

INTERNSHIP PROJECT REPORT
on
Network Traffic Analysis Using Wireshark
Internship Domain: Ethical Hacking

Submitted in partial fulfilment of the requirements for the completion of Ethical Hacking

From:

Name: Pavan Kumar V

Intern ID: ITID4887

Submitted to:

Inlighnx Global Pvt Ltd (Inlighnx Tech)

From: 1/10/2025 to 01/01/2026

Duration: 3 months

1. Abstract:

Network security is essential for safeguarding digital communications. Analysing network traffic is vital for comprehending how data moves within a network and for identifying potentially harmful activities. This project centres on examining network traffic through Wireshark, a popular tool for packet sniffing and analysis. In this undertaking, live network packets are collected and examined to investigate various network protocols, including TCP, UDP, HTTP, DNS, and ICMP. The project also highlights how unencrypted information can reveal confidential data and how attackers might take advantage of inadequate network setups. By scrutinizing packet behaviour and protocol statistics, it becomes possible to pinpoint potential security vulnerabilities. This project offers practical experience in packet capture, filtering, protocol analysis, and traffic scrutiny, which are key competencies for cybersecurity experts.

2. INTRODUCTION

2.1 Network Traffic Analysis

Network traffic analysis is the process of capturing, inspecting, and analyzing network packets to understand communication patterns, detect anomalies, and identify security threats. It is widely used in network troubleshooting, intrusion detection, and forensic investigations.

2.2 About Wireshark

Wireshark is a widely used **network protocol analyzer** that allows users to capture and inspect network traffic in real time. It helps in understanding how data packets flow across a network. Wireshark supports hundreds of protocols, including TCP, UDP, HTTP, DNS, and ICMP. The tool displays detailed packet information such as headers, payloads, and protocol fields. It is commonly used by network administrators, cybersecurity professionals, and ethical hackers. Wireshark provides powerful filtering options to isolate specific traffic. It is an open-source and cross-platform tool. Wireshark is widely used for troubleshooting network issues. It also plays an important role in security analysis and forensic investigations. Overall, Wireshark is an essential tool for network traffic analysis.

2.3 Importance of the Project

- Helps understand real network communication
- Used in Security Operations Centres (SOC)
- Detects suspicious or malicious activities
- Improves cybersecurity skills

3.OBJECTIVES & SCOPE

3.1 Objectives

- To capture live network traffic using Wireshark
- To analyze common network protocols
- To identify potential security risks
- To understand packet-level communication

3.2 Scope of the Project

- Analysis of TCP, UDP, HTTP, DNS, and ICMP traffic
- Identification of unencrypted data
- Study of protocol hierarchy and traffic flow
- Basic anomaly detection

This project is limited to basic traffic analysis and does not include advanced malware reverse engineering.

4.SYSTEM REQUIREMENTS

4.1 Hardware Requirements

- Laptop / Desktop Computer
- Minimum 4 GB RAM
- Network Interface Card

4.2 Software Requirements

- Operating System: Windows / Linux
- Wireshark Tool
- Internet Browser
- Npcap Driver (for Windows)

4.3 Tools Used

- Wireshark

- Command Prompt / Terminal
- Web Browser

5.METHODOLOGY

5.1 Installation of Wireshark

Wireshark was installed from the official website. During installation, Npcap was installed to enable packet capturing.

5.2 Packet Capture Process

1. Launch Wireshark
2. Select active network interface (Wi-Fi/Ethernet)
3. Start packet capture
4. Perform internet activities (browsing, ping, login)
5. Stop capture after few minutes

5.3 capturing packets

Here we start sending and analysing packet using TryHackMe website. Upon sending a connection request to the TryHackMe platform, a well-organized and secure communication process is activated between the client and the server. When we enter the website name it converts the domain name into the server IP address and server send the connection request to the client.

