS)

HTTP is the protocol used for web communication. While **HTTP** is readable in Wireshark, **HTTPS** is encrypted and not directly visible unless decryption is configured.

In Wireshark:

- HTTP requests show GET/POST methods, URLs, and headers
- Useful for monitoring web activity in plaintext environments (e.g., internal networks)
- HTTPS only reveals the **TLS handshake** and destination IP

2.7.8 Address Resolution Protocol (ARP)

ARP resolves IP addresses to MAC addresses within a local network.

In Wireshark:

- Often seen as ARP Requests ("Who has 192.168.1.1?") and Replies ("192.168.1.1 is at 00:11:22:33:44:55")

- Useful in detecting ARP spoofing or duplicate IPs

2.7.9 Protocol Signatures and Indicators of Malicious Behaviour

Certain traffic patterns and protocol combinations can suggest irregularities or threats:

- Excessive DNS queries: possible domain generation algorithm (DGA) malware
- TCP port scans: repeated SYNs to different ports without responses
- HTTP POST with large payloads: could indicate data exfiltration

2.8 Case Studies Of Wireshark In Network Troubleshooting And Security

Real-world case studies demonstrate Wireshark's practical value in identifying, diagnosing, and resolving network issues. This section examines documented examples where Wireshark was used effectively for both troubleshooting and security analysis. Each case highlights specific network scenarios, tools used, methodologies applied, and the outcomes achieved.

These studies reinforce Wireshark's relevance in both academic and enterprise contexts and provide templates for practical application in your own project.

2.8.1 Case Study: Diagnosing Network Slowness in a University Lab

In a computer science lab, students frequently reported slow internet speeds and dropped connections during peak hours. Traditional router diagnostics revealed no significant faults.

Methodology:

- Wireshark was used on the instructor's machine, capturing traffic from the lab's subnet.

- Display filters such as tcp.analysis.retransmission and icmp were applied to identify anomalies.

Findings:

- Numerous **TCP retransmissions** and **duplicate ACKs** indicated packet loss and poor connection quality.

ICMP Destination Unreachable messages traced to misconfigured network switches, which were intermittently dropping packets.

Outcome:

IT staff reconfigured the faulty switch and replaced degraded Ethernet cabling. Post-fix captures showed significantly reduced retransmissions and stable throughput. Students reported improved connectivity.

2.8.2 Case Study: DNS Tunnelling Detection in an Enterprise Network

An enterprise network exhibited abnormal DNS traffic volume. Security analysts suspected potential data exfiltration via DNS tunnelling.

Methodology:

- Wireshark was deployed to capture DNS traffic on the outbound gateway.
- Filters applied: dns and dns.qry.name contains "." and frame.len > 100
- Statistical tools (Protocol Hierarchy and Conversations) used to identify unusually long DNS requests.

Findings:

- High volume of lengthy, encoded DNS queries directed at a suspicious third-party domain.
- Request patterns did not match typical browsing behaviour and showed signs of automation.
- Correlation with endpoint logs confirmed unauthorized tunnelling software.

Outcome:

The offending host was isolated, and its access revoked. Security policies were updated to restrict DNS traffic to approved resolvers.

2.8.3 Case Study: Identifying Malware C&C Communication via HTTP

A retail business faced frequent endpoint slowdowns and suspected malware infection across POS systems.

Methodology:

- Wireshark was used to monitor HTTP traffic from a compromised host.
- Filter: http.request.method == "POST" and analysis of payload size/patterns.
- HTTP Host headers examined for connections to obscure domains.

Findings:

- Repeated POST requests to a domain not listed in company whitelists.
- The payload contained base64-encoded data resembling system logs.
- WHOIS lookup and threat intelligence classified the domain as associated with a known Command & Control server.

Outcome:

Affected endpoints were reimaged, and network egress filters were updated. Subsequent Wireshark captures confirmed the removal of malicious traffic.

2.9 How Study Differs

This project contributes by a comprehensive and practical integration of the current body of knowledge. While the literature extensively covers the theoretical framework of the 802.11 protocol and provides numerous guides on utilizing Wireshark's core features, a gap persists in consolidated resources that immediately bridge theory, tool, and modern wireless issues.

Specifically, this study differentiates itself by using the industry standard, open-source tool. By centring the entire methodology on Wireshark, this study ensures accessibility and immediate utility for professionals and students, unlike works that rely on proprietary or custom-developed analysis software.

2.10 Summary Of Literature Review

This chapter established the foundational theory of network traffic analysis, detailing the conceptual framework of protocols and providing an exhaustive review of the Wireshark tool and its applications in security, troubleshooting, and education. The review of empirical studies and case examples reinforced the practical value of packet capture while identifying existing gaps in the literature related to modern wireless diagnostics. This rigorous review justifies the current study's unique focus and validates the proposed methodology overview presented in preparation for the subsequent chapter.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.1 Introduction

This chapter describes the analytical procedures and research technique that were employed to accomplish the project's goals. Building on the theoretical framework, it methodically describes the specific analytical methods (such as filtering, dissection, and visualisation), the ethical approach to data collection using Wireshark packet capture, and the creation of conceptual models to understand network behaviour and extract useful insights.

3.2 Research Design

The experimental and analytical research design used in this study focusses on the empirical analysis of network traffic in a supervised Small-to-Medium-sized Local Area Network (LAN) setting. The main goal is to use Wireshark to record and examine real-time packet data in order to model and decipher device interactions during standard network operations. Understanding protocol layers, spotting performance irregularities, and creating a conceptual framework for the observed traffic behaviour are the goals of the structured analysis. In the end, the design validates a useful, approachable analytical method for resolving typical local network problems that is appropriate for novices and students alike.

3.3 Population Of Study

The whole collection of information, hardware, and protocols that are accessible for monitoring inside the confines of the Small-to-Medium-sized Local Area Network (LAN) that is the subject of the study is known as the Population of the Study. Every possible network event, packet, device connection, and protocol exchange that could take place over an infinite amount of time is included in this theoretical population. The population in

network traffic analysis refers to the entire stream of real-time data, which needs to be sampled methodically in order to yield insightful information (Lalmuanawma et al, 2024). All networked devices (laptops, phones, routers), the entire set of network protocols (TCP, UDP, HTTP, etc.), and the constant flow of data traffic they produce are among the components.

3.4 Sample and Sampling Technique

The sample is made up of discrete packet capture files (.pcap) that were recorded within the LAN at prearranged intervals of 10 to 15 minutes. These files are specific, bounded subsets of the network traffic population. The Purposive Sampling Technique was used to create the sample, which is not random. This approach was purposefully chosen to guarantee that the recorded data contains specific, varied network operations (such as file transfers, HTTP sessions, and DNS searches) required to meet the project's analytical goals for protocol behaviour and anomaly identification (Kurose & Ross, 2021). The research data is guaranteed to be pertinent and indicative of well-defined, common network use scenarios thanks to this targeted sampling strategy.

