Ex:No:14

Date:

14.To capture, save, and analyze network traffic on TCP / UDP / IP / HTTP / ARP /DHCP /ICMP /DNS using Wireshark Tool

Aim

To capture, filter, and analyze live network traffic to understand the structure, function, and interaction of the following key protocols:
TCP, UDP, IP, HTTP, ARP, DHCP, ICMP, and DNS. The goal is to observe the packets at a low level to verify the theoretical operation of the OSI/TCP-IP model layers.

Theory

Wireshark is a **network protocol analyzer** (or packet sniffer) that captures and displays the raw data streams traveling over a network. It places the network interface card (NIC) into **promiscuous mode** (where possible) to capture all traffic visible to the host, then reconstructs the packets and presents them in a human-readable format based on the structure of the protocols.

Protocol	OSI Layer	Function Observed in Wireshark	Key Wireshark Filter
ARP (Address Resolution Protocol)	Data Link (2)	Maps an IP address to a physical MAC address on the local network. Look for Request (broadcast) and Reply (unicast).	arp
IP (Internet Protocol)	Network (3)	Provides logical addressing (IPv4/IPv6) and routing across networks. Forms the base of nearly all packets.	ip
ICMP (Internet Control	Network (3)	Used for network diagnostics (e.g., Ping) and error reporting.	icmp

Protocol	OSI Layer	Function Observed in Wireshark	Key Wireshark Filter
Message Protocol)			
TCP (Transmission Control Protocol)	Transport (4)	Connection-oriented, reliable transport. Observe the 3-way handshake (SYN, SYN-ACK, ACK) and session termination (FIN, ACK).	tcp
UDP (User Datagram Protocol)	Transport (4)	Connectionless, fast, but unreliable transport. Data is sent without prior connection establishment.	udp
DHCP (Dynamic Host Configuration Protocol)	Application (7)	Assigns IP addresses to hosts. Observe the DORA process (Discover, Offer, Request, Acknowledge).	bootp or dhcp
DNS (Domain Name System)	Application (7)	Resolves human-readable domain names to numerical IP addresses (typically uses UDP port 53).	dns
HTTP (Hypertext Transfer Protocol)	Application (7)	The protocol for web pages and data transfer. Look for unencrypted GET and POST requests.	http

Export to Sheets

Procedure

Part 1: Initial Capture and IP/TCP/UDP Analysis

Select Interface: Launch Wireshark. From the initial screen, select the primary network interface (e.g., Ethernet or Wi-Fi) that has active traffic.

Start Capture: Click the **Start** button (shark fin icon).

Generate Traffic: Open a command prompt/terminal and perform a basic network task, such as:

ping 127.0.0.1 (Local ICMP)

ping google.com (ICMP and DNS)

Open a browser and visit a non-HTTPS site like http://example.com (HTTP, TCP, DNS).

Stop and Save: Click the **Stop** button. Save the capture file as a .pcapng file (e.g., network_analysis.pcapng).

Part 2: Protocol-Specific Filtering and Observation

Use the **Display Filter** bar in Wireshark for targeted analysis, noting the observations in the Packet Details pane (middle section).

Protocol	Wireshark Filter	Action to Generate Traffic	Expected Observation
ICMP	icmp	ping 8.8.8.8 (or any public IP)	Pairs of packets: **Echo Request** (Type 8) followed by **Echo Reply** (Type 0).
DNS	dns	Browse to a new website or run nslookup example.com	DNS **Standard Query** (sent via UDP) followed by **Standard Query Response** containing the resolved IP address.
ARP	arp	Ping a local IP that hasn't been contacted recently (e.g., your router)	**ARP Request** (Who has IP X? Tell IP Y), which is a broadcast, followed by a **ARP Reply** (I am IP X, my MAC is Z).
DHCP	bootp or dhcp	Force your NIC to release and renew its IP address (e.g., ipconfig /renew)	DHCP **Discover** (D) → **Offer** (O) → **Request** (R) → **ACK** (A).

Protocol	Wireshark Filter	Action to Generate Traffic	Expected Observation
TCP	tcp.port == 80	Visit an HTTP site.	Observe the **three-way handshake**: SYN, SYN-ACK, ACK. The relative sequence numbers track the connection state.
HTTP	http	Visit http://example.com	Observe the **HTTP GET** request, containing the requested resource path, followed by the **HTTP OK** response packet. The content is visible in the Packet Bytes pane.

Export to Sheets

Observation

The key observations highlight the function of each protocol layer:

> **Layer 2 (Data Link) - ARP:** ARP packets were essential for local communication. The ARP Request used a **broadcast MAC address** (ff:ff:ff:ff:ff:ff) to discover the MAC for a known IP, confirming it is a local network protocol.
>
> **Layer 3 (Network) - IP & ICMP:** Every single routable packet contained an IP header, providing the source and destination logical addresses. ICMP was used exclusively for diagnostic messages (ping), carrying no application data but instead checking for reachability.
>
> **Layer 4 (Transport) - TCP & UDP:**
>
> > **TCP** (for HTTP traffic) demonstrated reliability by successfully executing the SYN-SYN-ACK handshake to establish a connection before data transfer. The packet headers contained sequence and acknowledgment numbers.

UDP (for DNS traffic) showed efficiency by sending the query immediately without a handshake, highlighting its connectionless nature.

Layer 7 (Application) - HTTP, DHCP, DNS: These protocols confirmed the application-layer functions:

The HTTP packets clearly showed the unencrypted text of the GET request.

DNS packets contained the successful translation of a domain name (e.g., google.com) into its corresponding IP address.

DHCP packets showed the client requesting network parameters and the server providing them, detailing the assigned IP address and subnet mask within the DHCP payload.