Step 1: Installed Wireshark

- **Wireshark was successfully installed on my system.**
- **Npcap was installed alongside Wireshark to allow network packet capture.**
- **The application launched without errors.**

Step 2: Started capturing on active network interface

- **I selected the active network interface (Wi-Fi) from the list in Wireshark.**
- **Clicked the Start Capturing Packets button.**
- **Real-time packet capture began, showing packets scrolling in the main pane.**

Step 3: Generated network traffic

- **I opened a web browser and visited a website (e.g., http://neverssl.com) to generate HTTP traffic.**
- **Traffic packets were visible in Wireshark during these activities.**

Step 4: Stopped the capture

- **After approximately one minute, I clicked the Stop Capturing Packets button.**
- **The capture stopped, and a complete list of packets appeared with columns like No., Time, Source, Destination, Protocol, Length, Info.**

Step 5: Filtered packets by protocol

- **Applied protocol filters in Wireshark to isolate specific traffic:**
  - **HTTP: http**
  - **DNS: dns**
  - **TCP: tcp**
- **Filtered packets allowed easier analysis of protocol-specific activity.**

Step 6: Identified at least 3 different protocols

- **From the captured packets, I identified the following protocols:**
    1. **DNS – Used for domain name resolution.**
    2. **TCP – Used for establishing reliable connections.**
    3. **HTTP – Web page requests and responses.**
- **I examined individual packet details to confirm the protocols and their functions.**

Step 7: Exported the capture as a .pcap file

- **Went to File → Export → Save As….**
- **Selected file type Wireshark/tcpdump (*.pcap).**
- **Saved the capture to a local folder for further analysis or submission.**

Step 8: Summarized findings and packet details

- **Total packets captured: 992.**
- **Protocol breakdown:**
    - **DNS: [58] packets, showing domain queries and responses.**
    - **TCP: [626] packets, showing connection establishment and teardown.**
    - **HTTP: [5] packets, showing web page requests/responses.**

Detailed Network Traffic Analysis Report

Source File: Packets Traffic.pcap

Target Website: http://neverssl.com

Analysis Focus: Protocol identification, filtering, and summarization of packet details.

1. Identified Protocols

The network capture reveals traffic utilizing multiple protocols across the TCP/IP stack. The primary protocols necessary for accessing http://neverssl.com, along with common background traffic, have been identified:

Protocol	Layer	Role in Web Request
TCP (Transmission Control Protocol)	Transport	Provides **reliable, ordered delivery** of data for the HTTP application session.
HTTP (Hypertext Transfer Protocol)	Application	The protocol used to send the **web request** (GET /) and receive the response on standard port **80**.
DNS (Domain Name System)	Application	Essential for **translating the domain name** (neverssl.com) into a numerical IP address.
SSDP (Simple Service Discovery Protocol)	Application (over UDP)	**Background noise** protocol used for discovering UPnP/DLNA devices on the local network.

2. Summary of Findings and Packet Details

The following sections detail the observed packet flows and key fields for the primary protocols involved in accessing neverssl.com.

Protocol Filter 1: TCP (Transmission Control Protocol)

TCP is the most fundamental protocol in this capture, as it provides the connection-oriented service for HTTP.

Flow Step	Sample Packet Flags/Details	Key Packet Function
Connection Initiation (Client → Server)	**Flags: SYN** (Synchronize)	The client sends a packet with a random **Initial Sequence Number (ISN)** to begin the **three-way handshake**.
Connection Establishment (Server → Client)	**Flags: SYN, ACK** (Acknowledge)	The server responds, acknowledging the client's ISN and sending its own ISN.
Data Transfer	**Flags: ACK** and **PSH, ACK** (Push and Acknowledgment)	Once established, all subsequent data segments, including the HTTP request and response, are carried with these flags.
Source/Dest Ports	High Port (Client) → **80** (Server)	All HTTP traffic is encapsulated within these TCP port pairings.

Protocol Filter 2: HTTP (Hypertext Transfer Protocol)

This protocol handles the actual content transfer on port 80. The simple, non-secure nature of neverssl.com means this traffic is sent in **plaintext**.

Flow Step	Sample Packet Content/Details	Key Packet Function
Web Request (Client → Server)	**Method**: GET / HTTP/1.1 **Host Header**: Host: neverssl.com	The client is requesting the main page (/) of the website.
Web Response (Server → Client)	**Status Code**: 200 OK or possibly a 301/302 Redirect	The server delivers the requested content (the HTML for the page) or

Flow Step	Sample Packet Content/Details	Key Packet Function
		instructs the client to go to a different address.
Background Traffic	**Method**: NOTIFY * HTTP/1.1	**SSDP** utilizes HTTP methods over UDP multicast to announce device services (e.g., Server: Android AccessTwine/1.0-DMR).

Protocol Filter 3: DNS (Domain Name System)

DNS is typically the very first step in the entire transaction, occurring before the TCP handshake.

Flow Step	Sample Packet Details	Key Packet Function
Query (Client → DNS Server)	**Source Port**: Random High Port. **Destination Port**: **53 (UDP)**.	The client asks the DNS server to provide the **A record** (IPv4 address) for neverssl.com.
Response (DNS Server → Client)	**Source Port**: **53 (UDP)**. **Destination Port**: Random High Port.	The server returns the IP address, allowing the client to initiate the TCP connection to the correct destination IP.