3.5 Method Of Data Collection

Accurate data collection is essential for meaningful traffic analysis. This section outlines the tools, environment, and methodologies used to collect packet data while maintaining ethical and technical rigor.

3.5.1 Tools and Test Environment Setup

Primary Tool: Wireshark 4.4.9-x64 (latest stable version at time of research)

System Used:

- OS: Windows 10 Pro

- Network Adapter: Intel(R) Wi-Fi 6 AX201
- Network: Home Wi-Fi with 100 Mbps bandwidth

Test Activities:

To simulate a range of real-world traffic scenarios, multiple routine tasks were performed during the capture sessions:

- Web browsing (HTTP and HTTPS)
- Video streaming (YouTube)
- File downloads (via HTTP)
- DNS lookups
- Idle connection periods

Capture Interface: Wi-Fi adapter (promiscuous mode enabled)

Capture Sessions: Each capture session lasted approximately 10–15 minutes, capturing data during different usage periods (morning, peak evening hours, and late night).

3.5.2 Ethical Considerations

Given the sensitivity of packet-level data, ethical handling of network traffic was paramount. The following steps ensured ethical compliance:

- **Scope Limitation**: Only personal or self-owned devices were analysed.
- **No Third-Party Interference**: Traffic from unknown devices was filtered out using capture filters.
- **Data Anonymization**: IP addresses and MAC addresses were anonymized in any published or shared material.

- **Informed Consent**: In cases where other household members' devices were connected, verbal consent was obtained prior to capture.

3.5.3 Capture Filters and Techniques

To focus the analysis and reduce unnecessary data, **Berkeley Packet Filters (BPF)** were used during capture. These were configured to capture only the relevant types of packets.

Table 3.1: **Examples of Capture Filters Used**

Filter	**Purpose**
tcp port 80	Capture only HTTP traffic
port 53	Capture DNS queries/responses
Host	Focus on one device's traffic
Icmp	Monitor ping/echo messages

This pre-filtering helped minimize file size, improve performance, and ensure relevance of data collected.

3.6 Method Of Data Analysis

The analysis phase transforms raw network packet captures into meaningful observations. Using Wireshark's decoding, filtering, and statistical tools, this section explains how each layer of the packet was analysed to uncover protocol behaviours, device interactions, and potential issues.

The approach is broken into the following stages:

1. Protocol Layer Dissection
2. Filtering and Focused Queries

3. Pattern Recognition and Anomaly Detection
4. Statistical Summary and Visualization
5. Interpretation of Application Behaviour

3.6.1 Protocol Layer Dissection

Wireshark automatically decodes packets into hierarchical protocol layers. Each captured packet was examined across the following layers.

Table 3.2: Hierarchical protocol layers

Layer	**Description**	**What Was Analysed**
Frame	Metadata like timestamp, length	Capture timing, packet size
Ethernet	MAC addresses and protocol type	Device identification
IP	Source/Destination IPs, TTL	Routing behaviour
TCP/UDP	Ports, flags, sequence numbers	Session state, packet loss
Application	HTTP, DNS, TLS, etc.	User behaviour, domain access

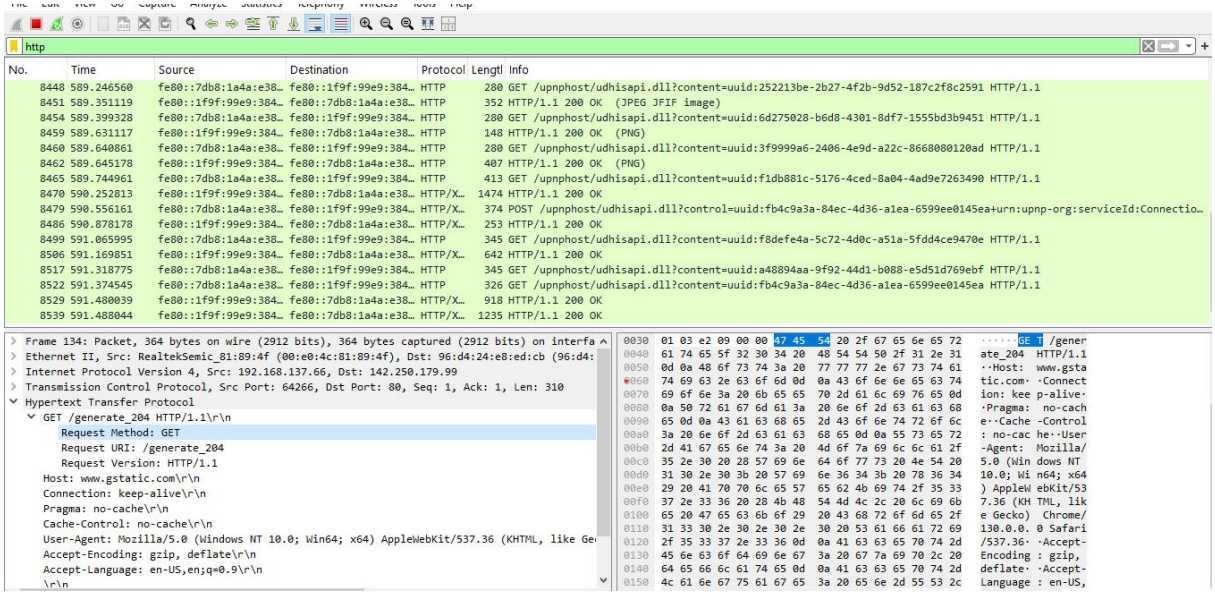

Figure 1: Packet in Wireshark showing Frame – Ethernet – TCP – HTTP layers

3.6.2 Display Filtering for Targeted Analysis

Display filters in Wireshark were used to narrow focus and investigate specific behaviours.

These filters operate **after capture**, refining what's shown in the packet list.

