# ZTS IMPLEMENTATION OF WIRESHARK TOOL

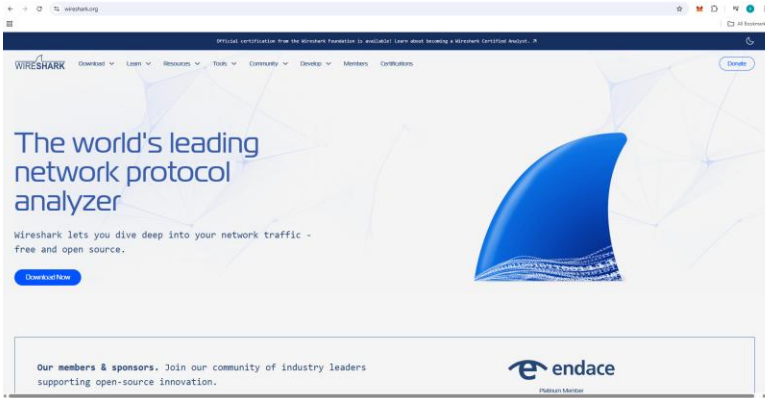
## 1. Introduction

Network security represents one of the most critical aspects of modern information technology infrastructure. Organizations today face an unprecedented volume of cyber threats ranging from data breaches and malware infections to ransomware attacks and Advanced Persistent Threats (APTs). To effectively defend against these threats, security professionals require robust tools and methodologies for monitoring, analyzing, and responding to suspicious network activity.

Wireshark is a powerful, free, and open-source network protocol analyzer that serves as a cornerstone tool in the cybersecurity toolkit. It enables security professionals, network engineers, and IT administrators to capture network traffic in real-time and examine it at the packet level—the most granular view of network communications. By analyzing individual packets, Wireshark reveals the detailed structure of network protocols, allowing investigators to identify anomalies, suspicious patterns, and potential security threats that might otherwise go unnoticed.

The importance of Wireshark in cybersecurity cannot be overstated. Whether conducting forensic investigations following a security incident, troubleshooting connectivity issues, validating security controls, or detecting real-time threats, Wireshark provides the visibility needed to understand exactly what is happening on the network. Its ability to decode hundreds of network protocols and display packet-level details makes it an indispensable resource for anyone responsible for network or information security.

This project aims to explore Wireshark comprehensively—from installation and basic configuration through advanced packet filtering and analysis. By working through practical scenarios and real-world use cases, this project demonstrates how Wireshark can be leveraged to detect security threats, analyze network behavior, and support informed decision-making in cybersecurity operations.



Official certification from the Wireshark Foundation is available! Learn about becoming a Wireshark Certified Analyst.

The world's leading network protocol analyzer

Wireshark lets you dive deep into your network traffic - free and open source.

Our members & sponsors. Join our community of industry leaders supporting open-source innovation.

Figure 1: Wireshark Official Logo and Homepage

## 2. Overview of Wireshark

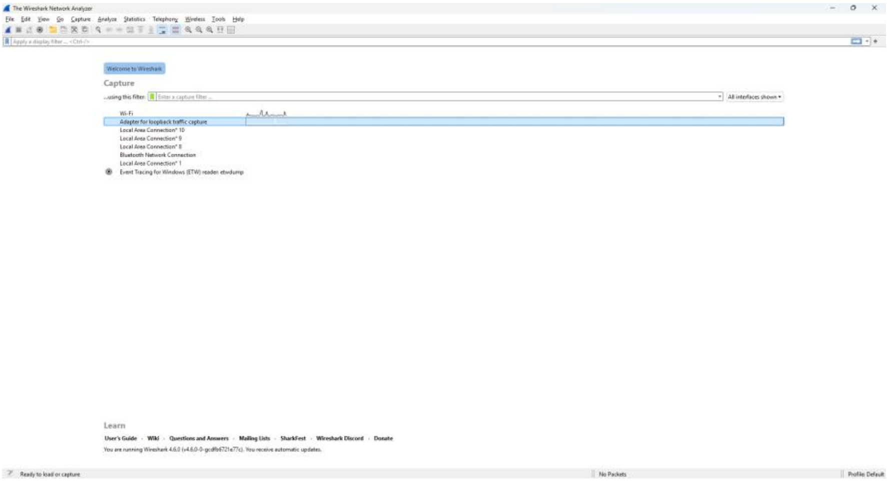
### 2.1 What is Wireshark?

Wireshark is the world's most widely-used network protocol analyzer. It is free and open-source software available under the GNU General Public License (GPL), making it accessible to organizations of all sizes and individual security researchers. Wireshark was originally developed as "Ethereal" in 1998 and was renamed Wireshark in 2006 following trademark issues.

At its core, Wireshark works by capturing network packets transmitted across a network interface and displaying them in a format that human analysts can understand and interpret. Unlike many network analysis tools that only provide aggregate traffic statistics, Wireshark allows you to drill down into individual packets and examine their complete structure, including headers, payloads, and protocol-specific information.

### 2.2 Core Capabilities of Wireshark

1. Live Packet Capture: Wireshark can capture packets in real-time from any active network interface on your computer. This includes Ethernet adapters, Wi-Fi interfaces, VPN connections, and loopback interfaces. Users can select which interface to monitor and customize capture filters to focus on specific types of traffic (e.g., only TCP traffic, only traffic to/from a specific IP address).


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate

Wi-Fi
Adapter for loopback traffic capture
Local Area Connection* 10
Local Area Connection* 9
Local Area Connection* 8
Bluetooth Network Connection
Local Area Connection* 1
Event Tracing for Windows (ETW) reader: etwdump

You are running Wireshark 4.6.0 (v4.6.0-0-gcdfb6721e77c). You receive automatic updates.

Ready to load or capture | No Packets | Profile: Default

Figure 2a: Network Interface Selection Screen

2. Protocol Analysis and Decoding: Wireshark supports dissection of over 1000 network protocols. When you capture packets, Wireshark automatically recognizes and decodes each protocol layer present in the packet. For example, when analyzing a web request, Wireshark will decode:

- Physical/Data Link layer (Ethernet)
- Network layer (IP/IPv6)
- Transport layer (TCP/UDP)

- Application layer (HTTP/HTTPS, DNS, etc.)

This hierarchical display allows analysts to examine how data is encapsulated and transmitted across different protocol layers.

3.Advanced Filtering Capabilities: One of Wireshark's most powerful features is its ability to filter packets based on virtually any criteria. There are two types of filters:

- Capture Filters: Applied during packet capture to limit what is recorded (syntax similar to tcpdump)
- Display Filters: Applied after capture to show/hide specific packets (more intuitive syntax)

Examples of display filters:

- http – Show only HTTP traffic
- tcp.port == 443 – Show only traffic on port 443 (typically HTTPS)
- ip.addr == 192.168.1.100 – Show only traffic to/from a specific IP
- tcp.flags.syn == 1 – Show only packets with SYN flag set (connection initiations)
- dns.qry.name contains "google.com" – Show DNS queries for specific domains

4. Statistical Analysis Tools: Wireshark includes several built-in analysis tools that provide high-level summaries of captured traffic:

- Protocol Hierarchy: Shows which protocols are present in the capture and how much traffic each represents
- Conversations: Lists all communication sessions between hosts/ports
- Endpoints: Identifies all unique IP addresses or MAC addresses communicating on the network
- Traffic Statistics: Generates graphs showing traffic volume over time

5. Deep Packet Inspection: Users can expand any packet in the packet details pane to see the exact structure of each protocol layer. This reveals hex values, checksums, sequence numbers, and other low-level details essential for security analysis and forensics.

