**Lab 1**

**What is wireshark?**

Wireshark is a powerful, open-source network protocol analyzer used for capturing and analyzing network traffic in real time. It helps in troubleshooting network issues, monitoring security threats, and understanding various network protocols like TCP, UDP, and HTTP. With its intuitive graphical interface, users can inspect packet details, apply filters, and diagnose performance issues efficiently. Wireshark is widely used by network administrators, cybersecurity professionals, and developers to gain deep insights into network communication.

An **IP (Internet Protocol) address** is a unique numerical identifier assigned to a device on a network. It allows devices to communicate over the internet or a local network.

### **Types of IP Addresses**

IP addresses are classified into different types based on their purpose and structure.

#### **1. IPv4 vs. IPv6**

| **Feature** | **IPv4** | **IPv6** |
| --- | --- | --- |
| Address Length | 32-bit | 128-bit |
| Format | Decimal (e.g., 192.168.1.1) | Hexadecimal (e.g., 2001:0db8:85a3::8a2e:0370:7334) |
| Total Addresses | ~4.3 billion | ~340 undecillion (virtually unlimited) |
| Usage | Most common | Slowly replacing IPv4 |

#### **2. Public vs. Private IPs**

* **Public IP Address** – Assigned by ISPs (Internet Service Providers) and is unique across the internet.
* **Private IP Address** – Used within local networks (e.g., 192.168.1.1, 10.0.0.1), not routable on the internet.

#### **3. Static vs. Dynamic IPs**

* **Static IP** – Does not change, useful for hosting servers.
* **Dynamic IP** – Assigned temporarily by DHCP (Dynamic Host Configuration Protocol) and changes periodically.

### **How IP Addresses Work**

1. When a device connects to a network, it is assigned an IP address.
2. This IP address identifies the device so it can send and receive data.
3. Domain Name System (DNS) translates human-readable domain names (e.g., google.com) into IP addresses.
4. Routers use IP addresses to direct packets of data across networks.

### **Subnetting**

Subnetting divides an IP network into smaller subnetworks to optimize performance and security.  
Example:

* **Class A:** 1.0.0.0 – 126.255.255.255 (Large networks)
* **Class B:** 128.0.0.0 – 191.255.255.255 (Medium networks)
* **Class C:** 192.0.0.0 – 223.255.255.255 (Small networks)

### **Key Protocols Related to IP Addresses**

* **TCP/IP (Transmission Control Protocol/Internet Protocol)** – Core protocol for internet communication.
* **NAT (Network Address Translation)** – Allows multiple devices to share a single public IP address.
* **DHCP (Dynamic Host Configuration Protocol)** – Automatically assigns IP addresses to devices.

### **1. Ethernet Frame (Layer 2 - Data Link Layer)**

* **Definition:** The fundamental unit of communication in Ethernet networks.
* **Key Fields:**
  + **Source MAC Address**: The sender's MAC address.
  + **Destination MAC Address**: The receiver's MAC address.
  + **EtherType**: Specifies the protocol used (e.g., IPv4, IPv6).
  + **Payload**: Contains higher-layer data (IP, TCP, UDP, etc.).
  + **Frame Check Sequence (FCS)**: Used for error detection.

### **2. IP Packet (Layer 3 - Network Layer)**

* **Definition:** Encapsulates data for routing across networks.
* **Key Fields:**
  + **Source IP Address**: The sender's IP address.
  + **Destination IP Address**: The receiver's IP address.
  + **Protocol**: Identifies transport layer protocol (TCP, UDP, ICMP).
  + **Time-To-Live (TTL)**: Prevents infinite looping of packets.
  + **Header Checksum**: Ensures integrity of the packet.

### **3. TCP Packet (Layer 4 - Transport Layer)**

* **Definition:** A connection-oriented protocol ensuring reliable communication.
* **Key Fields:**
  + **Source Port & Destination Port**: Identifies sending and receiving applications.
  + **Sequence Number**: Tracks data order.
  + **Acknowledgment Number**: Confirms receipt of data.
  + **Flags (SYN, ACK, FIN, RST, PSH, URG)**: Controls communication process.
  + **Checksum**: Validates packet integrity.

### **4. UDP Packet (Layer 4 - Transport Layer)**

* **Definition:** A connectionless, lightweight protocol for fast communication.
* **Key Fields:**
  + **Source Port & Destination Port**: Identifies sender and receiver.
  + **Length**: Indicates total packet size.
  + **Checksum**: Ensures packet integrity.
* **Use Cases:** Streaming, VoIP, DNS, gaming.