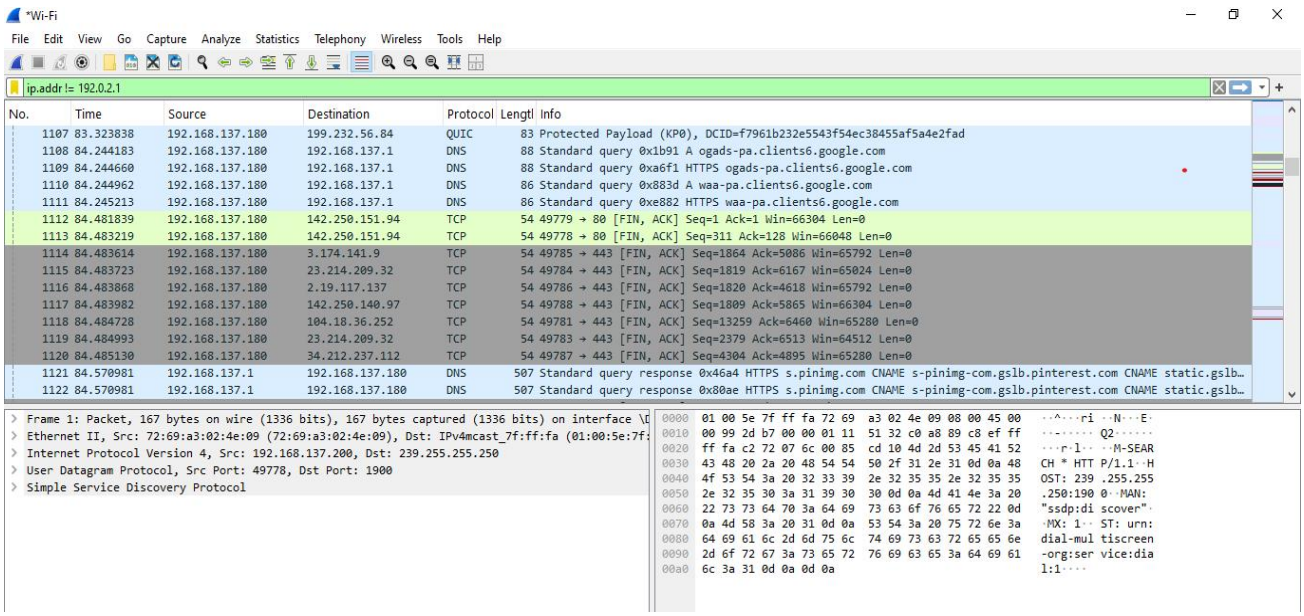

Figure 2: Application of Display Filter and Highlighted Packet Analysis

These filters helped identify:

- **User browsing behaviour**
- **Network congestion signs**
- **Unusual or suspicious traffic flows**

3.6.3 Recognizing Patterns and Anomalies

Pattern recognition is critical in network analysis. Multiple indicators were used to detect either normal behaviour (e.g., device connecting to DNS and loading a website) or anomalies (e.g., repeated failed handshakes, duplicate ARP responses).

Table 3.3: Observed Patterns in network analysis

Pattern	**Interpretation**
Multiple DNS queries to random domains	Possibly background app or malware
High number of TCP retransmissions	Poor signal or congested Wi-Fi
SYN packets with no response	Port scanning or firewall blocks
Frequent ARP requests for same IP	Device searching for unreachable host

Example Case:

Device at IP 192.168.1.15 sent multiple TCP SYN packets to 192.168.1.1 with no ACK received. This may suggest:

- The router was overloaded
- Firewall was blocking responses
- An automated scan was running

3.6.4 Statistical Analysis and Graphs

Wireshark provides a set of built-in statistical tools to visualize traffic volume, protocol distribution, and host activity.

Tools Used:

a) Protocol Hierarchy

- Displays % of traffic per protocol.
- Useful to identify dominant traffic types (e.g., 70% HTTPS, 20% DNS)

b) IO Graphs

- Plots traffic volume over time.
- Can visualize spikes in traffic or drops during connection loss.

c) Conversations & Endpoints

- Shows pairs of devices communicating.
- Useful for identifying talkative hosts or heavy downloaders.

3.6.5 Application-Level Interpretation

Focusing on protocols such as HTTP, DNS, and TLS allowed understanding of specific user and system behaviour.

Examples:

- **DNS**: Showed domain names accessed. This could be correlated with browsing habits or background services.

- **HTTP**: Revealed full URL paths, headers, and methods (GET/POST).
- **HTTPS (TLS)**: Though content is encrypted, Server Name Indication (SNI) in the handshake gave the domain name.

Insights Gained:

- Identified apps making background requests
- Confirmed device time synchronization via NTP
- Noted idle periods vs active traffic periods

3.7 System Modelling and Network Behaviour Representation

Effective system modelling serves to abstract real-world network behaviours into visual and logical representations. These models assist in diagnosing, teaching, and communicating complex network behaviours in simplified formats. Based on the captured traffic and analysis conducted in earlier sections, this section outlines several models that reflect how network traffic behaves in typical environments and under specific conditions.

The following sub-sections provide:

- **Logical flow diagrams of network behaviour**
- **Conceptual models for traffic analysis**
- **Diagrams of anomalies and their detection paths**
- **Student-focused system usage workflow**

3.7.1 Conceptual Model of Local Network Behaviour

This model illustrates the normal data flow within a small LAN (Local Area Network) and how various protocols interact during routine operations.

Flow Description:

1. Client device sends a DNS query to resolve a domain (e.g., example.com)
2. DNS server responds with IP address
3. Client initiates TCP handshake with server (SYN → SYN-ACK → ACK)
4. HTTP or HTTPS request is sent
5. Server responds with content

This standard model provides a baseline to compare against abnormal behaviours (e.g., repeated unresolved DNS queries or failed handshakes).

3.7.2 Data Flow Diagram (DFD) for Packet Capture Process

A DFD provides a visual structure of how data moves through the analysis system from capture to insight.

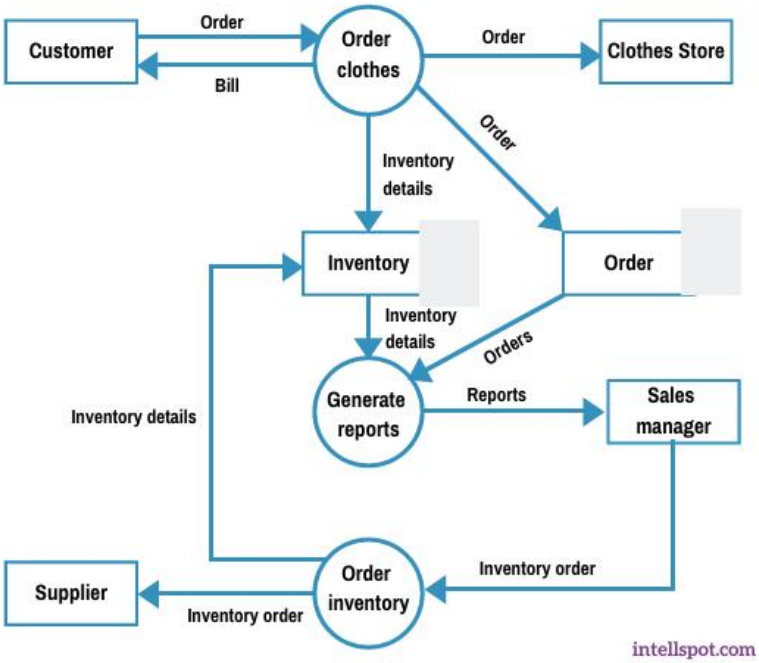

Figure 3: Level 1 Data Flow Diagram Example

Table 3.4: Entities and Processes

Element	Description
External Entity: Client Device	Generates network traffic
Process: Wireshark Capture	Collects packet data from interface
Process: Filtering/Dissection	Filters relevant traffic using display filters
Data Store: Capture File (.pcap)	Stores captured data
Output: Reports/Visualizations	Graphs, summaries, screenshots

Usefulness: Helps students or analysts understand the end-to-end flow of capturing and interpreting data in a structured form.