6. Offline Analysis and Export: Wireshark can save captured traffic to files (typically in .pcap or .pcapng format) for later analysis. This is crucial for forensic investigations where traffic is analyzed days or weeks after it was captured. Captured data can also be exported in various formats (CSV, JSON, XML, etc.) for integration with other tools or for creating reports.

**2.3 Wireshark's Role in Cybersecurity**

Wireshark serves multiple critical functions in cybersecurity operations:

Threat Detection and Analysis : Security analysts use Wireshark to identify malicious network traffic patterns. This includes:

- Detecting port scans from external attackers or compromised internal systems
- Identifying Command & Control (C2) communications from malware
- Spotting data exfiltration attempts where sensitive data leaves the network
- Finding brute force login attempts captured in plaintext protocols
- Analyzing botnet communications

Forensic Investigation: When a security incident occurs, Wireshark is instrumental in reconstructing what happened:

- Analyzing capture files from the time of the incident to determine attack vectors
- Identifying the source and scope of breaches
- Reconstructing malicious file downloads or lateral movement within the network
- Providing evidence for incident response reports and legal proceedings

Vulnerability and Compliance Verification: Wireshark helps organizations verify their security posture:

- Confirming that encryption is properly implemented (by observing TLS/SSL handshakes and protocol versions)
- Detecting unencrypted transmission of sensitive data (passwords, API keys, credit cards)
- Verifying that deprecated or insecure protocols are not in use
- Checking for compliance with data protection policies

Network Troubleshooting: Beyond security, Wireshark aids in diagnosing connectivity and performance issues:

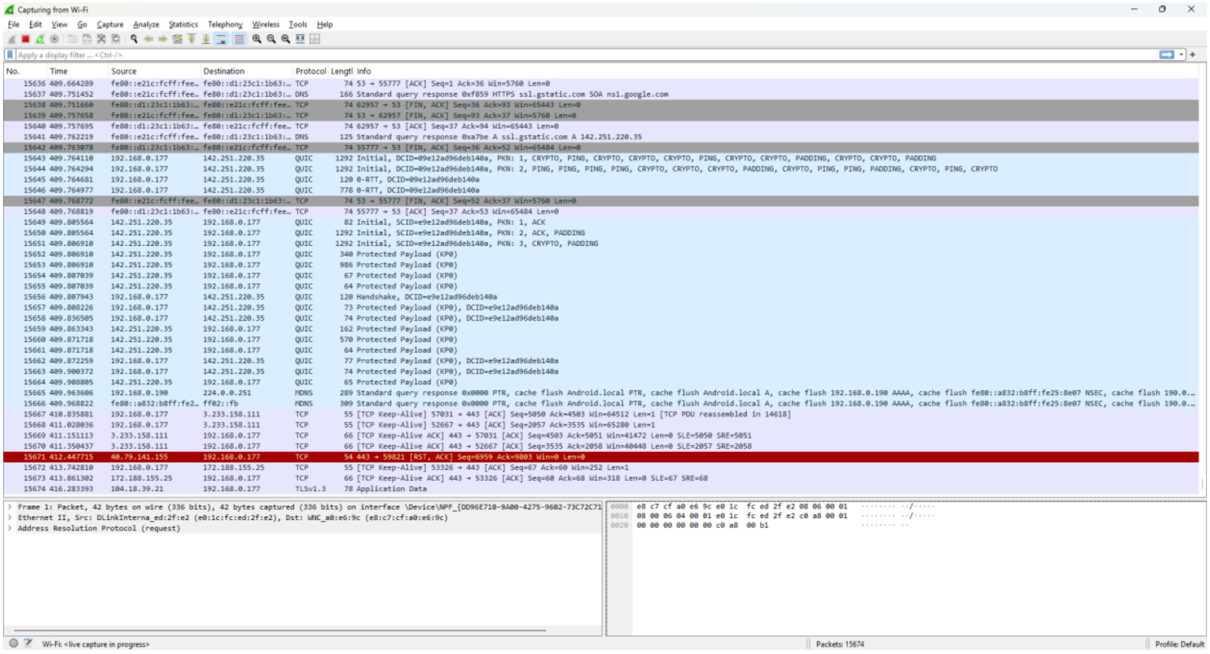
- Identifying connectivity failures or timeouts
- Detecting misconfigured DNS or DHCP services
- Finding latency or packet loss issues
- Analyzing application-level protocol errors

### 2.4 Wireshark Architecture and Components

Graphical User Interface(GUI): The primary way most users interact with Wireshark. The GUI is organized into several panes:

- Packet List Pane: Displays summary information for each captured packet
- Packet Details Pane: Shows the hierarchical protocol structure of the selected packet
- Packet Bytes Pane: Displays the raw hexadecimal data and ASCII representation

