**Comprehensive Network Traffic Analysis Using Wireshark**

# Objective

This project aims to capture, analyze, and understand various types of network traffic using Wireshark. This includes capturing DNS traffic, analyzing secure web traffic using TLS, and studying the distribution of different protocols within a network. The project highlights how Wireshark can be used to monitor network health, identify potential issues, and ensure secure communications.

## 1. DNS Query Traffic Analysis

In this step, I captured and analyzed DNS traffic using Wireshark. The screenshot shows a Follow UDP Stream view, highlighting a DNS query to fonts.gstatic.com. This domain is part of Google's content delivery network (CDN), which provides fonts and other assets for websites.

**Key Insights**:  
- DNS traffic is captured and analyzed to understand how domains are resolved to IP addresses.  
- The stream shows the complete DNS query-response cycle, including the resolution of fonts.gstatic.com.

**Tools Used:**  
- Wireshark for live packet capture.  
- UDP Stream Analysis to follow the DNS communication flow.

## 2. Detailed DNS Query and Response for Mozilla Services

This screenshot captures another DNS stream, this time for contile.services.mozilla.com. The DNS query and response are shown in the stream, revealing that the domain resolves to an IP address belonging to Mozilla's content delivery system, likely hosted on AWS.

**Key Insights:**  
- The DNS query successfully resolves the Mozilla domain to 34.117.188.166, indicating proper DNS functioning.  
- The traffic provides insights into Mozilla’s use of external services (AWS DNS) for domain resolution.

**Tools Used:**  
- Follow UDP Stream to track DNS query and response.  
- Wireshark for real-time DNS traffic capture.

## 3. General DNS Traffic and ICMP Error Messages

I captured a broader view of DNS traffic and ICMP error messages here. The traffic shows successful DNS queries for domains like contile.services.mozilla.com and ICMP Destination Unreachable messages, which indicate that some queries failed due to network issues.

**Key Insights**:  
- Successful and failed DNS traffic are both visible.  
- ICMP messages provide information on network issues, such as unreachable destinations, helping diagnose potential connectivity problems.

**Tools Used:**  
- DNS and ICMP Filters applied in Wireshark to isolate specific types of traffic.

## 4. OCSP (Online Certificate Status Protocol) Traffic Analysis

In this part of the project, I captured OCSP traffic, which is used to verify the validity of SSL certificates during HTTPS connections. The OCSP requests and responses are part of the communication that ensures web certificates haven’t been revoked, helping secure the web browsing experience.

**Key Insights:**  
- OCSP traffic confirms that SSL certificates are being validated in real-time during HTTPS connections.  
- This is essential for ensuring secure communications and verifying that certificates used in HTTPS sessions haven’t been compromised.

**Tools Used:**  
- Wireshark to capture and filter OCSP traffic.  
- Protocol Filters applied to focus on OCSP communication.

## 5. TCP Handshake and TLSv1.3 Encrypted Traffic

This screenshot captures the TCP handshake and encrypted traffic over TLSv1.3. It shows the three-way handshake (SYN, SYN-ACK, ACK) necessary to establish a TCP connection, followed by encrypted data transfer using TLSv1.3. This is a critical part of secure HTTPS communication.

**Key Insights:**  
- The TCP handshake establishes reliable communication between client and server.  
- TLSv1.3 encrypts the session, ensuring the data remains confidential.

**Tools Used:**  
- TCP Filters in Wireshark to capture and analyze the handshake and encryption process.  
- Follow TCP Stream to monitor secure communication over HTTPS.

## 6. Encrypted TCP Stream (TLSv1.3)

This screenshot shows the encrypted TCP stream that follows the initial handshake. The data is unreadable due to TLSv1.3 encryption, which protects the session's content from unauthorized access.

**Key Insights:**  
- The encryption of this stream demonstrates how secure protocols like TLSv1.3 protect sensitive information during transmission.  
- Only the initial handshake information (such as headers) is visible while the payload is encrypted.

**Tools Used:**  
- Wireshark to follow encrypted streams.  
- TLSv1.3 Analysis for observing secure communication.

## 7. Protocol Hierarchy Statistics

This final screenshot provides an overview of the Protocol Hierarchy Statistics, showing the distribution of different protocols in the captured traffic. This view helps analyze the types and traffic volume on the network.

**Key Insights:**  
- IPv4 and IPv6 traffic are observed, indicating a dual-stack network environment.  
- DNS traffic is prominent and carried over UDP and ICMP messages for network diagnostics.

**Tools Used:**  
- Wireshark Protocol Hierarchy to analyze the distribution of protocols in the captured session.

# Conclusion

Through this comprehensive network traffic analysis using Wireshark, I captured, analyzed, and understood various forms of network communication, from DNS queries to secure TCP connections. Each step of the analysis highlights key aspects of network behavior, such as domain resolution, secure communications, and protocol distribution, providing valuable insights into how network traffic flows and is secured in modern networks.

**Skills Demonstrated:**  
- Packet capture and analysis using Wireshark.  
- Protocol analysis including DNS, TCP, ICMP, and OCSP.  
- Secure communication analysis using TLSv1.3.  
- Network troubleshooting through ICMP error analysis.