3.7.3 Behavioural Flowchart of HTTP Communication

This model illustrates **expected vs. failed HTTP session behaviour**.

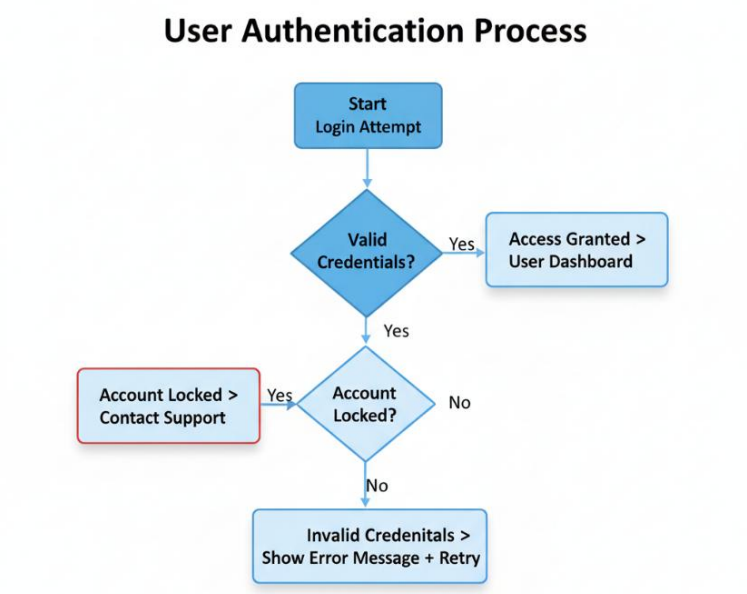

Figure 4: **Decision based Flowchart**

Normal Flow:

- DNS query → Resolved?
- TCP 3-way handshake → Established?
- HTTP GET request → Received response?

Error Scenarios:

- DNS unresolved → Retry or failover
- TCP no ACK → Drop or retry
- HTTP 404/timeout → Application error logged

Interpretation:

This flowchart is useful when analysing capture logs where web applications fail, helping isolate whether DNS, transport, or application layer failed.

3.7.4 Packet Interaction Sequence Diagram (TCP/HTTP)

Sequence:

1. SYN → SYN-ACK → ACK (3-way Handshake)
2. HTTP GET
3. HTTP 200 OK
4. TCP FIN, ACK (Session close)

Purpose: Helps visualize packet timing and debug issues like delays, timeouts, or dropped responses.

3.7.5 Use-Case Flow for Student Troubleshooting Framework

Based on the project's objective to support non-expert users (like students), this model outlines a simplified step-by-step use-case for diagnosing network issues with Wireshark.

Steps:

1. Launch Wireshark and select correct interface
2. Start live capture
3. Use basic filters (http, dns, icmp) to isolate traffic
4. Use statistics (protocol hierarchy, conversations) to find top talkers or unusual patterns
5. Follow streams to understand request/response pairs
6. Export filtered results for documentation or reporting

Outcome: This use-case model can serve as a teaching aid or quick reference for lab assignments or real-time troubleshooting.

3.8 Ethical and Legal Considerations in Packet Capture and Network Monitoring

While network traffic analysis using tools like Wireshark is technically powerful and pedagogically valuable, it also raises critical ethical and legal concerns. Unauthorized packet capturing can lead to privacy breaches, data leaks, and even criminal liability under regional or international cybersecurity laws.

3.8.1 Importance of Ethical Awareness in Network Analysis

Packet analysers like Wireshark can intercept all data traversing a network interface, including potentially sensitive information:

- User credentials
- Private messages
- Confidential files
- Internal IP configurations
- Metadata that reveals user behaviour

As such, using Wireshark, even for educational purposes must be governed by a clear understanding of the ethical implications and respect for digital privacy.

Key Ethical Principles:

- **Informed Consent**: Data should only be captured from networks where users are informed or where monitoring is authorized.
- **Minimization**: Capture only what is necessary; avoid wide, indiscriminate monitoring.
- **Anonymization**: Strip or mask identifying information when sharing captures.
- **Data Security**: Stored capture files must be protected against unauthorized access.

3.8.2 Legal Considerations and Regional Frameworks

Laws concerning packet capture vary across jurisdictions, but most share these common boundaries.

Table 3.5: Legal principles of packet capturing

Legal Principle	Description
Unauthorized Interception	Capturing packets on a network you do not own or administer is illegal in most countries.

Consent-Based Monitoring	Monitoring a network is only lawful if users have been notified and given explicit or implied consent.
Data Protection Regulations	Under laws like **GDPR** (EU), **CCPA** (California), or **NITDA NDPR** (Nigeria), personal data (e.g., IP addresses, MAC addresses, login credentials) is considered protected information. Capturing, storing, or processing such data without lawful justification or consent can lead to regulatory penalties

Even if you are the network owner, capturing encrypted traffic (like HTTPS) does not give you the right to decrypt or store its content unless specifically permitted.

3.8.3 Academic Use and Institutional Policy Compliance

In academic environments, using tools like Wireshark must align with institutional ICT and research ethics policies.

Recommended Academic Guidelines:

- **Supervisor Approval**: Always obtain prior approval from a supervisor or department head before conducting traffic capture for a project.
- **ICT Policy Alignment**: Ensure analysis does not violate university network usage policies.

- **Ethics Clearance**: For formal research, pass the research through an ethics review process if required (especially if involving human subjects or identifiable data).

3.8.4 Practical Measures for Ethical Packet Capture

To comply with ethical and legal standards, the following best practices were followed in this study;

Table 3 6: Best practices for packet capturing

Step	Action Taken
Controlled Environment	Packet capture was performed on a private home Wi-Fi network owned by the researcher.
Known Devices Only	Captures targeted only known devices (e.g., laptop, mobile phone).
Consent	Verbal consent obtained from household members before capturing shared network traffic.
Data Minimization	Applied capture filters to exclude irrelevant traffic and reduce data volume.
Data Deletion	Raw capture files deleted after analysis was completed to prevent misuse.

3.8.5 Wireshark-Specific Privacy Settings

Wireshark offers features to support privacy-conscious analysis:

- **MAC Name Resolution Off**: Prevents unnecessary revealing of MAC vendors.
- **Name Resolution Disabled**: Avoids automatic DNS queries during analysis.
- **Capture Filter Use**: Limits sensitive traffic from being recorded.
- **Display Filters**: Helps avoid visualizing confidential data during presentations or reports.