- Display Filter Bar: Where users enter filter expressions



Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time       Source              Destination         Protocol Lengtl Info
15636 409.664289 fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… TCP       74 53 → 55777 [ACK] Seq=1 Ack=36 Win=5760 Len=0
15637 409.751452 fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS      166 Standard query response 0xf859 HTTPS ssl.gstatic.com SOA ns1.google.com
15638 409.751660 fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… TCP       74 62957 → 53 [FIN, ACK] Seq=36 Ack=93 Win=65443 Len=0
15639 409.757658 fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… TCP       74 53 → 62957 [FIN, ACK] Seq=93 Ack=37 Win=5760 Len=0
15640 409.757695 fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… TCP       74 62957 → 53 [ACK] Seq=37 Ack=94 Win=65443 Len=0
15641 409.762219 fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS      125 Standard query response 0xa7be A ssl.gstatic.com A 142.251.220.35
15642 409.763078 fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… TCP       74 55777 → 53 [FIN, ACK] Seq=36 Ack=52 Win=65484 Len=0
15643 409.764110 192.168.0.177        142.251.220.35       QUIC    1292 Initial, DCID=09e12ad96deb140a, PKN: 1, CRYPTO, PING, CRYPTO, CRYPTO, CRYPTO, PING, CRYPTO, CRYPTO, PADDING, CRYPTO, CRYPTO, PADDING
15644 409.764294 192.168.0.177        142.251.220.35       QUIC    1292 Initial, DCID=09e12ad96deb140a, PKN: 2, PING, PING, PING, PING, CRYPTO, CRYPTO, CRYPTO, PADDING, CRYPTO, PING, PING, PADDING, CRYPTO, PING, CRYPTO
15645 409.764681 192.168.0.177        142.251.220.35       QUIC     120 0-RTT, DCID=09e12ad96deb140a
15646 409.764977 192.168.0.177        142.251.220.35       QUIC     778 0-RTT, DCID=09e12ad96deb140a
15647 409.768772 fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… TCP       74 53 → 55777 [FIN, ACK] Seq=52 Ack=37 Win=5760 Len=0
15648 409.768819 fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… TCP       74 55777 → 53 [ACK] Seq=37 Ack=53 Win=65484 Len=0
15649 409.805564 142.251.220.35       192.168.0.177        QUIC      82 Initial, SCID=e9e12ad96deb140a, PKN: 1, ACK
15650 409.805564 142.251.220.35       192.168.0.177        QUIC    1292 Initial, SCID=e9e12ad96deb140a, PKN: 2, ACK, PADDING
15651 409.806910 142.251.220.35       192.168.0.177        QUIC    1292 Initial, SCID=e9e12ad96deb140a, PKN: 3, CRYPTO, PADDING
15652 409.806910 142.251.220.35       192.168.0.177        QUIC     340 Protected Payload (KP0)
15653 409.806910 142.251.220.35       192.168.0.177        QUIC     986 Protected Payload (KP0)
15654 409.807039 142.251.220.35       192.168.0.177        QUIC      67 Protected Payload (KP0)
15655 409.807039 142.251.220.35       192.168.0.177        QUIC      64 Protected Payload (KP0)
15656 409.807943 192.168.0.177        142.251.220.35       QUIC     120 Handshake, DCID=e9e12ad96deb140a
15657 409.808226 192.168.0.177        142.251.220.35       QUIC      73 Protected Payload (KP0), DCID=e9e12ad96deb140a
15658 409.836505 192.168.0.177        142.251.220.35       QUIC      74 Protected Payload (KP0), DCID=e9e12ad96deb140a
15659 409.863343 142.251.220.35       192.168.0.177        QUIC     162 Protected Payload (KP0)
15660 409.871718 142.251.220.35       192.168.0.177        QUIC     570 Protected Payload (KP0)
15661 409.871718 142.251.220.35       192.168.0.177        QUIC      64 Protected Payload (KP0)
15662 409.872259 192.168.0.177        142.251.220.35       QUIC      77 Protected Payload (KP0), DCID=e9e12ad96deb140a
15663 409.900372 192.168.0.177        142.251.220.35       QUIC      74 Protected Payload (KP0), DCID=e9e12ad96deb140a
15664 409.908805 142.251.220.35       192.168.0.177        QUIC      65 Protected Payload (KP0)
15665 409.963606 192.168.0.190        224.0.0.251          MDNS     289 Standard query response 0x0000 PTR, cache flush Android.local PTR, cache flush Android.local A, cache flush 192.168.0.190 AAAA, cache flush fe80::a832:b8ff:fe25:8e07 NSEC, cache flush 190.0.…
15666 409.968822 fe80::a832:b8ff:fe2… ff02::fb             MDNS     309 Standard query response 0x0000 PTR, cache flush Android.local PTR, cache flush Android.local A, cache flush 192.168.0.190 AAAA, cache flush fe80::a832:b8ff:fe25:8e07 NSEC, cache flush 190.0.…
15667 410.835881 192.168.0.177        3.233.158.111        TCP       55 [TCP Keep-Alive] 57031 → 443 [ACK] Seq=5050 Ack=4503 Win=64512 Len=1 [TCP PDU reassembled in 14618]
15668 411.028036 192.168.0.177        3.233.158.111        TCP       55 [TCP Keep-Alive] 52667 → 443 [ACK] Seq=2057 Ack=3535 Win=65280 Len=1
15669 411.151113 3.233.158.111        192.168.0.177        TCP       66 [TCP Keep-Alive ACK] 443 → 57031 [ACK] Seq=4503 Ack=5051 Win=41472 Len=0 SLE=5050 SRE=5051
15670 411.350437 3.233.158.111        192.168.0.177        TCP       66 [TCP Keep-Alive ACK] 443 → 52667 [ACK] Seq=3535 Ack=2058 Win=40448 Len=0 SLE=2057 SRE=2058
15671 412.447715 40.79.141.155        192.168.0.177        TCP       54 443 → 59821 [RST, ACK] Seq=6959 Ack=9803 Win=0 Len=0
15672 413.742810 192.168.0.177        172.188.155.25       TCP       55 [TCP Keep-Alive] 53326 → 443 [ACK] Seq=67 Ack=60 Win=252 Len=1
15673 413.861302 172.188.155.25       192.168.0.177        TCP       66 [TCP Keep-Alive ACK] 443 → 53326 [ACK] Seq=60 Ack=68 Win=318 Len=0 SLE=67 SRE=68
15674 416.283393 104.18.39.21         192.168.0.177        TLSv1.3   78 Application Data
```

```
> Frame 1: Packet, 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{DD96E710-9A00-4275-9602-73C72C71
> Ethernet II, Src: DLinkInterna_ed:2f:e2 (e0:1c:fc:ed:2f:e2), Dst: WNC_a0:e6:9c (e8:c7:cf:a0:e6:9c)
> Address Resolution Protocol (request)
```

