Task – 5

1 What is wireshark used for ?

* Wireshark is used to capture, inspect, and analyze network traffic (data packets) to troubleshoot network issues, investigate security incidents, and understand network protocols in real-time. It allows professionals and developers to deep-dive into packet data, identify performance problems, monitor network activity, and gain insights into various network protocols like TCP, UDP, and HTTP.

2 What is a packet ?

* A **packet** is a small unit of data transmitted over a network. When information (like a web page, video, or email) is sent from one computer to another, it is broken down into smaller chunks called **packets**. These packets travel independently across the network and are reassembled at the destination to form the original message.

3 How to filter packets in wireshark ?

* Filtering packets in **Wireshark** is a core feature that allows you to narrow down the network traffic to only what you're interested in. There are **two main types of filters** in Wireshark:

**1 Capture Filters**

2. **Display Filters**

4 What is the difference between tcp and udp ?

* The key difference between **TCP (Transmission Control Protocol)** and **UDP (User Datagram Protocol)** lies in **how they handle data transmission** over a network.

Here’s a clear breakdown:

1. **1. Connection Type**

* **TCP**: Connection-oriented  
  → Establishes a connection before sending data (like a phone call).
* **UDP**: Connectionless  
  → Sends data without establishing a connection (like sending a letter).

1. **✅ 2. Reliability**

* **TCP**: **Reliable**  
  → Guarantees delivery, checks for errors, and retransmits lost packets.
* **UDP**: **Unreliable**  
  → No guarantees for delivery, order, or duplication.

1. **3. Data Ordering**

* **TCP**: Maintains **order** of data packets.
* **UDP**: **No guarantee** of order — packets may arrive out of sequence.

1. **4. Speed**

* **TCP**: **Slower**, due to overhead from error checking, acknowledgments, etc.
* **UDP**: **Faster**, with minimal overhead — ideal for real-time apps.

1. **5. Use Cases**

| **TCP** | **UDP** |
| --- | --- |
| Web browsing (HTTP/HTTPS) | Live video/audio streaming |
| Email (SMTP, IMAP, POP3) | Online gaming |
| File transfers (FTP) | DNS lookups |
| Remote access (SSH) | VoIP (Skype, Zoom, etc.) |

1. **6. Error Checking**

* **TCP**: Performs error checking and correction.
* **UDP**: May perform error checking, but **no correction**.

5 What is a dns query packet ?

* A **DNS query packet** is a network message sent by a device (like your computer or phone) to a **DNS server** to resolve a **domain name** (like example.com) into an **IP address** (like 93.184.216.34).

1. **DNS Overview:**

The **Domain Name System (DNS)** works like the internet's phonebook. When you type a web address into your browser, your device sends a DNS query to ask, *"What is the IP address of this domain?"*

1. **Structure of a DNS Query Packet**

A DNS query packet typically uses the **UDP protocol** (port 53), though TCP can be used for large queries (like DNSSEC or zone transfers).

Here’s a breakdown of the **main parts** of a DNS query packet:

1. **1. Header (12 bytes)**

Contains control information, including:

* **Transaction ID (2 bytes)**: Random ID to match responses with queries.
* **Flags (2 bytes)**: Indicates if the packet is a query or response, etc.
* **QDCOUNT (2 bytes)**: Number of questions (usually 1 in a standard query).
* **ANCOUNT, NSCOUNT, ARCOUNT (2 bytes each)**: Counts for answers, authority, and additional sections (all 0 in a query).

1. **2. Question Section**

This is the actual question being asked:

* **QNAME**: The domain name being queried (e.g., www.google.com), written in a special DNS name format.
* **QTYPE**: Type of query (e.g., A for IPv4 address, AAAA for IPv6, MX for mail server).
* **QCLASS**: Usually IN (Internet class).

6 How can packet capture help in troubleshooting ?

* Packet capture is one of the most powerful techniques in troubleshooting network, system, or security issues because it records the actual data packets moving across a network. Here’s how it helps:

1. **1. Identifying Connectivity Issues**

* You can see whether packets are actually leaving the source and arriving at the destination.
* Helps confirm if the problem is due to **network routing, firewall blocks, or host issues**.

1. **2. Diagnosing Performance Problems**

* Packet capture shows **latency, retransmissions, or dropped packets**, which may point to congestion or faulty equipment.
* Detects **slow applications** caused by delayed responses or excessive retries.

1. **3. Troubleshooting Security Problems**

* Reveals suspicious traffic such as **malware communication, scanning, or unauthorized access attempts**.
* Helps detect **spoofing, man-in-the-middle attacks, or unusual protocols**.

1. **4. Protocol & Application Debugging**

* Lets you analyze the behavior of protocols (e.g., HTTP, DNS, TCP handshakes).
* Useful for checking **whether an application is sending malformed requests or misconfigured messages**.

1. **5. Verifying Configurations**

* Confirms if **NAT, firewall rules, or VPN tunnels** are working as intended.
* Shows whether the correct ports and IPs are being used.

1. **⚡ Tools Used**

* **Wireshark** (most popular for deep analysis)
* **tcpdump / tshark** (command-line captures)
* **Network taps / SPAN ports** (for large-scale monitoring)

7 What is a protocol ?

* A **protocol** is basically a set of rules or standards that define how data is transmitted and communicated between devices, systems, or applications.

8 Can wireshark decryt encrypted traffic ?

* sometimes — yes, but only when you also have the right keys/secrets or captured handshakes. Wireshark **cannot magically break modern cryptography** (ECDHE/TLS 1.3/QUIC) on its own.

Here’s a practical, quick guide to when and how Wireshark can decrypt encrypted traffic, and the limits.

1. **When Wireshark can decrypt**
2. **You have the session secrets (recommended method).**
   * Browsers (Chrome/Firefox) can be made to write a SSLKEYLOGFILE (pre-master/master secrets). Point Wireshark to that file and it will decrypt TLS (including TLS 1.2 and TLS 1.3 traffic that used those secrets).
3. **You captured the WPA2/WPA-PSK 4-way handshake + know the Wi-Fi passphrase.**
   * Wireshark can derive the PTK and decrypt 802.11 (WPA/WPA2) frames if the handshake is in the capture.
4. **You have IPsec/ESP keys or other symmetric keys.**
   * If you can supply the SA keys, Wireshark can decrypt IPsec traffic.
5. **Older RSA key exchange (rare):**
   * If a server used RSA key exchange (not ECDHE) and you have the server private key, Wireshark can decrypt — but modern sites use ephemeral key exchanges so this is rarely useful.
6. **When Wireshark cannot decrypt**

* **Perfect Forward Secrecy (PFS)** algorithms (ECDHE, DHE) where you don’t have session secrets. Possessing the server private key won’t help for these.
* **QUIC** and **TLS 1.3**: they can be decrypted *only* if you supply the session secrets (e.g. SSLKEYLOGFILE). You cannot decrypt them with just the server private key when ephemeral key exchange is used.
* **SSH** and many proprietary encrypted tunnels: generally not decryptable unless you control an endpoint and can export keys.