No.	Time	Source	Destination	Protocol	Length	Info
156	3.172976495	192.168.219.128	192.168.219.2	DNS	75	Standard query 0x7020 A ads.mozilla.org
157	3.173081363	192.168.219.128	192.168.219.2	DNS	75	Standard query 0x651a AAAA ads.mozilla.org
165	3.194401389	192.168.219.2	192.168.219.128	DNS	221	Standard query response 0x651a AAAA ads.mozilla.org CNAME mc.pr
168	3.196066372	192.168.219.2	192.168.219.128	DNS	144	Standard query response 0x7020 A ads.mozilla.org CNAME mc.pr
214	3.974343589	192.168.219.128	192.168.219.2	DNS	74	Standard query 0x99a8 HTTPS www.google.com
215	3.974671081	192.168.219.128	192.168.219.2	DNS	74	Standard query 0x99a5 A www.google.com
216	3.974696283	192.168.219.128	192.168.219.2	DNS	74	Standard query 0xbfa7 AAAA www.google.com
217	3.998585384	192.168.219.2	192.168.219.128	DNS	99	Standard query response 0x99a8 HTTPS www.google.com HTTPS
218	3.999383583	192.168.219.2	192.168.219.128	DNS	99	Standard query response 0x59a5 A www.google.com A 142.250.18
219	4.0004448615	192.168.219.2	192.168.219.128	DNS	102	Standard query response 0x8fa7 AAAA www.google.com AAAA 2404
284	19.100665199	192.168.219.128	192.168.219.2	DNS	77	Standard query 0xa2cc HTTPS fonts.gstatic.com
297	19.103783810	192.168.219.128	192.168.219.2	DNS	77	Standard query 0xdde1 A fonts.gstatic.com
298	19.103834575	192.168.219.128	192.168.219.2	DNS	77	Standard query 0x9e5 AAAA fonts.gstatic.com
313	19.121626419	192.168.219.128	192.168.219.2	DNS	84	Standard query 0xb676 A www.googleadservices.com
314	19.121723151	192.168.219.128	192.168.219.2	DNS	84	Standard query 0xdb79 HTTPS www.googleadservices.com
315	19.121811439	192.168.219.128	192.168.219.2	DNS	84	Standard query 0xa74 AAAA www.googleadservices.com

Fig 5.3.1 DNS data Packets

This process commences with the TCP three-way handshake, a crucial method employed to create a dependable connection. The handshake entails the interchange of SYN, SYN-ACK, and

ACK packets, verifying that both the client and server are prepared to send data and that the connection settings are aligned. After the TCP connection is successfully formed, ongoing data transmission occurs between the client and the server. Throughout this stage, numerous packets are exchanged, including TCP, UDP, and various protocol-specific packets. TCP packets play a key role in ensuring dependable data delivery by maintaining packet order, performing error checks, and resending packets if any are lost. UDP packets may also be present, generally utilized for swift, connectionless data transfer where minimal delay is desired.

