

Task 5: Capture and Analyze Network Traffic Using Wireshark

Objective:

Capture live network packets and identify basic protocols and traffic types using Wireshark.

Tool Used:

Wireshark (Version: [Insert version])

Steps Performed:

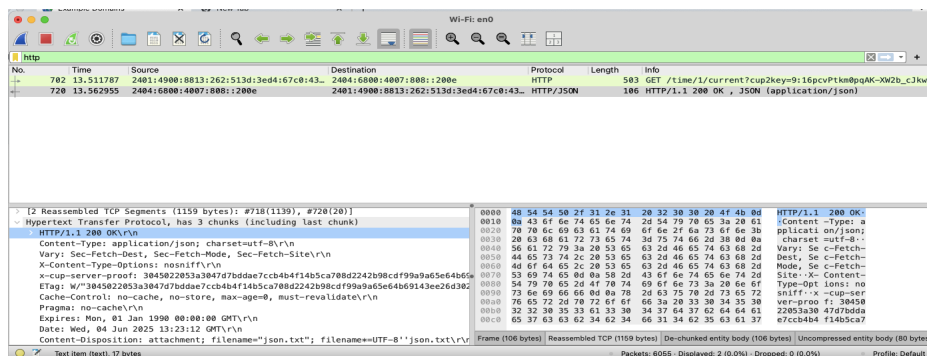
1. **Installed Wireshark** on the system.
2. **Selected active network interface** (Wi-Fi) for live packet capture.
3. Generated traffic by:
 - **Browsing a website** (https://example.com)
 - **Pinging 8.8.8.8** (Google DNS)
 - **Sending email**
4. Stopped capture after approximately **1 minute**.
5. Applied protocol filters in Wireshark (http, dns, tcp, udp, tls, icmp, mdns, etc.)
6. Identified 3+ distinct protocols from the captured traffic.
7. Exported capture file as: task5_traffic_analysis.pcap

Protocols Identified:-

Different Protocols Sample Packet Captured are:-

1. HTTP (HyperText Transfer Protocol):

- The Hyper Text Transport Protocol is a text-based request-response client-server protocol.
- Sample HTTP Packet captured:
 - **Source port: 80**
 - **Destination port: 49414**
 - **Method: GET**
 - **Response Code: 200 OK**
 - **Screenshot:**



2. MDNS(Multicast DNS [mDNS] :

- mDNS is a protocol that facilitates zero-configuration networking, enabling devices on a local network to discover and resolve hostnames without needing a central DNS server.
- mDNS utilizes the User Datagram Protocol (UDP) and specifically port 5353 for communication.
- Sample MDNS packet captured Screenshot:

No.	Time	Source	Destination	Protocol	Length	Info
22420	89.368424	192.168.1.4	224.0.0.251	MDNS	503	Standard query response 0x0000 PTR, cache flush AI
22421	89.368430	fe80::826d:71ff:fe93:16b1	ff02::fb	MDNS	523	Standard query response 0x0000 PTR, cache flush AI
24106	121.401456	192.168.1.4	224.0.0.251	MDNS	503	Standard query response 0x0000 PTR, cache flush AI
24107	121.401463	fe80::826d:71ff:fe93:16b1	ff02::fb	MDNS	523	Standard query response 0x0000 PTR, cache flush AI
24159	122.417053	192.168.1.4	224.0.0.251	MDNS	191	Standard query 0x0000 ANY Android.local, "QM" que
24160	122.417059	fe80::826d:71ff:fe93:16b1	ff02::fb	MDNS	211	Standard query 0x0000 ANY Android.local, "QM" que
24165	122.663033	192.168.1.4	224.0.0.251	MDNS	191	Standard query 0x0000 ANY Android.local, "QM" que
24166	122.663042	fe80::826d:71ff:fe93:16b1	ff02::fb	MDNS	211	Standard query 0x0000 ANY Android.local, "QM" que
24178	122.914059	192.168.1.4	224.0.0.251	MDNS	191	Standard query 0x0000 ANY Android.local, "QM" que
24179	122.914065	fe80::826d:71ff:fe93:16b1	ff02::fb	MDNS	211	Standard query 0x0000 ANY Android.local, "QM" que
24184	123.164476	192.168.1.4	224.0.0.251	MDNS	503	Standard query response 0x0000 PTR, cache flush AI
24185	123.164484	fe80::826d:71ff:fe93:16b1	ff02::fb	MDNS	523	Standard query response 0x0000 PTR, cache flush AI