```
0000  e8 c7 cf a0 e6 9c e0 1c  fc ed 2f e2 08 06 00 01   ········ ··/·····
0010  08 00 06 04 00 01 e0 1c  fc ed 2f e2 c0 a8 00 01   ········ ··/·····
0020  00 00 00 00 00 00 c0 a8  00 b1                     ········ ··
```

Figure 2c: Wireshark GUI Layout and Main Components

TShark(Command-Line Interface): TShark is Wireshark's command-line counterpart. While less intuitive than the GUI, TShark is useful for:

- Automating packet capture and analysis tasks
- Running analysis on remote servers without graphical displays
- Scripting and integration with other tools
- Processing large capture files more efficiently

Example TShark command: tshark -i eth0 -f "tcp port 80" -w capture.pcap

This captures traffic on interface eth0, filters for TCP port 80 (HTTP), and saves to a file.

### 2.5 Supported Protocols

Wireshark's protocol support spans all OSI layers:

- Data Link Layer: Ethernet, Wi-Fi (802.11), PPP, Frame Relay
- Network Layer: IP (v4 and v6), ICMP, IGMP, routing protocols
- Transport Layer: TCP, UDP, SCTP
- Application Layer: HTTP, HTTPS, DNS, FTP, SSH, Telnet, SMTP, POP3, IMAP, SIP, SNMP, and hundreds more

This breadth of protocol support makes Wireshark applicable to virtually any network analysis scenario.

## 3. Installation and Configuration Steps

### 3.1 Pre-Installation Requirements

Before installing Wireshark, ensure your system meets the following requirements:

Hardware Requirements:

- Processor: Dual-core CPU at 2 GHz or higher
- RAM: Minimum 4 GB (8 GB or more recommended for analyzing large capture files)
- Storage: At least 500 MB free space for Wireshark installation; additional space needed for packet captures (depending on network traffic volume)
- Network Interface: Ethernet adapter or Wi-Fi adapter capable of packet capture

Software Requirements:

- Operating System: Windows 10/11, Ubuntu 18.04+, Debian 10+, CentOS 7+, macOS 10.13 or later
- Administrator or root privileges for packet capture (required for most network interfaces)
- Internet connection for downloading Wireshark and updates

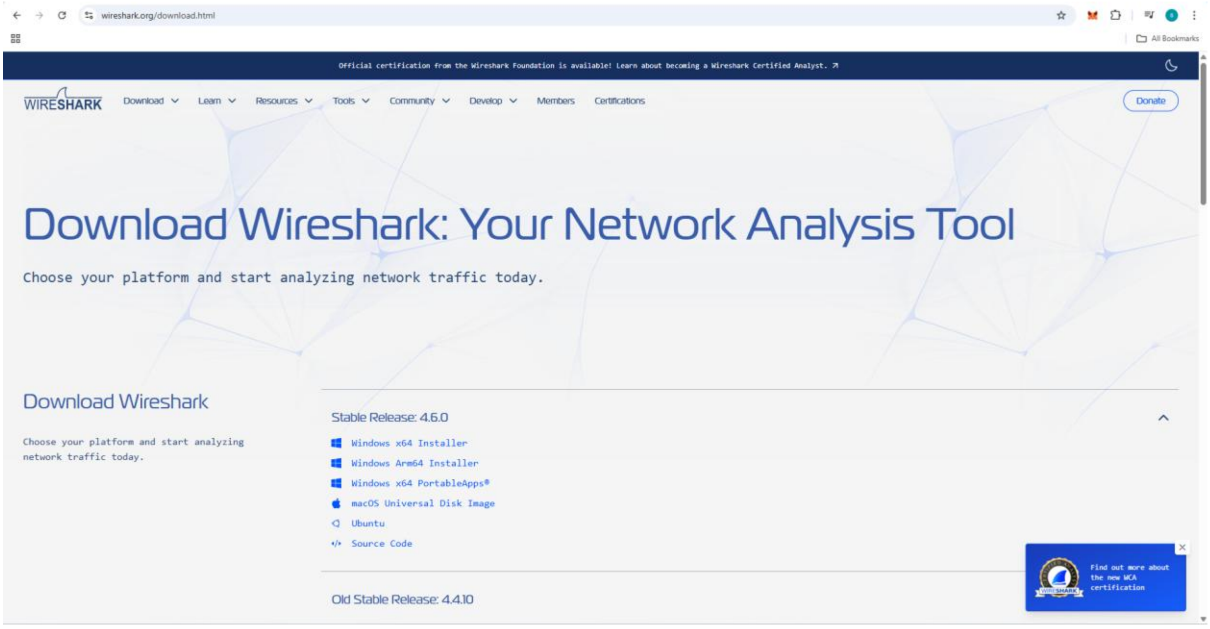
Packet Capture Libraries: Wireshark relies on underlying packet capture libraries:

- Windows: Npcap (recommended) or WinPcap
- Linux: libpcap
- macOS: libpcap or ChmodBPF

### 3.2 Installation on Windows

Step 1: Download

1. Visit the official Wireshark website: **https://www.wireshark.org/download.html**
2. Locate the Windows installer section
3. Download the latest stable version (as of 2025, version 4.x or higher recommended)
4. Choose the appropriate architecture (32-bit or 64-bit; 64-bit recommended for modern systems)



Official certification from the Wireshark Foundation is available! Learn about becoming a Wireshark Certified Analyst. ↗

Download Wireshark: Your Network Analysis Tool

Choose your platform and start analyzing network traffic today.

Download Wireshark

Choose your platform and start analyzing network traffic today.

Stable Release: 4.6.0

- Windows x64 Installer
- Windows Arm64 Installer
- Windows x64 PortableApps®
- macOS Universal Disk Image
- Ubuntu
- Source Code

Old Stable Release: 4.4.10

Figure 3a: Wireshark Download Page

Step 2: Run Installer

1. Open File Explorer and locate the downloaded .exe file
2. Right-click on the installer and select "Run as administrator"
3. If prompted by User Account Control, click "Yes" to allow the installation

Step 3: Follow Installation Wizard

1. Read and accept the license agreement
2. Select installation directory (default is C:\Program Files\Wireshark)
3. Choose components to install:
    - Wireshark GUI (required)
    - TShark (command-line tool; recommended)
    - Plugins & Extensions (optional but useful)
    - Documentation (optional)
4. Select "Install Npcap" as the packet capture mechanism (when prompted)
    - Npcap is modern, more secure, and compatible with newer Windows versions
    - Allow Npcap to run in "WinPcap API-compatible Mode" for maximum compatibility

Step 4: Npcap Installation

1. A separate Npcap installer window will appear

2. Follow the Npcap installation wizard (typically just click "Next" through defaults)
3. When asked about "WinPcap API-compatible Mode", select "Yes" for compatibility
4. Allow the installer to install Npcap Network Service

Step 5: Complete Installation

1. Once both Wireshark and Npcap are installed, the wizard will finish
2. You may be prompted to restart your computer; click "Finish" and restart if needed
3. After restart, Wireshark is ready to use

**3.3 Installation on Linux (Ubuntu/Debian)**

Step1:UpdatePackageManager
Open a terminal and run:

```
sudo apt update

sudo apt upgrade
```

Step 2: Install Wireshark and Dependencies

```
sudo apt install wireshark wireshark-common tshark
```

The installer will ask if non-root users should be able to capture packets. Select "Yes" for convenience (alternatively, you can configure this manually later).

Step3:Configure Non-Root Capture Permissions: If you want to capture packets without using sudo, add your user to the wireshark group:

```
sudo usermod -aG wireshark $USER
```

After running this command, you must log out and log back in (or restart) for the group membership change to take effect.

Step4:Verify Installation: Check that Wireshark was installed correctly:

```
wireshark --version

tshark --version
```

Both commands should return version information.

**3.4 Installation on macOS**

Step 1: Download macOS Installer

1. Visit [https://www.wireshark.org/download.html](https://www.wireshark.org/download.html)
2. Download the macOS installer (.dmg file)

3. Wait for download to complete

Step 2: Install from DMG

1. Open Finder and locate the downloaded .dmg file
2. Double-click to mount the disk image
3. A Finder window will open showing the Wireshark installer
4. Drag the Wireshark icon to the Applications folder

Step 3: Install ChmodBPF (for Packet Capture)

1. In the same DMG window, you'll find "ChmodBPF" installer
2. Double-click the ChmodBPF installer
3. Follow the installation prompts and enter your password when requested
4. ChmodBPF allows non-root packet capture on macOS

Step 4: Launch Wireshark

1. Open Applications folder
2. Find and double-click Wireshark
3. Grant any necessary permissions when prompted
4. Wireshark will launch and display the network interface selection screen

**3.5 Post-Installation Configuration**

Step1:LaunchWireshark
After installation completes successfully:

- Windows: Click Start menu and search for "Wireshark", or double-click the desktop shortcut
- Linux: Type wireshark in terminal or find it in your applications menu
- macOS: Navigate to Applications and double-click Wireshark

Step2:SelectNetworkInterface
When Wireshark opens, the first screen shows available network interfaces:

- Ethernet: Physical wired network connection (if available)
- Wi-Fi/WLAN: Wireless network connection (typically shown as "wlan0" on Linux, "en0" on macOS)
- Loopback: Local traffic on the same machine (typically "lo" or "lo0")
- Virtual Interfaces: From VPN, virtual machines, or containerization

Step3:Verify Capture Permissions: To ensure packet capture works, click on your intended capture interface. The interface name should highlight in blue, and the graph area should begin updating (showing no packets initially). If nothing happens, check:

- You have administrator/root privileges
- Npcap (Windows), libpcap (Linux), or ChmodBPF (macOS) is properly installed
- Your firewall isn't blocking Wireshark

### 3.6 Initial Configuration and Preferences

Accessing Preferences

- Windows/Linux: Edit → Preferences
- macOS: Wireshark → Preferences

Important Preference Areas:

Capture Settings

- Default interface to use for captures
- Capture filter settings
- Packet buffer size (larger for high-traffic networks)

Display Settings

- Colorize packet list (helps visually distinguish different types of traffic)
- Show milliseconds in timestamps
- Font and zoom preferences for readability

Protocol Settings

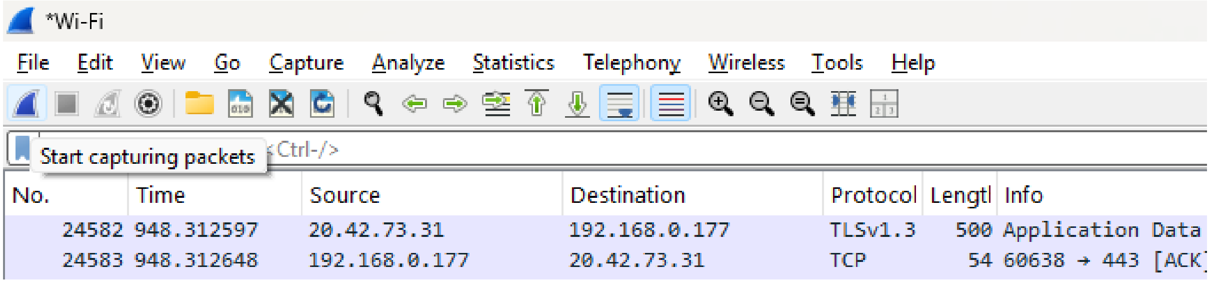
- Configure how specific protocols are displayed
- Enable/disable protocol dissectors as needed
- Set maximum packet dissection depth

### 3.7 Creating a Test Capture

To verify everything is configured correctly, perform a simple test:

Step 1: Start Capture

1. Select your active network interface from the startup screen
2. Click the blue shark fin icon (or Capture → Start)

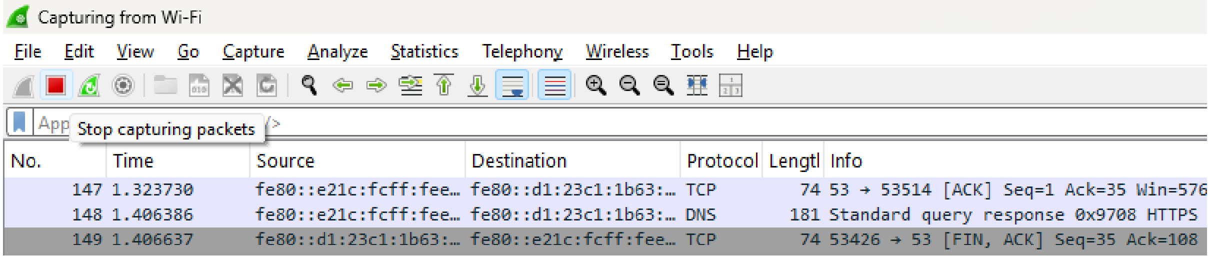

*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Start capturing packets
<Ctrl-/>

| No. | Time | Source | Destination | Protocol | Lengtl | Info |
|---|---|---|---|---|---|---|
| 24582 | 948.312597 | 20.42.73.31 | 192.168.0.177 | TLSv1.3 | 500 | Application Data |
| 24583 | 948.312648 | 192.168.0.177 | 20.42.73.31 | TCP | 54 | 60638 → 443 [ACK] |

Figure 3b: Starting a Packet Capture

Step2:GenerateTraffic
To see packets, you need network traffic. Open a web browser and visit a website (e.g., google.com).

Step3:StopCapture
After a few seconds, click the red square icon (or Capture → Stop)


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
App
Stop capturing packets
/>

| No. | Time | Source | Destination | Protocol | Lengtl | Info |
|---|---|---|---|---|---|---|
| 147 | 1.323730 | fe80::e21c:fcff:fee… | fe80::d1:23c1:1b63:… | TCP | 74 | 53 → 53514 [ACK] Seq=1 Ack=35 Win=576 |
| 148 | 1.406386 | fe80::e21c:fcff:fee… | fe80::d1:23c1:1b63:… | DNS | 181 | Standard query response 0x9708 HTTPS |
| 149 | 1.406637 | fe80::d1:23c1:1b63:… | fe80::e21c:fcff:fee… | TCP | 74 | 53426 → 53 [FIN, ACK] Seq=35 Ack=108 |

Figure 3c: Live Packet Capture in Progress

Step 4: Examine Packets

- Look at the packet list (top pane)
- Click a packet to see its details (middle pane)
- View raw data in the bottom pane

If you see HTTP packets or DNS queries, your installation is working correctly!

## 5. Experimentation and Testing

### 5.1 Basic Traffic Capture and Observation

Objective: Capture and observe normal network traffic to understand protocol behavior and baseline network activity.

Procedure:

5.1.1 Capturing Web Traffic

1. Start Wireshark and select your primary network interface
2. Click "Start" to begin capture

3. Open a web browser and navigate to an unencrypted website (e.g., example.com or your own HTTP site if available)
4. Allow traffic to flow for 10-15 seconds
5. Click "Stop" to end the capture

Analysis:
In the packet list, you should observe:

- DNS queries (protocol: DNS) to resolve the domain name
- TCP packets showing the three-way handshake (SYN, SYN-ACK, ACK)
- HTTP GET requests and responses (protocol: HTTP)
- TCP FIN packets showing connection closure

Click on an HTTP packet and examine the details pane to see:

- IP source and destination addresses
- TCP source and destination ports (usually destination port 80 for HTTP)
- HTTP method (GET, POST, etc.)
- Request headers (Host, User-Agent, etc.)

5.1.2 Capturing DNS Queries

1. Apply a display filter: dns
2. Open a terminal and run: nslookup google.com (Windows) or dig google.com (Linux/macOS)
3. Observe the DNS query and response packets

Analysis:

- DNS queries use UDP port 53
- Query packet contains the domain name being looked up
- Response packet contains the resolved IP address(es)
- TTL (Time to Live) value indicates how long the DNS record can be cached

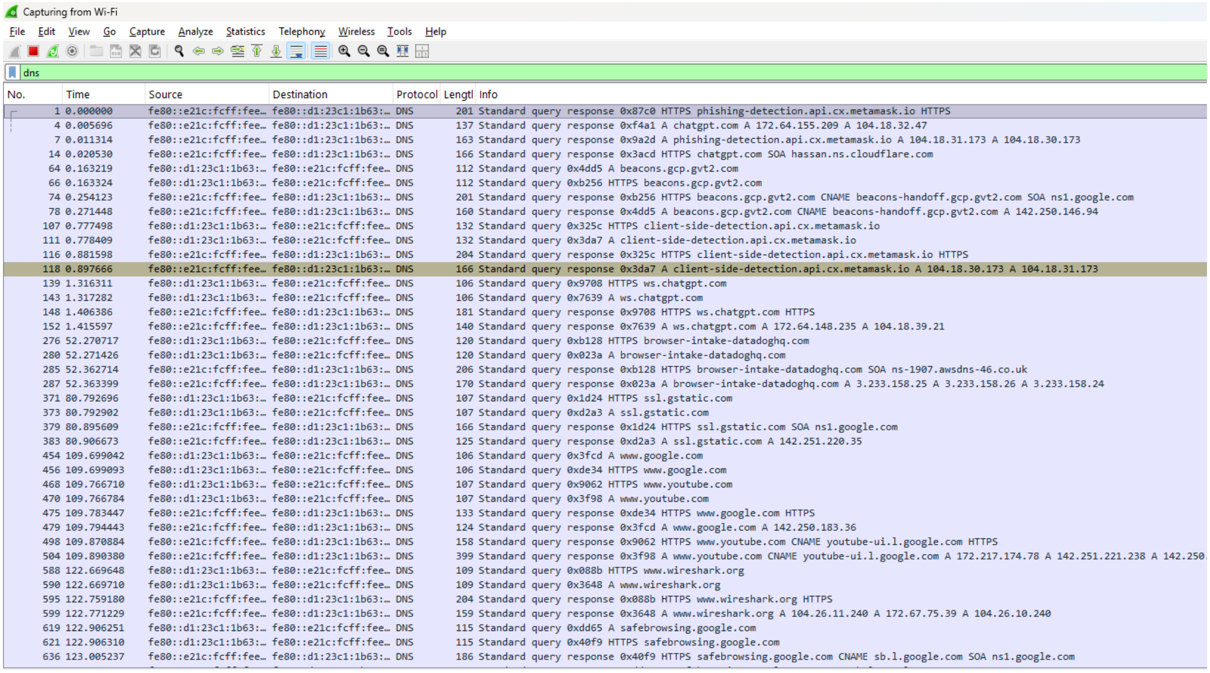
### 5.2 Filtering and Advanced Packet Selection

Objective: Use Wireshark's powerful filtering capabilities to focus on specific types of traffic.

5.2.1 Protocol-Based Filtering

Create new captures and apply the following filters to the packet list:

| Filter | Purpose | Result |