### **5. ICMP Packet (Layer 3 - Network Layer)**

* **Definition:** Used for diagnostics and error reporting.
* **Key Fields:**
  + **Type & Code**: Specifies the message type (e.g., Echo Request, Echo Reply, Destination Unreachable).
  + **Checksum**: Ensures correctness.
  + **Data**: Optional payload for additional information.
* **Use Cases:** Ping (connectivity testing), Traceroute.

### **6. ARP Packet (Layer 2 - Data Link Layer)**

* **Definition:** Resolves IP addresses to MAC addresses.
* **Key Fields:**
  + **Sender IP & MAC Address**: Identifies the sender.
  + **Target IP & MAC Address**: Identifies the intended receiver.
  + **Operation (Request/Reply)**: Specifies the ARP message type.
* **Use Case:** Local network communication.

### **7. DNS Packet (Layer 7 - Application Layer)**

* **Definition:** Resolves domain names to IP addresses.
* **Key Fields:**
  + **Query Type (A, AAAA, CNAME, MX, etc.)**: Specifies the type of record requested.
  + **Transaction ID**: Uniquely identifies the query.
  + **Flags**: Specifies request/response details.
  + **Answers**: Contains resolved IP addresses.

### **8. HTTP/HTTPS Packet (Layer 7 - Application Layer)**

* **Definition:** Protocol for web communication.
* **Key Fields:**
  + **Method (GET, POST, PUT, DELETE)**: Defines the request type.
  + **Headers**: Includes host, user-agent, cookies, etc.
  + **Status Code**: Response code (200 OK, 404 Not Found, etc.).
  + **Body**: Contains webpage data or API response.

**Lab Title: Network Traffic Analysis Using Wireshark**

**Objective:**

* Learn how to capture and analyze network traffic using Wireshark.
* Understand the basics of different network protocols (e.g., TCP, UDP, HTTP).
* Identify and interpret key information from captured packets.

**Prerequisites:**

* Basic understanding of networking concepts (IP addresses, protocols, packet structure).
* Wireshark installed on the lab computers.

**Materials Needed:**

* Computers with Wireshark installed.
* A networked environment (could be a local network or internet access).
* Sample capture files (optional).

**Lab Tasks:**

**Task 1: Introduction to Wireshark**

1. **Installing Wireshark** (if not already installed):
2. **Visit the Official Website:**
   * Open a web browser and go to the [Wireshark official website](https://www.wireshark.org/).
3. **Download the Installer:**
   * On the homepage, click on the "Download" button.
   * Choose the appropriate installer for your operating system (e.g., Windows, macOS, Linux).
4. **Run the Installer:**
   * Once the download is complete, locate the installer file (e.g., Wireshark-win64.exe for Windows).
   * Double-click the installer to run it.
5. **Follow the Installation Wizard:**
   * The Wireshark Setup Wizard will open. Click "Next" to proceed.
   * Accept the license agreement.
   * Choose the installation components (you can keep the default settings).
   * **Note for Windows Users:** The installer will prompt you to install **WinPcap or Npcap**, which are necessary for packet capturing. Ensure you select the option to install Npcap.
6. **Complete the Installation:**
   * Click "Install" to begin the installation process.
   * Once the installation is complete, click "Finish" to exit the Setup Wizard.
7. **Launch Wireshark:**
   * After installation, you can start Wireshark from the Start menu .
   * On first launch, you may need to grant **network permissions** for packet capturing.
8. **Getting Familiar with the Interface**:
   * Open Wireshark and explore the interface.
   * Identify key sections: **capture interfaces**, **packet** list pane, **packet details pane**, and **packet bytes pane**.
9. **Selecting a Network Interface**:
   * Identify the correct network interface (e.g., Ethernet, Wi-Fi) that will be used for capturing traffic.
   * Explain how different interfaces can show different traffic based on the network setup.

**Task 2: Capturing Network Traffic**

1. **Starting a Capture Session**:
   * Select the appropriate network interface and start capturing traffic.
   * Perform typical network activities (e.g., browsing websites, pinging a server) to generate traffic.
   * Stop the capture after a few minutes.
2. **Saving and Opening Capture Files**:
   * Save the captured traffic to a .pcap file.
   * Explain how to open saved capture files for future analysis.