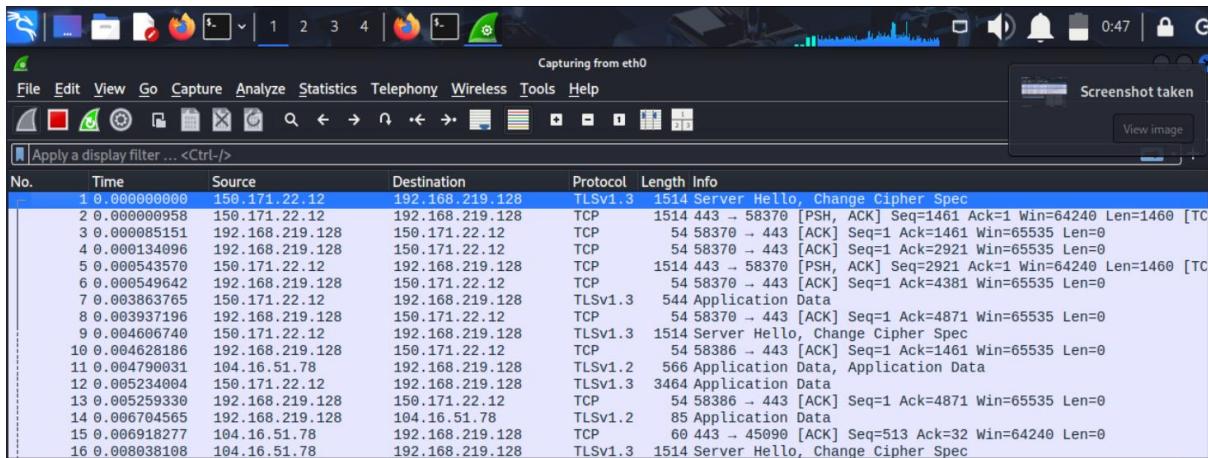


Fig 5.3.2 TCP data Packets

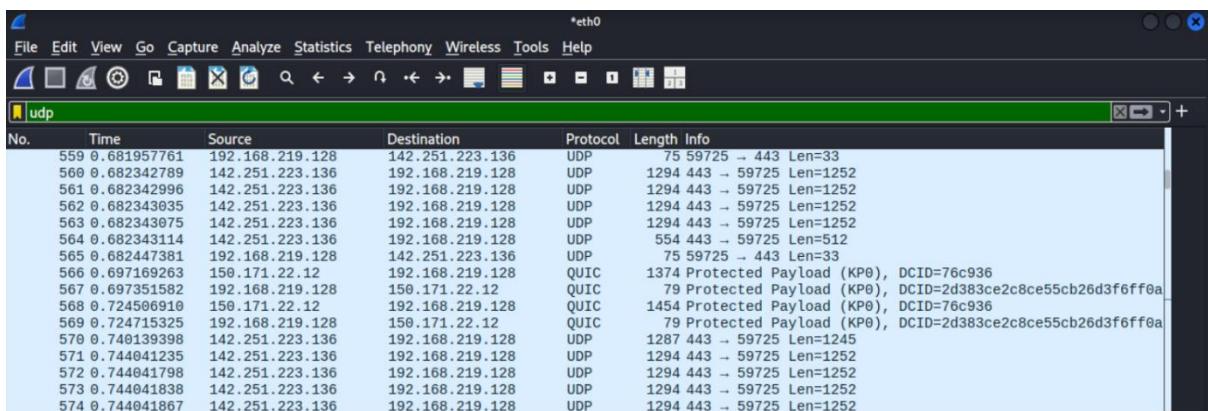


Fig 5.3.3 UDP data Packets

Here we are getting TCP and UDP packets after sending the connection request to the server with different secure encryption protocols. After sending and receiving TCP and UDP packets, the communication between the client and server continues in an organized manner to ensure successful data exchange. TCP packets maintain reliability through acknowledgments, sequencing, and retransmissions, while UDP packets enable faster, connectionless data transfer where required. These packets carry application-level data as well as control information necessary for session management. Secure protocols may then encrypt the transmitted data to

protect it from unauthorized access. Overall, this packet exchange ensures efficient, reliable, and secure network communication