|---|---|---|
| http | Show only HTTP traffic | Displays web browsing sessions |
| https or tls | Show only encrypted HTTPS/TLS traffic | Displays secure web traffic (encrypted payloads) |
| dns | Show only DNS queries | Displays domain name resolution |
| tcp | Show only TCP protocol | Displays connection-oriented traffic |
| udp | Show only UDP protocol | Displays connectionless traffic |
| icmp | Show only ICMP (ping) | Displays ping requests and replies |


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
dns
No.   Time        Source                 Destination            Protocol Lengtl Info
    1 0.000000    fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    201 Standard query response 0x87c0 HTTPS phishing-detection.api.cx.metamask.io HTTPS
    4 0.005696    fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    137 Standard query response 0xf4a1 A chatgpt.com A 172.64.155.209 A 104.18.32.47
    7 0.011314    fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    163 Standard query response 0x9a2d A phishing-detection.api.cx.metamask.io A 104.18.31.173 A 104.18.30.173
   14 0.020530    fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    166 Standard query response 0x3acd HTTPS chatgpt.com SOA hassan.ns.cloudflare.com
   64 0.163219    fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    112 Standard query 0x4dd5 A beacons.gcp.gvt2.com
   66 0.163324    fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    112 Standard query 0xb256 HTTPS beacons.gcp.gvt2.com
   74 0.254123    fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    201 Standard query response 0xb256 HTTPS beacons.gcp.gvt2.com CNAME beacons-handoff.gcp.gvt2.com SOA ns1.google.com
   78 0.271448    fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    160 Standard query response 0x4dd5 A beacons.gcp.gvt2.com CNAME beacons-handoff.gcp.gvt2.com A 142.250.146.94
  107 0.777498    fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    132 Standard query 0x325c HTTPS client-side-detection.api.cx.metamask.io
  111 0.778409    fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    132 Standard query 0x3da7 A client-side-detection.api.cx.metamask.io
  116 0.881598    fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    204 Standard query response 0x325c HTTPS client-side-detection.api.cx.metamask.io HTTPS
  118 0.897666    fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    166 Standard query response 0x3da7 A client-side-detection.api.cx.metamask.io A 104.18.30.173 A 104.18.31.173
  139 1.316311    fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    106 Standard query 0x9708 HTTPS ws.chatgpt.com
  143 1.317282    fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    106 Standard query 0x7639 A ws.chatgpt.com
  148 1.406386    fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    181 Standard query response 0x9708 HTTPS ws.chatgpt.com HTTPS
  152 1.415597    fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    140 Standard query response 0x7639 A ws.chatgpt.com A 172.64.148.235 A 104.18.39.21
  276 52.270717   fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    120 Standard query 0xb128 HTTPS browser-intake-datadoghq.com
  280 52.271426   fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    120 Standard query 0x023a A browser-intake-datadoghq.com
  285 52.362714   fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    206 Standard query response 0xb128 HTTPS browser-intake-datadoghq.com SOA ns-1907.awsdns-46.co.uk
  287 52.363399   fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    170 Standard query response 0x023a A browser-intake-datadoghq.com A 3.233.158.25 A 3.233.158.26 A 3.233.158.24
  371 80.792696   fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    107 Standard query 0x1d24 HTTPS ssl.gstatic.com
  373 80.792902   fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    107 Standard query 0xd2a3 A ssl.gstatic.com
  379 80.895609   fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    166 Standard query response 0x1d24 HTTPS ssl.gstatic.com SOA ns1.google.com
  383 80.906673   fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    125 Standard query response 0xd2a3 A ssl.gstatic.com A 142.251.220.35
  454 109.699042  fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    106 Standard query 0x3fcd A www.google.com
  456 109.699093  fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    106 Standard query 0xde34 HTTPS www.google.com
  468 109.766710  fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    107 Standard query 0x9062 HTTPS www.youtube.com
  470 109.766784  fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    107 Standard query 0x3f98 A www.youtube.com
  475 109.783447  fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    133 Standard query response 0xde34 HTTPS www.google.com HTTPS
  479 109.794443  fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    124 Standard query response 0x3fcd A www.google.com A 142.250.183.36
  498 109.870884  fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    158 Standard query response 0x9062 HTTPS www.youtube.com CNAME youtube-ui.l.google.com HTTPS
  504 109.890380  fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    399 Standard query response 0x3f98 A www.youtube.com CNAME youtube-ui.l.google.com A 172.217.174.78 A 142.251.221.238 A 142.250.
  588 122.669648  fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    109 Standard query 0x088b HTTPS www.wireshark.org
  590 122.669710  fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    109 Standard query 0x3648 A www.wireshark.org
  595 122.759180  fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    204 Standard query response 0x088b HTTPS www.wireshark.org HTTPS
  599 122.771229  fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    159 Standard query response 0x3648 A www.wireshark.org A 104.26.11.240 A 172.67.75.39 A 104.26.10.240
  619 122.906251  fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    115 Standard query 0xdd65 A safebrowsing.google.com
  621 122.906310  fe80::d1:23c1:1b63:… fe80::e21c:fcff:fee… DNS    115 Standard query 0x40f9 HTTPS safebrowsing.google.com
  636 123.005237  fe80::e21c:fcff:fee… fe80::d1:23c1:1b63:… DNS    186 Standard query response 0x40f9 HTTPS safebrowsing.google.com CNAME sb.l.google.com SOA ns1.google.com
```