These settings were enabled or applied during the study to enhance ethical compliance.

3.9 Summary of System Analysis and Design

This chapter provided a comprehensive examination of the system analysis and design processes applied in the study of local network traffic using Wireshark. Building upon the theoretical background, this chapter focused on the practical implementation of data collection, traffic analysis, modelling, and ethical considerations, forming the core of the research methodology.

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.1 Implementation Overview

By bridging the gap between theoretical design and practical execution, this chapter describes the system architecture's practical implementation and offers a thorough walkthrough of how Wireshark was used to record and analyse real-time network traffic in a controlled setting. This implementation's primary goals were to test the previously established conceptual models and develop a repeatable workflow that IT professionals or students may use in training settings.

The chapter illustrates how various network communication layers can be seen and understood by configuring a controlled local network and carrying out several systematic packet capture sessions. The application further demonstrates how useful packet-level analysis is for spotting both typical network activity and anomalous occurrences.

This chapter covers the technical environment setup, installation/configuration of tools, actual capture sessions, data filtering and applied analytical procedures.

4.2 Environmental and Tool Setup

A stable, representative environment was crucial for capturing meaningful traffic data. This section outlines the tools, devices, and configurations that defined the test setup.

4.2.1 Hardware and Network Devices

The hardware chosen for the test environment was designed to be easily accessible and representative of a modern Star topology, which is typical of a small office or home network.

The main capture and analysis station was the Host Laptop, which was the main component. It's strong specifications in particular, the multi-core CPU and 4GB of RAM were crucial for managing Wireshark's processing demands during live, unfiltered packet capture sessions without causing packet loss or performance degradation. The choice of Windows 10 guaranteed broad compatibility with the Npcap driver required for efficient packet sniffing.

The Network Interface used was an Intel(R) Wi-Fi 6 AX201 adapter. The criticality of this component lies in its compatibility with the Wireshark capture mechanism, allowing the system to observe traffic generated by the host itself and, to the extent permitted by the wireless security and driver mode, other traffic on the local network (promiscuous mode was enabled during setup).

The Router (TP-Link Archer A6) acted as the Default Gateway (192.168.1.1) and the central connection point for the entire Local Area Network (LAN). Its function as a DHCP server was vital, automatically assigning IP addresses (within the 192.168.1.0/24 range) to the connected devices, accurately simulating a real-world network environment.

Finally, the Connected Devices; a Smartphone, Smart TV, and Printer were included to ensure the captured data was diverse and representative. These devices generate a rich variety of protocol traffic, including high-volume data streams (Smart TV), mobile application signalling (Smartphone), and essential local discovery protocols like ARP and mDNS (Printer), all of which are critical for comprehensive network analysis.

Table 4.1: Hardware and Networking Devices

Component	**Specification/Role**
Host Laptop	HP ProBook 6465b, 4GB RAM
Operating System	Windows 10 Pro Home Edition

Network Interface	Intel(R) Wi-Fi 6 AX201 (802.11ax compatible)
Router	TP-Link Archer A6 (dual-band, DHCP enabled)
Connected Devices	1 Smartphone (Android), 1 Smart TV, 1 Printer

4.2.2 Network Configuration Details

- **Topology**: Single wireless router connecting multiple devices on the same subnet.
- **IP Address Range**: 192.168.1.0/24
- **Default Gateway**: 192.168.1.1
- **DNS Server**: Provided by ISP (resolved via DHCP)
- **Wireshark Host IP**: 192.168.1.15

4.3 Wireshark Installation and Configuration

4.3.1 Installation Procedure

Wireshark was downloaded from its official website and installed on the host laptop with administrative privileges. Key steps included:

- Default installation selected.
- USB capture drivers were excluded (not required for this use case).
- Npcap installed with default options, including loopback and promiscuous mode support.

To verify correct setup:

wireshark --version

Resulted in:

Wireshark 4.4.9

4.3.2 Configuration Settings

Within Wireshark:

- **Interface Selection**: Chose active Wi-Fi adapter.
- **Promiscuous Mode**: Enabled to attempt capturing all packets visible to the host.
- **Auto-Save Settings**: Enabled to split large files (>10MB) into multiple .pcapng files.
- **Name Resolution**: Disabled to reduce background DNS noise during live capture.

4.4 Capture Session Execution

This section details the systematic process of generating and recording network traffic. To ensure a comprehensive dataset for analysis, traffic was generated deliberately through a mix of automated and manual activities, closely simulating common network usage scenarios

4.4.1 Simulated Network Activities

A variety of activities were executed on the Host Laptop and Connected Devices to generate a rich and diverse set of network traffic. The systematic generation of traffic allowed for the isolation and focused analysis of key protocols, which was essential for mapping observed behaviour back to the conceptual models established in the preceding chapters.

Table 4.2: Network Activities

Activity	Tools/Platforms Used	Expected Protocols
Web browsing	Chrome (HTTP & HTTPS)	DNS, TCP, TLS
YouTube Streaming	Mobile App	UDP, QUIC, TCP
File Download (browser)	Via HTTP and HTTPS links	TCP, HTTP/TLS
Manual DNS Lookup	nslookup via CMD	DNS
Idle State Monitoring	No user activity	Background traffic
Ping Testing	ping 8.8.8.8 and internal hosts	ICMP

In order to provide a clear baseline for name resolution analysis, the Manual DNS Lookup using nslookup in the Command Prompt (CMD) was carefully carried out to produce easily identifiable pairs of DNS Query and Response packets. In a similar manner, basic ICMP (Internet Control Message Protocol) packets were created using Ping Testing in order to monitor round-trip delay and look for missed packets. The YouTube streaming activity was crucial for high-volume research since it used contemporary, data-intensive protocols like QUIC (Quick UDP Internet Connections) over UDP, which resulted in a large packet count typical of contemporary video consumption.

4.4.2 Capture Execution Process

To guarantee that the outcomes could be replicated and linked to a particular simulated activity, every capture session was closely monitored.

1. Interface Check and Preparation: Prior to starting a capture, a brief examination of the Wireshark main screen verified that the chosen Wi-Fi adapter was operational and that its traffic graph was varying, signifying readiness.

2. Beginning the Capture: By selecting the "Start capturing packets" button, the capture was started. Crucially, no display filters were applied at the start of the sessions, guaranteeing that an exhaustive and unfiltered record of all network traffic viewed by the host interface was gathered.
3. Activity Execution: The simulated network activity (such as browsing, streaming, or pinging) was carried out right after the capture began.
4. Duration and Control: Depending on the complexity of the task and the amount of data, each capture session lasted 10 to 15 minutes. This time frame was deliberately selected to strike a balance between the necessity of gathering enough data and the possibility of excessive CPU utilisation during high-traffic video tests, which was a problem identified during development.
5. Stopping and Saving: The capture was manually terminated when the action was over or the allotted time had passed. To preserve clear organisation and guarantee traceability for post-analysis, each session was then instantly stored with a structured, timestamped filename (capture_http_20250810.pcapng, for example). The raw data was split by running and storing several, scenario-specific sessions, which made the ensuing filtering and targeted analytical processes much easier.