Frame 22420: 503 bytes on wire (4024 bits), 503 bytes captured (4024 bits) on interface Wi-Fi: en0
 Ethernet II, Src: AmazonTechno_93:16:b1 (80:6d:71:93:16:b1), Dst: Apple_d5:58:3a (7c:00:0d:00:00:00)
 Internet Protocol Version 4, Src: 192.168.1.4, Dst: 224.0.0.251
 User Datagram Protocol, Src Port: 5353, Dst Port: 5353
 Multicast Domain Name System (response)
 Transaction ID: 0x0000
 [Expert Info (Warning/Protocol): DNS response retransmission. Original response
 Flags: 0x8400 Standard query response, No error
 Questions: 0
 Answer RRs: 8
 Authority RRs: 0
 Additional RRs: 5
 Answers

3. TLSv1.3 (Transport Layer Security v1.3):

- TLS 1.3 is a secure communication protocol used to encrypt internet traffic between a client and a server. Packet inspection refers to the process of analyzing network packets to identify specific patterns or characteristics. When it comes to TLS 1.3 packet inspection, it can be challenging due to its encryption and lack of predictable packet structure.
- Sample TLS packet captured Screenshot:

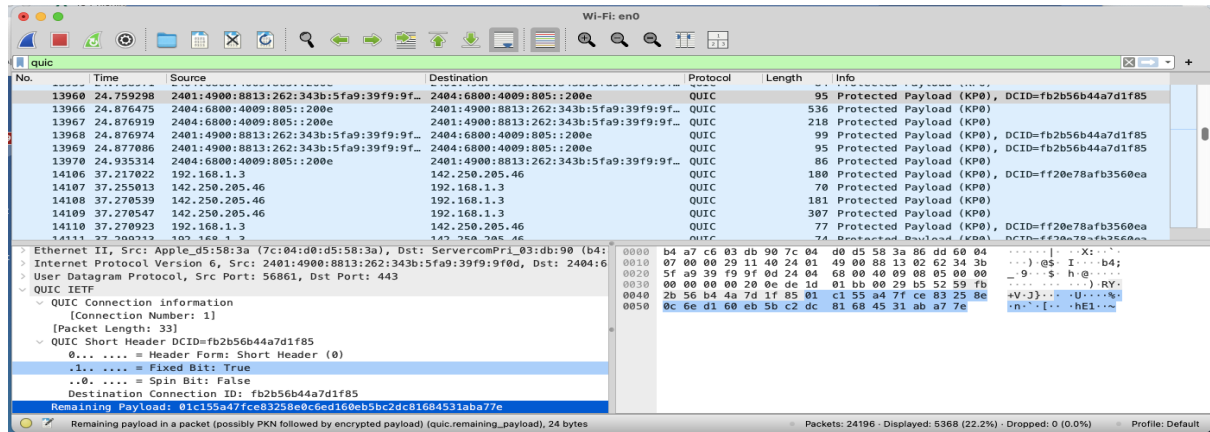
No.	Time	Source	Destination	Protocol	Length	Info
13949	24.710451	2401:4900:8813:262:343b:5fa9:39f9:9f...	2606:4700:9760:f2c2:2376:6:6810:ba29	TLSv1.3	178	Application Data
13950	24.710452	2401:4900:8813:262:343b:5fa9:39f9:9f...	2606:4700:9760:f2c2:2376:6:6810:ba29	TLSv1.3	623	Application Data
13951	24.710452	2401:4900:8813:262:343b:5fa9:39f9:9f...	2606:4700:9760:f2c2:2376:6:6810:ba29	TLSv1.3	1362	Application Data
13953	24.735132	2606:4700:9760:f2c2:2376:6:6810:ba29	2401:4900:8813:262:343b:5fa9:39f9:9f...	TLSv1.3	598	Application Data, Application Data
13954	24.735137	2606:4700:9760:f2c2:2376:6:6810:ba29	2401:4900:8813:262:343b:5fa9:39f9:9f...	TLSv1.3	117	Application Data
13957	24.735464	2401:4900:8813:262:343b:5fa9:39f9:9f...	2606:4700:9760:f2c2:2376:6:6810:ba29	TLSv1.3	117	Application Data
13963	24.803521	192.168.1.3	23.3.70.186	TLSv1.3	778	Application Data
13971	24.965800	2606:4700:9760:f2c2:2376:6:6810:ba29	2401:4900:8813:262:343b:5fa9:39f9:9f...	TLSv1.3	1161	Application Data
13972	24.965806	2606:4700:9760:f2c2:2376:6:6810:ba29	2401:4900:8813:262:343b:5fa9:39f9:9f...	TLSv1.3	118	Application Data
13973	24.965807	2606:4700:9760:f2c2:2376:6:6810:ba29	2401:4900:8813:262:343b:5fa9:39f9:9f...	TLSv1.3	117	Application Data
13975	24.968478	2401:4900:8813:262:343b:5fa9:39f9:9f...	2606:4700:9760:f2c2:2376:6:6810:ba29	TLSv1.3	121	Application Data
13976	25.060015	23.3.70.186	192.168.1.3	TLSv1.3	225	Application Data