Figure 5a: Applied Display Filter Showing DNS Traffic Only

### 5.2.2 Address-Based Filtering

Filter by IP addresses or MAC addresses:

ip.addr == 192.168.1.100 # All traffic to/from a specific IP

```
ip.src == 192.168.1.100         # Traffic originating from this IP

ip.dst == 8.8.8.8              # Traffic destined to Google DNS

ip.addr == 192.168.1.0/24       # Traffic within a subnet
```

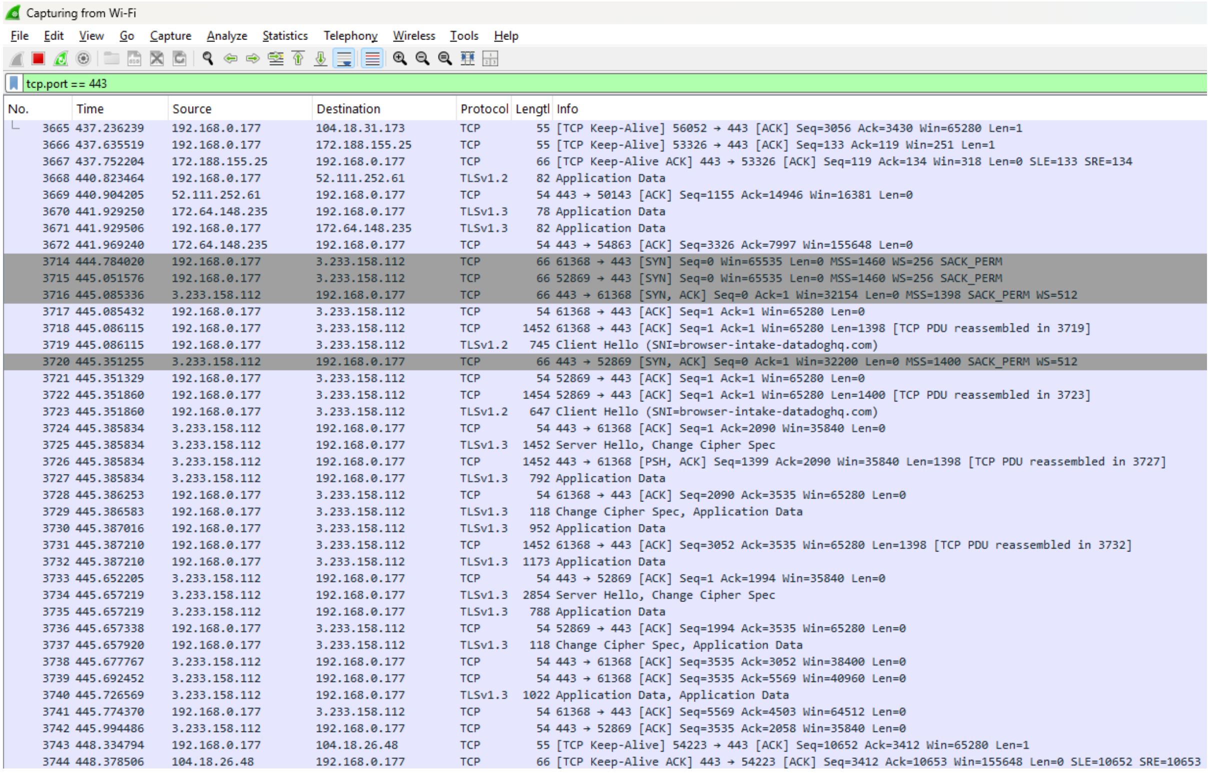
5.2.3 Port-Based Filtering

```
tcp.port == 443              # HTTPS traffic

tcp.port == 80               # HTTP traffic

tcp.port == 22               # SSH traffic

udp.port == 53               # DNS traffic
```



