🛡️ Task 5 — Capture and Analyze Network Traffic Using Wireshark

Author: Yuvraj Yadav
Date: 29-10-2025
Platform Used: macOS
Tool Used: Wireshark (v4.x)
Network Interface: Wi-Fi (en0)

1) Objective

The objective of this task was to perform **live packet capture and network traffic analysis** using **Wireshark** on macOS.
The goal was to identify and study common network protocols — such as **DNS, HTTP, HTTPS, ICMP, and ARP** — and understand how data flows between systems within a TCP/IP model.

2) Tools and Environment Setup

Operating System: macOS (Intel-based)
Tool Used: Wireshark — GUI-based network protocol analyzer
Network Adapter: Wi-Fi (en0) — connected to local home network

Installation Method:

```
brew install wireshark
```

(Alternatively, Wireshark can be installed via the official DMG package.)

Capture Interface:
Selected `Wi-Fi: en0` to monitor live packets over the active wireless interface.

3) Steps Performed

Step 1 – Start Packet Capture

1. Opened **Wireshark**.
2. Selected **Wi-Fi (en0)** as the capture interface.
3. Clicked **Start Capturing Packets** (blue shark fin icon).
4. Performed some online activity (visited a website, pinged google.com) to generate traffic.

Step 2 – Stop Capture

After approximately 1 minute of capture:

- Clicked **Stop (red square icon)**.
- The capture contained several hundred packets across multiple protocols.
- File saved as: `network_capture_yuvraj.pcapng`

Step 3 – Apply Filters and Analyze Protocols

Used display filters to focus on specific protocol types:

Protocol	**Filter Command**	**Purpose**
DNS	`dns`	View name resolution queries & responses
HTTP	`http`	Analyze unencrypted web requests
HTTPS	`tls` or `ssl`	View encrypted web traffic metadata
ICMP	`icmp`	Monitor ping requests and responses
ARP	`arp`	Identify local network device discovery
TCP	`tcp`	Inspect connection establishment & data transfer

Each filter revealed a distinct network layer behavior.

Step 4 – Identify Protocols

Observed the following during analysis:

Layer	Protocol	Description
Application	**HTTP / HTTPS**	Web browsing requests and secure connections
Application	**DNS**	Domain name lookups for website IPs
Network	**ICMP**	Ping messages to test connectivity
Data Link	**ARP**	MAC address resolution within the local network
Transport	**TCP / UDP**	Session management and data transport

Step 5 – Export Capture

The capture file was exported for documentation and submission:

File → Export Specified Packets → network_capture_yuvraj.pcapng

This `.pcapng` file can be opened in Wireshark to replicate all analyses.

4 Packet Analysis and Findings

DNS (Domain Name System)

- Filter: `dns`
- **Description:** DNS requests to resolve domain names like `www.google.com`.
- **Observation:**
    - Protocol: UDP
    - Port: 53
    - Response time: <50 ms
    - Demonstrates how browsers find server IPs.

HTTP (Hypertext Transfer Protocol)

- Filter: `http`
- **Description:** Plain-text web requests and responses.
- **Observation:**
  - GET requests observed for resources (`index.html`, `favicon.ico`)
  - Protocol: TCP (Port 80)
  - Displays unencrypted headers such as `Host`, `User-Agent`.

🔒 HTTPS (Secure HTTP)

- Filter: `tls`
- **Description:** Encrypted communication with web servers.
- **Observation:**
  - Protocol: TCP (Port 443)
  - Encrypted payloads are visible but not readable.
  - TLS handshake details, such as `Client Hello` and `Server Hello, were` captured.

ICMP (Internet Control Message Protocol)

- Filter: `icmp`
- **Description:** Used for diagnostic and connectivity tests (ping).
- **Observation:**
  - Echo Request sent to `8.8.8.8`
  - Echo Reply received (round-trip success)

    - Confirms an active internet connection.

ARP (Address Resolution Protocol)

- Filter: `arp`
- **Description:** Maps IP addresses to MAC addresses on local LAN.
- **Observation:**
    - Requests: "Who has 192.168.1.1?"
    - Replies: "192.168.1.1 is at [MAC Address]"
    - Confirms local device discovery process.

5 Summary of Findings

Protocol	Port	Layer	Packets Observed	Description
DNS	53	Application	40+	Resolves domain names
HTTP	80	Application	20+	Plain web requests
HTTPS	443	Application	50+	Secure, encrypted web traffic
ICMP	—	Network	10	Connectivity check (ping)
ARP	—	Data Link	15	Local address resolution

6 Observations and Learnings

- Learned how **Wireshark captures, decodes, and categorizes network packets** in real time.
- Understood how **different protocols operate across OSI layers**.

- Gained hands-on experience with **protocol filtering**, **header analysis**, and **packet metadata**.
- Observed **unencrypted (HTTP)** vs **encrypted (HTTPS)** communications.
- Understood that **DNS and ARP** are vital background processes for web browsing.
- Realized that **packet capture** is a key technique in **network troubleshooting and cyber forensics**.

7 Outcome

The task achieved its objective of capturing and analyzing live network traffic using Wireshark on macOS.
At least five distinct protocols were identified and analyzed successfully.
The `.pcapng` capture file and screenshots provide practical evidence of traffic analysis and protocol understanding.

Skills Gained:

- Packet capture & analysis
- Use of Wireshark filters
- Protocol identification & classification
- Network troubleshooting basics
- Understanding TCP/IP and OSI models

8 Attachments (for GitHub Repo)

1. **network_capture_yuvraj.pcapng** — packet capture file
2. **screenshots/** folder — includes:
   `01_start_capture.png`, `02_dns_filter.png`, `03_http_traffic.png`, `04_tls_handshake.png`, `05_icmp_ping.png`
3. **report.md** — this detailed documentation

4. **README.md** — concise project summary