**Task 3: Analyzing Captured Packets**

1. **Basic Packet Analysis**:
   * Explore the packet list pane to see different captured packets.
   * Highlight packets related to common protocols like TCP, UDP, and HTTP.
   * Explain key fields in the packet details pane (e.g., source and destination IP addresses, ports, sequence numbers).
2. **Using Display Filters**:
   * Introduce the concept of display filters to focus on specific traffic types.
   * Practice using filters like ip.src == 192.168.1.1/10.x.x.x, tcp, udp, and http. e.g
   * Show how to combine filters with logical operators (e.g., tcp && ip.addr == 192.168.1.1/10.x.x.x).
3. **Reconstructing HTTP Streams**:
   * Use Wireshark’s "Follow TCP Stream" feature to reconstruct and view HTTP sessions. Select **Analyze**> **Follow** > **TCP Stream**
   * Analyze the HTTP requests and responses to understand how data is exchanged between a client and server.
4. **Identifying Protocols**:
   * Discuss common protocols observed in the capture (e.g., ARP, DNS, DHCP).
   * Explain how Wireshark’s protocol dissectors work to interpret and display protocol data.

**Task 4: Advanced Analysis**

1. **Analyzing Latency and Performance**:
   * Measure round-trip time (RTT) and analyze the TCP handshake process.  
     **Statistics>TCP stream Graphs> Round Trip Time**
   * Discuss how to identify potential performance issues using Wireshark (e.g., high latency, packet loss).
2. **Detecting Network Issues**:
   * Identify common network issues such as retransmissions, duplicate ACKs, and packet fragmentation.
   * Discuss how these issues can impact network performance.

**Assessment:**

* Submit a report that includes:
  + Screenshots of specific packets with explanations.
  + A summary of key findings from their analysis.
  + Answers to any specific questions posed during the lab (e.g., identifying certain types of traffic, explaining packet contents).

**Conclusion:**

By the end of this lab, you will have a solid understanding of how to use Wireshark for capturing and analyzing network traffic. They will be able to interpret packet data and identify different types of network protocols and potential issues.

Note. Some filters

**Filter Traffic:**

* Use display filters to isolate TCP traffic.
* Example filter for TCP handshake:

Example 1: tcp.flags.syn == 1 && tcp.flags.ack == 0

Note:

The Stream Index column **displays a unique number for each stream**, such as 1 for the first stream, 2 for the second stream, et cetera. A stream is a related collection of TCP packets, typically beginning with the 3-way handshake, then the data transfer, and ending with the session tear download

Download

<https://learn.microsoft.com/en-us/sysinternals/downloads/psping>

psping <hostname or IP>:<port>

psping google.com:443

Here are some **commonly used network packets** and their **introduction**:

### **1. Using AND (**and**)** – Both conditions must be **true**

✅ **Example:**

ip.src == 192.168.1.1 and tcp.port == 80

🔹 **Explanation:** Captures packets **only** where:

* The **source IP is 192.168.1.1**
* The **TCP port is 80 (HTTP traffic)**

### **2. Using OR (**or**)** – At least one condition must be **true**

✅ **Example:**

tcp.port == 443 or udp.port == 53

🔹 **Explanation:** Captures packets where:

* The **TCP port is 443 (HTTPS traffic)**
* OR the **UDP port is 53 (DNS traffic)**

### **3. Using NOT (**not**)** – Excludes certain packets

✅ **Example:**

not arp and ip.addr == 192.168.1.100

🔹 **Explanation:** Captures packets where:

* The **IP address is 192.168.1.100**
* **Excludes ARP packets**

### **4. Using Parentheses for Complex Filters**

✅ **Example:**

(ip.src == 10.0.0.1 or ip.dst == 10.0.0.1) and tcp.port == 22

🔹 **Explanation:** Captures packets where:

* **Either the source or destination IP is 10.0.0.1**
* AND the **TCP port is 22 (SSH traffic)**

### **5. Filtering Specific TCP Flags (Example: SYN Packets Only)**

✅ **Example:**

tcp.flags.syn == 1 and tcp.flags.ack == 0

🔹 **Explanation:** Captures only **TCP SYN packets** (used in connection establishment).

### **Troubleshooting Errors**

✔️ **Use and, or, not instead of &&, ||, !**  
✔️ **Check correct field names using Wireshark's "Expression" button**