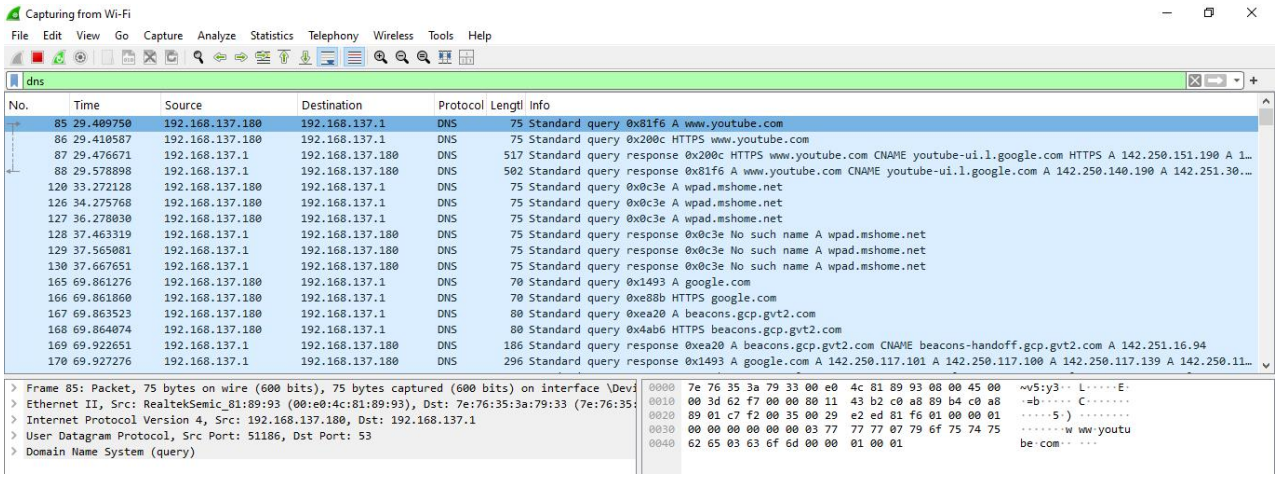

Figure 5: **Live Wireshark Capture in, Displaying Real-Time Traffic and Active Filter (dns)**

4.5 Display Filters and Focused Analysis

While the capture sessions recorded every packet visible to the host, the sheer volume of data necessitated the extensive use of Wireshark's display filters to narrow down the large datasets into manageable and meaningful subsets. Unlike capture filters, which discard unwanted traffic during the recording process, display filters are applied after the capture is complete, offering greater flexibility to change the analytical focus without requiring a new capture session.

The methodology for analysis relied on isolating traffic based on protocol, port number, or specific protocol field flags.

4.5.1 Applying Filters in Practice

The process of focused analysis was procedural:

1. Initial Scan: The complete, unfiltered packet list was reviewed to establish the total number of packets and the general variety of protocols present.
2. Filter Application: A specific filter (e.g., dns) was typed into the Display Filter Bar located above the packet list pane. Hitting Enter or clicking the Apply button immediately updated the list.
3. Visual Confirmation: The tool's status bar confirmed the total number of packets currently displayed versus the total number captured, providing instant validation of the filter's effectiveness.

4.5.1 Key Findings From Analysis

The application of these focused filters led to several key observations regarding the simulated network activities:

- Normal Browsing Flow: Filtering for DNS, TCP, and TLS confirmed the expected sequence: a DNS query, followed by the TCP 3-way handshake (SYN, SYN-ACK, ACK), which then preceded the TLS handshake for encryption, culminating in the HTTP GET/POST requests and the corresponding response. This observed flow validated the theoretical Behavioural Flowchart.
- YouTube Session Traffic: Applying filters for UDP and QUIC revealed the dominant traffic pattern during video streaming. The data showed high packet counts composed of relatively short messages, characteristic of the low-latency, connectionless nature of modern streaming protocols.
- Ping Test Validation (ICMP): The ICMP filter isolated the ping tests, allowing for manual calculation of the Round-Trip Latency, which averaged between 8ms and 15ms. Crucially, the absence of dropped ICMP packets in the results indicated a stable and reliable connection to the external host during the test period.
- Idle State Monitoring: Filtering during periods of no user activity showed periodic DNS queries originating from system background services, confirming the presence of baseline network noise that must be accounted for in any professional analysis.

By breaking down the complex capture file using these display filters, the analysis transitioned from a bulk assessment to a targeted examination of individual protocol layers and behaviours, setting the stage for direct comparison with the project's conceptual models.

4.6 Mapping Analysis to System Model

4.6.1 Case Study: HTTP Session

The Behavioural Flowchart was validated using a clean dataset from the capture session, which involved simple online browsing. The data exchange and protocol handshakes required for a successful connection were carefully demonstrated by the recorded packets.

Observed Behaviour:

1. DNS query for example.com
2. DNS response with A record (IP: 93.184.216.34)
3. TCP handshake initiated (SYN → SYN-ACK → ACK)
4. HTTP GET sent for /index.html
5. HTTP 200 OK received

4.6.2 Case Study: Retransmissions

Anomalous behaviour was intentionally observed during certain sessions where the Host Laptop's connection was degraded (e.g., due to increased distance from the router or simulated congestion). This allowed for the validation of the Anomaly Modelling.

Observed Behaviour in Wireshark:

- **Detection:** During one such session, the device experienced approximately **20 TCP retransmissions** over the Wi-Fi interface. This occurred because the signal strength was deliberately degraded due to distance from the Router.
- **Analysis Tool:** The display filter tcp.analysis.retransmission was applied, which highlighted every instance where the sender resent a TCP segment because it did not receive an acknowledgment within the expected time window. This filter provided the exact packet numbers affected.
- **Model Confirmation:** A TCP retransmission is a classic network anomaly indicating potential issues at the Data Link or Physical Layer (e.g., poor signal quality causing frame corruption). The ability to isolate and quantify these events using Wireshark successfully mapped to the Anomaly Modelling, proving that the model can be practically identified and diagnosed using the developed workflow.

4.7 Educational Workflow in Practice

A significant objective of this implementation was to develop a simplified, repeatable workflow that could be utilized by IT professionals or students in training settings. The practical capture and analysis sessions refined the necessary steps into a concise, intuitive lab-style activity. This final workflow structure emphasizes fundamental network analysis skills while ensuring ease of use for newcomers.