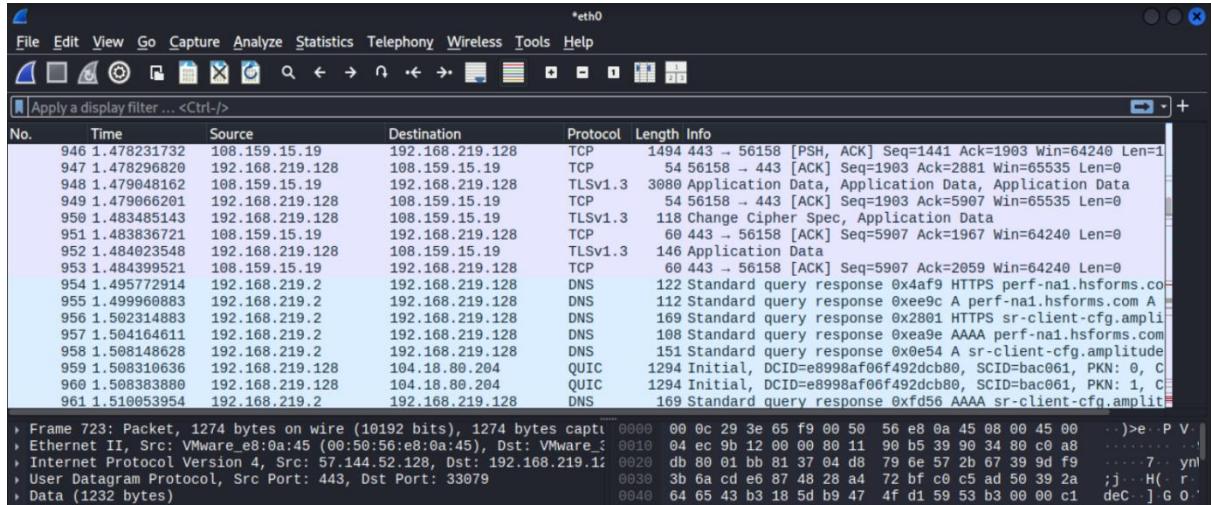


Fig 5.3.4 TLSv1.3 Secure encryption Protocol

This entire process takes place not only when the user logs in but also when they communicate with the TryHackMe platform or send text messages. Every action taken by the user creates encrypted data packets that are safely sent and acknowledged. The significance of encryption and protocol layering in contemporary network security is highlighted by packet analysis using programs like Wireshark, which aids in understanding how secure client-server communication is created, maintained, and shielded against interception. **TLS v1.3 (Transport Layer Security version 1.3)** is the **latest and most secure encryption protocol** used to protect data during communication over the internet.

It ensures that data exchanged between a client and a server remains **confidential, authentic, and tamper-proof**. TLS 1.3 improves security by removing outdated cryptographic algorithms and reducing the number of handshake steps. This results in **faster connection establishment** and **lower latency** compared to older versions like TLS 1.2. It is widely used in **HTTPS websites, online banking, cloud services, and secure applications**. Overall, TLS 1.3 provides stronger encryption, better performance, and enhanced privacy for modern network communications.

Then, I tested the http connection using the **TestPHP website**, which is an HTTP-based web application commonly used by ethical hackers for practice and learning. Since it uses the HTTP protocol, the data is transmitted in plaintext without encryption. This makes it suitable for observing request and response packets during network analysis. Ethical hackers use this website to understand web vulnerabilities and traffic behavior. The connection helps in analyzing HTTP

methods, headers, and server responses. It is widely used for educational and security testing purposes.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	44.228.249.3	192.168.219.128	TCP	60	443 → 40802 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
2	0.000000854	44.228.249.3	192.168.219.128	TCP	60	443 → 40804 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
3	0.000000914	44.228.249.3	192.168.219.128	TCP	60	443 → 40816 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
4	0.051188073	192.168.219.128	86.107.100.245	NTP	90	NTP Version 4, client
5	0.090195336	86.107.100.245	192.168.219.128	NTP	90	NTP Version 4, server
6	0.251295211	44.228.249.3	192.168.219.128	TCP	60	443 → 40828 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
7	0.552454950	192.168.219.128	44.228.249.3	HTTP	393	GET / HTTP/1.1
8	0.553218652	44.228.249.3	192.168.219.128	TCP	60	80 → 34814 [ACK] Seq=1 Ack=340 Win=64240 Len=0
9	0.822740744	44.228.249.3	192.168.219.128	HTTP	2613	HTTP/1.1 200 OK (text/html)
10	0.823320714	192.168.219.128	44.228.249.3	TCP	54	34814 → 80 [ACK] Seq=340 Ack=2560 Win=65535 Len=0
11	2.812970196	192.168.219.128	44.228.249.3	TCP	54	34796 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
12	2.813867896	44.228.249.3	192.168.219.128	TCP	60	[TCP ACKed unseen segment] 80 → 34796 [ACK] Seq=1 Ack=2 Win=
13	6.400840993	192.168.219.128	44.228.249.3	TCP	54	34830 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
14	6.401762761	44.228.249.3	192.168.219.128	TCP	60	[TCP ACKed unseen segment] 80 → 34830 [ACK] Seq=1 Ack=2 Win=
15	11.005146970	192.168.219.128	44.228.249.3	TCP	54	[TCP Keep-Alive] 34814 → 80 [ACK] Seq=339 Ack=2560 Win=65535
16	12.029594556	192.168.219.128	44.228.249.3	TCP	54	[TCP Keep-Alive] 34814 → 80 [ACK] Seq=339 Ack=2560 Win=65535

fig 5.3.5 HTTP packets

I captured HTTP packets in Wireshark during the network traffic analysis process. These packets show the communication between the client and the web server using the HTTP protocol. The captured data includes HTTP request methods such as GET and POST along with server responses. Since HTTP is not encrypted, the request headers and transmitted data are visible in plain text. This makes it easier to analyze URLs, parameters, and server responses. Observing HTTP packets helps in understanding how web communication works at the application layer.