Frame 13957: 117 bytes on wire (936 bits), 117 bytes captured (936 bits) on interface Wi-Fi: en0
 Ethernet II, Src: Apple_d5:58:3a (7c:04:d0:d5:58:3a), Dst: ServercomPri_03:db:90 (b4:00:06:00:00:00)
 Internet Protocol Version 4, Src: 192.168.1.3, Dst: 23.3.70.186
 Transmission Control Protocol, Src Port: 49315, Dst Port: 443, Seq: 3731, Ack: 4019, Win: 65535, Len: 117
 Transport Layer Security
 TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol

4. QUIC (Quick UDP Internet Connections):

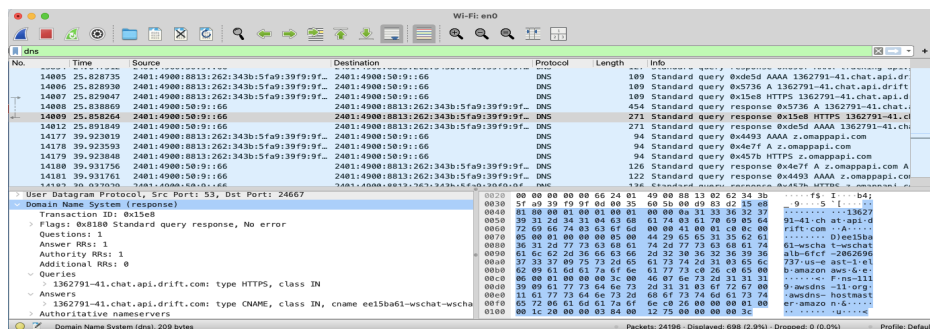
- QUIC is based on UDP: QUIC uses UDP as its transport protocol, but it adds features like connection multiplexing and flow control.
- Encrypted at the transport layer: QUIC mandates encryption for all connections, unlike TCP where it's optional.
- TLS handshake: QUIC uses a TLS handshake to establish a secure connection and encryption.
- HTTP/3 uses QUIC: HTTP/3 is built on top of QUIC.

- QUIC packet inspection refers to the process of analyzing and understanding the contents of packets transmitted over the QUIC (Quick UDP Internet Connections) protocol. QUIC is a transport-layer protocol designed to provide security, reliability, and efficiency for internet communications.
- It involves examining the packet headers, payloads, and other metadata to gain insights into the communication patterns, protocol errors, and potential security vulnerabilities.
- Sample QUIC packet captured Screenshot:



5. DNS (Domain Name System):

- DNS is a fundamental protocol on the internet, enabling users to access websites using human-friendly domain names rather than IP addresses.
- DNS typically uses UDP (User Datagram Protocol) for queries, but can also use TCP (Transmission Control Protocol) for certain types of queries.
- Wireshark's DNS dissector is fully functional and can help in various analyses, including security investigations and performance monitoring.
- Understanding DNS is crucial for network security, as malicious actors can exploit DNS for various attacks, such as domain hijacking or DNS poisoning.
- Sample DNS packet captured Screenshot:



6. UDP (User Datagram Protocol):

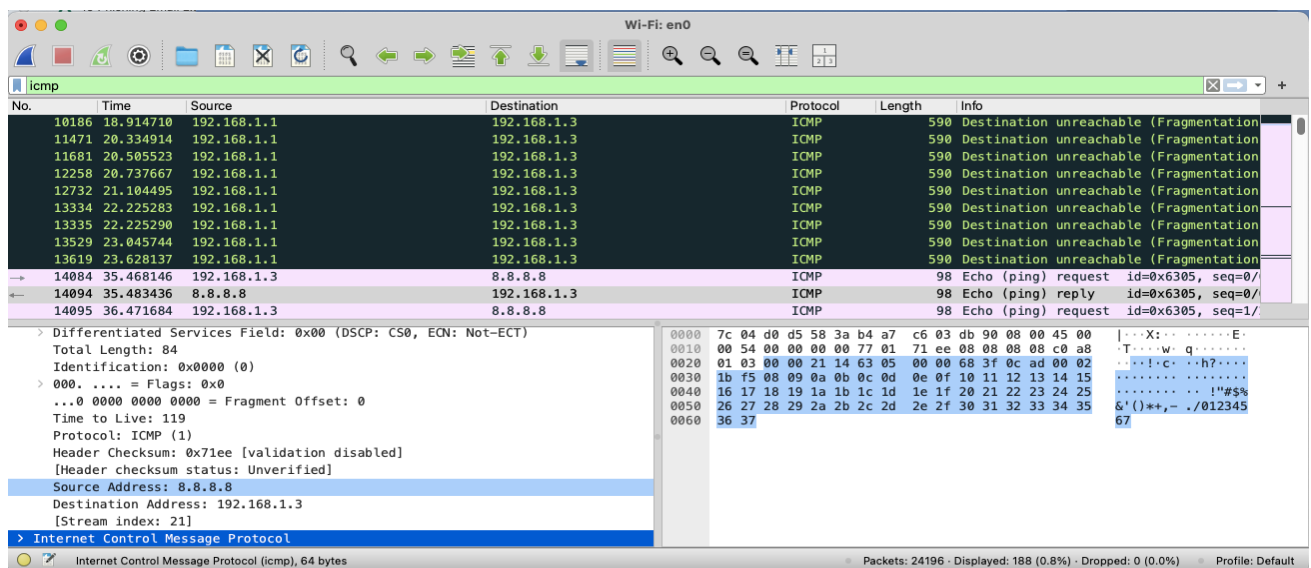
- UDP is a connectionless protocol used for applications like DNS and streaming where speed is prioritized over guaranteed delivery.
- Sample UDP packet captured Screenshot:

[illegible]

- Transmission Control Protocol (TCP) is a communications standard that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks.
- Sample TCP packet captured Screenshot:

8. ICMP(Internet Control Message Protocol):

- ICMP is a protocol used for error-reporting and diagnostic functions, including ping and traceroute.
- Understanding ICMP protocol in Wireshark is essential for network administrators and engineers to identify and resolve connectivity issues. This knowledge helps to optimize network performance and troubleshoot problems.
- Sample ICMP packet analysis:
 - **Type:** Echo Request (ping to 8.8.8.8)
 - **Reply:** Echo Reply received
 - **Source Address :** 8.8.8.8
 - **Destination Address :** 192.168.1.3
 - **Screenshot -**



4. Protocol : ICMP

Deliverable:

- **Packet Capture File (.pcap):** task5_traffic_analysis.pcap
- [You would normally attach/upload this file]

Summary:

The network traffic captured shows standard protocol usage:

- **HTTP** and **DNS** for website browsing
 - **TCP** underlying all connections
 - Optional **ICMP** from ping activity
- No suspicious or malicious activity was observed in the brief capture window. The exercise demonstrates baseline protocol behavior during normal network use.