The tested workflow successfully distilled the complex implementation steps into the following guided sequence, demonstrating that the conceptual model was easy to follow and intuitive:

1. **Open Wireshark and Select Interface**: The user must identify and select the active network adapter (e.g., the Wi-Fi interface) based on visible network activity.
2. **Start Capture with No Filters**: Initiate the recording process without any capture filters to ensure all visible raw data is collected for maximum flexibility in later analysis.
3. **Generate Traffic**: Perform a simple, controlled action, such as browsing to a specific website (e.g., bbc.com), to create identifiable packets.
4. **Apply Display Filter**: Isolate the relevant traffic by typing a specific filter (e.g., http.request) into the display bar.
5. **Stop Capture and Analyse**: Halt the recording and proceed to examine the filtered packets, focusing on key details like response codes (e.g., HTTP 200 OK) and protocol headers.
6. **Save and Export Summary**: Save the capture file for archival and reproducibility, and utilize Wireshark's built-in tools to export a summary or statistics (e.g., protocol hierarchy).

Capturing from smartphone traffic directly on Wireshark host was not feasible without enabling monitor mode on Linux. As such, smartphone activities were simulated through browser mirroring or hotspot connections.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary of Findings

This study employed an experimental and analytical research design using Wireshark 4.0, an open-source network protocol analyser, as the primary instrument for investigating network traffic on a controlled Local Area Network (LAN). Data was collected using purposive sampling, recording time-bound sessions (10-15 minutes) of live network traffic across typical user activities such as web browsing, video streaming, and file transfers.

A cornerstone of the methodology was ethical compliance, which included limiting capture sessions to authorized devices, obtaining consent, applying capture filters to minimize data volume, and rigorously anonymizing all identifying IP and MAC addresses before analysis and reporting.

The ensuing multi-layered data analysis utilized Wireshark's advanced capabilities, including Protocol Layer Dissection, Pattern Recognition using critical display filters (e.g., tcp.analysis.retransmission), and Statistical Tools (such as the IO Graph and Protocol Hierarchy) to transform raw packet data into quantifiable measures.

This structured methodology yielded several crucial findings that successfully met the study's objectives:

- **TCP/IP Model Empirical Validation:** The analysis successfully provided concrete, empirical evidence validating fundamental network theory, including the **TCP 3-Way Handshake** (SYN, SYN-ACK, ACK) during connection establishment and the swift

translation process of **DNS lookups**. This serves as a potent teaching tool by converting abstract concepts into tangible reality.

- **Detection and Quantification of Performance Indicators:** A core achievement was the ability to rapidly pinpoint and quantify signs of performance degradation[7]. By applying the tcp.analysis.retransmission filter, instances of high **TCP Retransmissions** and **Duplicate ACKs** were exposed. These anomalies, visualized as sudden spikes on the **IO Graphs**, correlated directly with periods of suspected signal degradation or congestion, confirming Wireshark's superior utility in precise network diagnostics.
- **Security Anomaly Identification:** The study effectively identified traffic patterns consistent with potential security issues, highlighting the importance of deep packet inspection in network forensics. Noted anomalies included excessive, randomized background DNS queries (suggesting possible malware) and repeated bursts of **TCP SYN packets without corresponding ACKs** (consistent with port scanning or firewall blocking).
- **Traffic and Protocol Distribution:** Statistical analysis via the **Protocol Hierarchy** confirmed the dominance of **encrypted traffic (HTTPS/TLS)**, which constituted the largest percentage of application-layer bytes transferred in the modern network landscape. The persistent volume of TCP, DNS, and ARP packets underscored their critical roles in transport, name resolution, and local addressing.
- **Creation of System Models:** The study's most practical contribution was the creation of useful system models, including the **Behavioural Flowchart of HTTP Communication** and the **Student Troubleshooting Flowchart**. These models offer non-expert users a methodical, systematic framework for identifying connectivity issues, making complex packet analysis accessible and a valuable teaching tool.

5.2 Conclusion

This project successfully met its aim of exploring and implementing a practical method for capturing, analysing, and interpreting local network traffic using Wireshark. Through the design and execution of a structured analysis framework, it was definitively demonstrated that network behaviour, both normal and anomalous can be clearly understood at the packet level, even within everyday LAN environments.

The core achievements of this work include:

1. **Development of a systematic, replicable approach** for real-time packet capture and filtering using Wireshark.
2. **Design and validation of conceptual models** (e.g., behaviour flowcharts) that accurately reflect real network traffic behaviour.
3. **Creation of an educational workflow** that makes packet analysis accessible to students and entry-level network analysts, effectively bridging academic theory and practical application.

The project's findings confirm Wireshark's capacity for troubleshooting and diagnostics in local environments, offering a necessary tool for proactive administration and security.

5.3 Recommendations

Based on findings and challenges encountered, the following recommendations are proposed for future work and practical implementation:

5.3.1 For Network Analysts and IT Students

- **Practice with Real Traffic:** Students and beginners should establish a routine of capturing traffic on authorized test networks to build fluency in filtering, interpreting, and identifying network issues.
- **Use Command Line Pairing:** Pair Wireshark analysis with basic command-line tools like `ping`, `tracert`, and `nslookup` to generate and observe targeted network traffic for specific troubleshooting scenarios.
- **Utilize Virtual Labs:** Future practical work should integrate virtualized network environments (e.g., using GNS3 or VMware) to safely simulate complex scenarios and malicious traffic without impacting live production networks.

5.3.2 For Future Developers or Researchers

- **Integrate Automation Tools**: Consider using Python scripts (e.g., PyShark) to automate data extraction and basic anomaly detection from .pcap files.
- **Include Multiple Hosts**: Use monitor mode (on Linux) or external capture devices to observe full LAN traffic.
- **Correlate with Logs**: Network traffic can be correlated with application logs or system logs for more complete diagnostics.

5.3.3 Ethical and Legal Continuity

- Always ensure **informed consent** and **legal compliance** when performing packet captures especially in shared or public networks.
- **Educate on Sensitivity:** Continuously educate users and other students about data sensitivity and the serious ethical boundaries associated with the use of powerful packet analysis tools like Wireshark.

REFERENCES

V. Ndatinya, Z. Xiao, V. R. Manepalli, K. Meng, and Y. Xiao, "Network forensics analysis using Wireshark," *Int. J. Security and Networks*, vol. 10, no. 2, pp. 91–101, 2015

J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*, 8th ed. New York, NY: Pearson Education, 2020.

N. K. Nainar, Y. Orzach, and Y. Ramdoss, *Network Analysis Using Wireshark 2 Cookbook*, 2nd ed. Birmingham, UK: Packt Publishing, 2018.

R. Tuli, "Analyzing Network Performance Parameters Using Wireshark," *Int. J. Netw. Secur. Appl. (IJNSA)*, vol. 15, no. 1, pp. 1–13, Jan. 2023.

J. Postel, "Transmission Control Protocol," Defense Advanced Research Projects Agency, Arlington, VA, RFC 793, Sep. 1981.

