**Network Traffic Analysis Report: A Practical Study**

- **Date of Analysis:** [12-08-2025]
- **Analyst:** [Vijay]
- **Packet Capture File:** capturing_packets.pcapng

**1. Executive Summary**

This report documents the analysis of network traffic captured from a Linux virtual machine over a [e.g., 3-minute] period. The primary objective was to dissect live network activity to identify and understand the behavior of core internet protocols. The analysis revealed a typical traffic pattern for a standard user session, dominated by DNS, TCP, and HTTPS for web browsing, alongside diagnostic ICMP traffic. Key findings include the resolution of [e.g., cnn.com] via DNS, the establishment of a TCP connection to [e.g., GitHub's IP address], and the subsequent encrypted TLS handshake.

**2. Methodology**

A controlled network environment was established using a Linux (Ubuntu) virtual machine hosted on VMware. Wireshark was utilized to capture all incoming and outgoing packets from the [e.g., eth0] interface. To generate meaningful data, the following specific actions were performed:

- **Connectivity Test:** The ping utility was used to send 10 ICMP echo requests to google.com.
- **DNS Resolution:** The dig command was used to query the IP address for [e.g., cnn.com].
- **Web Browsing:** A web browser was used to access [e.g., http://info.cern.ch/] and https://github.com.

The resulting capture file contains [e.g., 2,548] packets and was analyzed by applying display filters in Wireshark to isolate and inspect specific protocol data.

**3. Analysis of Identified Protocols**

The captured traffic was dissected to examine the behavior of individual protocols.

**3.1. DNS (Domain Name System)**

- **Observation:** DNS queries were the initial step for nearly all web-related communication. A standard query for [e.g., A record for cnn.com] was observed.
- **Specifics:**
    - **Source IP (Client):** [Find your VM's IP address]
    - **Destination IP (DNS Server):** [Find the IP of the DNS server, e.g., 8.8.8.8 or your router's IP]
    - **Transaction Details:** Packet [e.g., #112] contained the DNS query for cnn.com, and the response arrived in packet [e.g., #115] providing the IP address [e.g., 151.101.65.67]. This communication occurred over UDP port 53 as expected.
- **Wireshark Filter:** dns
- **Note :** I am not providing my IP here .I use some random IP.

**3.2. ICMP (Internet Control Message Protocol)**

- **Observation:** ICMP traffic was generated exclusively by the ping command to test network latency and reachability.
- **Specifics:**
    - A sequence of 10 "Echo (ping) request" and "Echo (ping) reply" pairs was identified between our host [Your VM's IP] and Google's server [Find the IP address ping resolved to, e.g., 142.250.190.78].
    - The average round-trip time calculated from the timestamps was approximately [e.g., 15.2 ms]. The seq= number in the packet details incremented from 1 to 10 for each request/reply pair.
- **Wireshark Filter:** icmp

**3.3. TCP (Transmission Control Protocol)**

- **Observation:** TCP was the workhorse protocol, responsible for establishing reliable connections for HTTPS traffic.
- **Specifics:**
    - A clear three-way handshake (SYN, SYN-ACK, ACK) was observed initiating a connection to [e.g., GitHub's IP at 140.82.121.4] on port 443.
    - **Example:** Packet [e.g., #250] was the initial SYN packet from our host. Packet [e.g., #252] was the SYN-ACK response from the server, and packet [e.g., #253] was the final ACK from our host, completing the handshake.
- **Wireshark Filter:** tcp.port == 443

**3.4. HTTPS (HTTP Secure) / TLS (Transport Layer Security)**

- **Observation:** The vast majority of web traffic was encrypted using HTTPS. After the TCP handshake, a TLS handshake was initiated to set up a secure channel.
- **Specifics:**
    - Following the TCP handshake to [e.g., 140.82.121.4], a "Client Hello" packet was sent to begin the TLS negotiation. The server responded with a "Server Hello," followed by certificate exchange and key negotiation.
    - All subsequent application data (the actual HTTP requests and responses) was encrypted and labeled in Wireshark as "Application Data" under the TLSv1.3 protocol. It was impossible to read the content, demonstrating the effectiveness of the encryption.
- **Wireshark Filter:** tls

**4. Conclusion**

This practical analysis provided significant insight into the orchestration of network protocols. The exercise moved beyond theoretical knowledge to hands-on identification of DNS resolution, TCP connection management, secure session negotiation via TLS, and network diagnostics using ICMP. It underscored the logical sequence of network events that are required for even a simple action like

accessing a secure website. This project has solidified my packet analysis skills and provided a foundational, real-world understanding of network communication.