Task 5: Network Traffic Analysis with Wireshark

Deliverable: The full packet capture file can be found in this repository: elevate_lab_task5.pcapng.

Objective

The goal of this task was to gain hands-on experience with packet analysis using Wireshark. This involved capturing live network packets from a local machine and using display filters to isolate and identify the roles of fundamental internet protocols.

Tool Used

- **Packet Analyzer:** Wireshark
- **Operating System:** Kali Linux

Process

1. **Capture:** Wireshark was used to start a live packet capture on the active network interface (eth0).
2. **Traffic Generation:** To generate a variety of traffic, I browsed several websites (both HTTP and HTTPS) and used the ping command in the terminal.
3. **Analysis:** After stopping the capture, display filters were used to isolate and examine specific protocols.
4. **Export:** The final capture was saved as a .pcapng file for submission.

Protocols Identified and Analyzed

The capture revealed a wide variety of network traffic. The following four protocols were isolated and analyzed as key examples of network communication.

- **DNS (Domain Name System)**
- **HTTP (Hypertext Transfer Protocol)**
- **TCP** (Transmission **Control Protocol)**
- **ICMP (Internet Control Message Protocol)**

Summary of Findings & Packet Details

1. DNS (Domain Name System)

- **Finding:** The capture showed that before any web connection was made, the host machine performed multiple DNS lookups to convert website names (e.g., google.com) into IP addresses. This confirms that DNS is a foundational step for most internet activity.

- **Packet Detail (from dns_capture.png):** The first packet in the screenshot is a **DNS Standard Query**.
  - **Source:** 192.168.29.174 (My Kali VM)
  - **Destination:** 192.168.29.1 (The local network's DNS resolver/router)
  - **Info:** The packet is asking for the A record (IPv4 address) for google.com. The subsequent packet is the resolver's response, providing the IP address 142.251.43.110.

2. HTTP (Hypertext Transfer Protocol)

- **Finding:** The analysis of HTTP traffic successfully captured an unencrypted web request. It was possible to see the exact request sent by the browser and the "200 OK" response from the server, which contained the website's HTML content.
- **Packet Detail (from http_capture.png):** A key packet in this capture is the **HTTP GET request**.
  - **Source:** 192.168.29.174 (My Kali VM)
  - **Destination:** 188.184.67.127 (The web server)
  - **Info:** The packet contains the command GET / HTTP/1.1, which is the browser's request for the main page of the website. This demonstrates the client-server request/response model of HTTP.

3. TCP (Transmission Control Protocol)

- **Finding:** The TCP filter revealed the underlying connection-oriented nature of web traffic. The classic three-way handshake (SYN, SYN, ACK, ACK) was observed, which establishes a reliable connection before any application data (like HTTP) is exchanged.
- **Packet** Detail (from **tcp_capture.png):** A highlighted packet in the screenshot shows a **TCP segment** with the [ACK] flag set.
  - **Source Port:** 54866 (An ephemeral port on my machine)
  - **Destination Port:** 443 (The standard port for HTTPS on the server)
  - **Info:** This [ACK] packet is part of an ongoing conversation, acknowledging the receipt of data from the server. This confirms TCP's role in ensuring reliable data delivery.

4. ICMP (Internet Control Message Protocol)

- **Finding:** Filtering for ICMP traffic clearly isolated the packets generated by the ping command. This demonstrated a direct and simple form of network communication used for diagnostics.
- **Packet Detail (from icmp_capture.png):** The first packet in the screenshot is an **ICMP Echo (ping) request**.
  - **Source:** 192.168.29.174 (My Kali VM)
  - **Destination:** 8.8.8.8 (Google's public DNS server)
  - **Info:** This packet is a direct request to the destination, asking "Are you there?" The subsequent "Echo (ping) reply" packet from 8.8.8.8 is the server's response, confirming that it is online and reachable.

Interview Questions & Answers

1. What is Wireshark used for?

Wireshark is a powerful network protocol analyzer used for network troubleshooting, analysis, software and communications protocol development, and education. It allows users to capture and interactively browse the traffic running on a computer network in real-time.

2. What is a **packet?**

A packet is a small segment of a larger message that is sent over a network. Data is broken down into packets to be sent more efficiently and reliably. Each packet contains a portion of the user data (the payload) plus a header with control information, such as the source and destination IP addresses, port numbers, and protocol type.

3. How to filter packets in Wireshark?

You can filter packets by using the "Apply a display filter" bar at the top of the Wireshark window. You can type in the name of a protocol (e.g., dns, http, tcp) or create more complex filters based on IP addresses (ip.addr == 8.8.8.8), ports (tcp.port == 443), or a combination of conditions.

4. What is the difference between TCP and UDP?

TCP (Transmission Control **Protocol)** is connection-oriented and reliable. It establishes a connection via a three-way handshake, ensures packets are delivered in order, and re-transmits lost packets. It's used for services that require high reliability, like web browsing (HTTP/S) and email (SMTP). **UDP** (User Datagram **Protocol)** is connectionless and unreliable. It's much faster because it sends data without establishing a connection or checking for errors. It's used for services where speed is more important than reliability, such as video streaming, online gaming, and DNS.

5. What is a DNS query packet?

A DNS query packet is a request sent from a client (like your computer) to a DNS server. The packet essentially asks the question, "What is the IP address associated with this domain name (e.g., https://www.google.com/search?q=www.google.com)?" The server then responds with a DNS reply packet containing the answer.

6. How **can packet capture help in troubleshooting?**

Packet capture is essential for troubleshooting because it allows network administrators to see exactly what is happening on the network at the lowest level. It can help diagnose slow network performance by identifying retransmissions, pinpoint the source of an error by showing malformed packets, and identify security issues by revealing unauthorized or suspicious traffic.

7. What is a protocol?

A protocol is a set of rules and conventions that governs how data is exchanged between devices on a network. It defines the format, timing, sequencing, and error control for communication. Examples include IP, TCP, UDP, HTTP, and DNS.

8. Can Wireshark decrypt encrypted traffic?

Generally, no. If traffic is encrypted with modern protocols like TLS (used in HTTPS), Wireshark will only show you gibberish. However, if you have access to the appropriate decryption keys (specifically, the pre-master