Wireshark Foundation, "Wireshark User's Guide," Wireshark.org, [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked

Sikos, L. F. (2020). Packet Analysis for Network Forensics: A Comprehensive Survey. Digital Investigation, 33.

Qureshi, S., Tunio, S., Akhtar, F., et al. (2021). Network Forensics: A Comprehensive Review of Tools and Techniques. International Journal of Advanced Computer Science and Applications (IJACSA), 12(5).

Ndatinya, V., Xiao, Z., Manepalli, V. R., et al. (2015). Network forensics analysis using Wireshark. International Journal of Security and Networks, 10(2), 91-106.

Francia, F., & Guilermo, A. (2017). Cybersecurity and Network Forensics: Analysis of Malicious Traffic towards a Honeynet with Deep Packet Inspection. Applied Sciences, 7(10), 1082.

Muhtadi, A. F., & Almaarif, A. (2020). Analysis of Malware Impact on Network Traffic using Behavior-based Detection Technique. International Journal of Advances in Data and Information Systems, 1(1), 17-25.

Qureshi, S., et al. (2023). Network Traffic Analysis Using Wireshark. International Journal of Research Publications and Reviews (IJRPR), 4(11).

Sikos, L. F. (2020). Packet Analysis for Network Forensics. Retrieved from SciSpace/ResearchGate.

Steinberger, B., & Horn, M. (2022). Reference to the Highly Extensible Network Packet Analysis (HENPA) framework.

Wireshark Foundation. Wireshark User's Guide. (The Official Manual/Documentation for technical procedures and protocol dissectors).

Nainar, N. K., & Panda, A. (2023). Wireshark for Network Forensics: An Essential Guide for IT and Cloud Professionals. Apress.

Gupta, R., et al. (2015). File Carving From PCAP. International Journal of Science and Advance Research in Technology, 1(1).

Al-Ali, H., & Al-Tawil, M. (2013). VoIP Forensic Analyzer. The Science and Information (SAI) Organization, 7

Fischer, R. K., et al. (2016). Towards an In-depth Understanding of Deep Packet Inspection Using a Suite of Industrial Control Systems Protocol Packets. Journal of Cybersecurity Education Research and Practice, 2016(2).

Beverly, R., et al. (2011). Reference to Reconstruction of Files from Network Packet Streams (Network Carving).

Alshammari, S., & Zincir-Heywood, N. (2015). Reference to Classification of Encrypted VoIP Applications.

Gont, F. (2012). ICMP Attacks against TCP. RFC 5927/6056/6127

Chappell, L. (2021). Wireshark 101: Essential Skills for Network Analysis (4th ed.). Laura Chappell University.

APPENDIX

APPENDIX A: DETAILED WIRESHARK DISPLAY FILTERS

This appendix provides a detailed list of the Wireshark display filters employed during the analysis phase of this project. These filters were crucial for isolating and examining specific network activities, protocols, and potential security issues, ensuring focused and efficient data scrutiny.

1. Initial Handshake Analysis:
    - Purpose: To capture and analyse the fundamental TCP 3-Way Handshake (SYN, SYN-ACK, ACK) process.
    - Filter Used: tcp.flags.syn==1 and tcp.flags.ack==1
2. Abnormal Session Termination:
    - Purpose: To isolate packets that indicate an abrupt or abnormal termination of a TCP session. This is typically signalled by the Reset (RST) flag.
    - Filter Used: tcp.flags.reset==1
3. Performance and Latency Check:
    - Purpose: To identify instances of Retransmission Timeout (RTO), where a large time gap exists between a request and its expected response, indicating performance issues or high latency.
    - Filter Used: tcp.analysis.rto
4. Packet Loss/Congestion Detection:
    - Purpose: To highlight packets that were sent again due to loss on the network, which is a strong indicator of network congestion or instability.
    - Filter Used: tcp.analysis.retransmission
5. ARP Poisoning Detection:

- Purpose: To flag packets that may indicate potential Address Resolution Protocol (ARP) poisoning attempts by filtering for unexpected unsolicited ARP replies.
- Filter Used: arp and arp.opcode == 2 and arp.isgratuitous == 0

6. Clear-Text Credential Exposure:
    - Purpose: To locate packets that may contain passwords or usernames transmitted over insecure protocols like HTTP or FTP, representing a security vulnerability.
    - Filter Used: http contains "password" or ftp
7. High Traffic Protocol Identification:
    - Purpose: To filter for traffic involving protocols known for high bandwidth consumption, such as video streaming traffic.
    - Filter Used: rtp or udp.port == 1935 (RTMP)
8. DNS Error Logging:
    - Purpose: To identify Domain Name System (DNS) queries that resulted in a non-zero response code, indicating query failures (e.g., "Not Found").
    - Filter Used: dns and dns.response.code != 0

APPENDIX B: RAW DATA AND CAPTURE FILE METADATA

This appendix provides a detailed inventory of the key packet capture files (.pcapng) generated and analysed for the core findings of this project, ensuring the reproducibility of the study.

Capture File Inventory

- Capture ID: LAN-A-001

- Activity/Purpose: Standard Web Browsing (HTTP, HTTPS, DNS)
- Total Packets: 87,452
- File Size (MB): 15.2
- Capture Duration (Minutes): 10
- Key Findings Supported: Protocol Hierarchy analysis, DNS query latency.

- Capture ID: LAN-A-002
  - Activity/Purpose: Large File Transfer (FTP/SMB)
  - Total Packets: 125,980
  - File Size (MB): 48.9
  - Capture Duration (Minutes): 5
  - Key Findings Supported: TCP Window Size and Sequence/Acknowledgment number flow analysis.
- Capture ID: LAN-B-003
  - Activity/Purpose: VoIP/Media Streaming (RTP/RTCP)
  - Total Packets: 65,120
  - File Size (MB): 22.1
  - Capture Duration (Minutes): 7
  - Key Findings Supported: Jitter and packet loss metrics, high UDP volume.

APPENDIX C: HARDWARE AND SOFTWARE SPECIFICATIONS

This appendix formally documents the specific technical environment used to conduct the packet capture and subsequent analysis.

I. Analysis Workstation (Laptop)

- Model: HP Probook 6465b

- Operating System: Windows 10 Pro
- RAM: 4 GB
- Network Interface Card (NIC): Intel Dual Band Wireless-AC 7260

II. Network Environment

- Router/Gateway: TP-Link Archer C7
- Network Topology: Switched Local Area Network (LAN)
- Capture Method: Promiscuous Mode via Switched Port Analyzer (SPAN) / ARP Spoofing

III. Software Used

- Wireshark: Version 4.4.9. Primary tool for packet capture and deep-packet analysis (DPI).
- tshark: Version 4.4.9. Command-line tool used for scripting and data extraction.
- Microsoft Word: Office 365 for Document preparation and formatting.