No.	Time	Source	Destination	Protocol	Length	Info
1630	16:17.9302993..	192.168.219.2	192.168.219.128	DNS	98	Standard query response 0x78a9 AAAA GOOGLE.COM AAAA 2404:680
1631	16:17.9332931..	192.168.219.128	142.251.221.174	ICMP	98	Echo (ping) request id=0x0001, seq=1/256, ttl=64 (reply in
1632	16:18.0404873..	142.251.221.174	192.168.219.128	ICMP	98	Echo (ping) reply id=0x0001, seq=1/256, ttl=128 (request
1633	16:18.0414521..	192.168.219.128	192.168.219.2	DNS	88	Standard query 0xb6b1 PTR 174.221.251.142.in-addr.arpa
1634	16:18.0644226..	192.168.219.2	192.168.219.128	DNS	128	Standard query response 0xb6b1 PTR 174.221.251.142.in-addr.a
1635	16:18.9334380..	192.168.219.128	142.251.221.174	ICMP	98	Echo (ping) request id=0x0001, seq=2/512, ttl=64 (reply in
1636	16:18.9648768..	142.251.221.174	192.168.219.128	ICMP	98	Echo (ping) reply id=0x0001, seq=2/512, ttl=128 (request
1637	16:19.9354956..	192.168.219.128	142.251.221.174	ICMP	98	Echo (ping) request id=0x0001, seq=3/768, ttl=64 (reply in
1638	16:19.9584239..	142.251.221.174	192.168.219.128	ICMP	98	Echo (ping) reply id=0x0001, seq=3/768, ttl=128 (request
1639	16:20.9380584..	192.168.219.128	142.251.221.174	ICMP	98	Echo (ping) request id=0x0001, seq=4/1024, ttl=64 (reply in
1640	16:20.9623580..	142.251.221.174	192.168.219.128	ICMP	98	Echo (ping) reply id=0x0001, seq=4/1024, ttl=128 (request
1641	16:21.9393592..	192.168.219.128	142.251.221.174	ICMP	98	Echo (ping) request id=0x0001, seq=5/1280, ttl=64 (reply in
1642	16:21.9628028..	142.251.221.174	192.168.219.128	ICMP	98	Echo (ping) reply id=0x0001, seq=5/1280, ttl=128 (request
1643	16:22.3195921..	192.168.219.128	57.144.53.32	TLSv1.3	123	Application Data
1644	16:22.3217465..	57.144.53.32	192.168.219.128	TCP	60	443 → 39072 [ACK] Seq=2474 Ack=2453 Win=64240 Len=0
1645	16:22.5340619..	57.144.53.32	192.168.219.128	TLSv1.3	125	Application Data

Fig 5.3.6 ICMP Packets

Using Ping to ping the google.com domain system generated an iterative process to retrieve the associated IP address, first determining the address via DNS resolution, then sending Echo Request via ICMP packets to the remote host, and receiving ICMP Echo Replies from the remote server, thereby verifying the remote server was reachable via IP address using ICMP. The subsequently captured packets in Wireshark contained the following additional details: source and destination

addresses (IP), a packet's sequence number and Time To Live (TTL) values. This capability to capture and analyse ICMP packets provides network connectivity verification, identifies latency in the network, and is an example of 'connectless communications' – ICMP does not provide a session; instead ICMP uses 'connectionless communication'. Therefore, it should be noted that the above captured packets provide examples of request-to-response communication. In general, ICMP packet analysis is a common method for diagnosing network connectivity and performance issues.

6. Conclusion

This project successfully demonstrated how network traffic can be captured and analyzed using Wireshark. It provided practical exposure to real network packets and highlighted the importance of secure communication. The project enhanced understanding of protocols and network security threats.

6.1 Learning Outcomes

- Hands-on packet analysis
- Understanding of network protocols
- Awareness of cybersecurity risks
- Practical security investigation skills

6.2 Future Scope

- Malware traffic analysis
- Integration with IDS/IPS
- Advanced forensic analysis
- SIEM-based traffic monitoring