tcp.port == 443

| No. | Time | Source | Destination | Protocol | Lengtl | Info |
|---|---|---|---|---|---|---|
| 3665 | 437.236239 | 192.168.0.177 | 104.18.31.173 | TCP | 55 | [TCP Keep-Alive] 56052 → 443 [ACK] Seq=3056 Ack=3430 Win=65280 Len=1 |
| 3666 | 437.635519 | 192.168.0.177 | 172.188.155.25 | TCP | 55 | [TCP Keep-Alive] 53326 → 443 [ACK] Seq=133 Ack=119 Win=251 Len=1 |
| 3667 | 437.752204 | 172.188.155.25 | 192.168.0.177 | TCP | 66 | [TCP Keep-Alive ACK] 443 → 53326 [ACK] Seq=119 Ack=134 Win=318 Len=0 SLE=133 SRE=134 |
| 3668 | 440.823464 | 192.168.0.177 | 52.111.252.61 | TLSv1.2 | 82 | Application Data |
| 3669 | 440.904205 | 52.111.252.61 | 192.168.0.177 | TCP | 54 | 443 → 50143 [ACK] Seq=1155 Ack=14946 Win=16381 Len=0 |
| 3670 | 441.929250 | 172.64.148.235 | 192.168.0.177 | TLSv1.3 | 78 | Application Data |
| 3671 | 441.929506 | 192.168.0.177 | 172.64.148.235 | TLSv1.3 | 82 | Application Data |
| 3672 | 441.969240 | 172.64.148.235 | 192.168.0.177 | TCP | 54 | 443 → 54863 [ACK] Seq=3326 Ack=7997 Win=155648 Len=0 |
| 3714 | 444.784020 | 192.168.0.177 | 3.233.158.112 | TCP | 66 | 61368 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM |
| 3715 | 445.051576 | 192.168.0.177 | 3.233.158.112 | TCP | 66 | 52869 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM |
| 3716 | 445.085336 | 3.233.158.112 | 192.168.0.177 | TCP | 66 | 443 → 61368 [SYN, ACK] Seq=0 Ack=1 Win=32154 Len=0 MSS=1398 SACK_PERM WS=512 |
| 3717 | 445.085432 | 192.168.0.177 | 3.233.158.112 | TCP | 54 | 61368 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0 |
| 3718 | 445.086115 | 192.168.0.177 | 3.233.158.112 | TCP | 1452 | 61368 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=1398 [TCP PDU reassembled in 3719] |
| 3719 | 445.086115 | 192.168.0.177 | 3.233.158.112 | TLSv1.2 | 745 | Client Hello (SNI=browser-intake-datadoghq.com) |
| 3720 | 445.351255 | 3.233.158.112 | 192.168.0.177 | TCP | 66 | 443 → 52869 [SYN, ACK] Seq=0 Ack=1 Win=32200 Len=0 MSS=1400 SACK_PERM WS=512 |
| 3721 | 445.351329 | 192.168.0.177 | 3.233.158.112 | TCP | 54 | 52869 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0 |
| 3722 | 445.351860 | 192.168.0.177 | 3.233.158.112 | TCP | 1454 | 52869 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=1400 [TCP PDU reassembled in 3723] |
| 3723 | 445.351860 | 192.168.0.177 | 3.233.158.112 | TLSv1.2 | 647 | Client Hello (SNI=browser-intake-datadoghq.com) |
| 3724 | 445.385834 | 3.233.158.112 | 192.168.0.177 | TCP | 54 | 443 → 61368 [ACK] Seq=1 Ack=2090 Win=35840 Len=0 |
| 3725 | 445.385834 | 3.233.158.112 | 192.168.0.177 | TLSv1.3 | 1452 | Server Hello, Change Cipher Spec |
| 3726 | 445.385834 | 3.233.158.112 | 192.168.0.177 | TCP | 1452 | 443 → 61368 [PSH, ACK] Seq=1399 Ack=2090 Win=35840 Len=1398 [TCP PDU reassembled in 3727] |
| 3727 | 445.385834 | 3.233.158.112 | 192.168.0.177 | TLSv1.3 | 792 | Application Data |
| 3728 | 445.386253 | 192.168.0.177 | 3.233.158.112 | TCP | 54 | 61368 → 443 [ACK] Seq=2090 Ack=3535 Win=65280 Len=0 |
| 3729 | 445.386583 | 192.168.0.177 | 3.233.158.112 | TLSv1.3 | 118 | Change Cipher Spec, Application Data |
| 3730 | 445.387016 | 192.168.0.177 | 3.233.158.112 | TLSv1.3 | 952 | Application Data |
| 3731 | 445.387210 | 192.168.0.177 | 3.233.158.112 | TCP | 1452 | 61368 → 443 [ACK] Seq=3052 Ack=3535 Win=65280 Len=1398 [TCP PDU reassembled in 3732] |
| 3732 | 445.387210 | 192.168.0.177 | 3.233.158.112 | TLSv1.3 | 1173 | Application Data |
| 3733 | 445.652205 | 3.233.158.112 | 192.168.0.177 | TCP | 54 | 443 → 52869 [ACK] Seq=1 Ack=1994 Win=35840 Len=0 |
| 3734 | 445.657219 | 3.233.158.112 | 192.168.0.177 | TLSv1.3 | 2854 | Server Hello, Change Cipher Spec |
| 3735 | 445.657219 | 3.233.158.112 | 192.168.0.177 | TLSv1.3 | 788 | Application Data |
| 3736 | 445.657338 | 192.168.0.177 | 3.233.158.112 | TCP | 54 | 52869 → 443 [ACK] Seq=1994 Ack=3535 Win=65280 Len=0 |
| 3737 | 445.657920 | 192.168.0.177 | 3.233.158.112 | TLSv1.3 | 118 | Change Cipher Spec, Application Data |
| 3738 | 445.677767 | 3.233.158.112 | 192.168.0.177 | TCP | 54 | 443 → 61368 [ACK] Seq=3535 Ack=3052 Win=38400 Len=0 |
| 3739 | 445.692452 | 3.233.158.112 | 192.168.0.177 | TCP | 54 | 443 → 61368 [ACK] Seq=3535 Ack=5569 Win=40960 Len=0 |
| 3740 | 445.726569 | 3.233.158.112 | 192.168.0.177 | TLSv1.3 | 1022 | Application Data, Application Data |
| 3741 | 445.774370 | 192.168.0.177 | 3.233.158.112 | TCP | 54 | 61368 → 443 [ACK] Seq=5569 Ack=4503 Win=64512 Len=0 |
| 3742 | 445.994486 | 3.233.158.112 | 192.168.0.177 | TCP | 54 | 443 → 52869 [ACK] Seq=3535 Ack=2058 Win=35840 Len=0 |
| 3743 | 448.334794 | 192.168.0.177 | 104.18.26.48 | TCP | 55 | [TCP Keep-Alive] 54223 → 443 [ACK] Seq=10652 Ack=3412 Win=65280 Len=1 |
| 3744 | 448.378506 | 104.18.26.48 | 192.168.0.177 | TCP | 66 | [TCP Keep-Alive ACK] 443 → 54223 [ACK] Seq=3412 Ack=10653 Win=155648 Len=0 SLE=10652 SRE=10653 |

Figure 5d: Filtering Traffic by Port Number

5.2.4 Complex Filters

Combine multiple criteria with logical operators:

text

```
(tcp.port == 443 or tcp.port == 80) and ip.dst == 8.8.8.8

http and ip.src != 192.168.1.100

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

## 5.6 Documentation of Findings

Capture File Management:

1. Save all significant captures with descriptive filenames:
    - web_browsing_normal.pcap
    - port_scan_attempt.pcap
    - malware_c2_traffic.pcap

Screenshot Documentation:
For each key finding, take screenshots showing:

- The filtered packet list
- A key packet's details
- Any statistical analysis results

Written Analysis:
Document your observations:

- What protocols were present?
- What were the source/destination IPs and ports?
- What could this traffic indicate about security?
- What actions would a security team take based on this finding?

---

## 7. Learning Outcomes and Challenges

### 7.1 Learning Outcomes

Through this hands-on project with Wireshark, the following competencies were developed:

Technical Knowledge:

- Deep understanding of network protocol structures and how data is encapsulated across protocol layers (OSI model)
- Ability to interpret packet-level network data and understand what each field represents
- Knowledge of hundreds of network protocols and their characteristics
- Understanding of how network attacks appear at the packet level

Practical Skills:

- Proficiency in using Wireshark's GUI for capturing and analyzing network traffic
- Mastery of capture and display filters to isolate specific traffic of interest
- Ability to use statistical analysis tools to understand overall network behavior

- Capability to perform network forensics and reconstruct past events from packet captures

Security Analysis Abilities:

- Recognition of common attack signatures and suspicious traffic patterns
- Ability to differentiate between normal and malicious network activity
- Understanding of how to detect port scans, brute force attempts, and data exfiltration
- Knowledge of unencrypted protocols and their security risks

Professional Competencies:

- Ability to generate clear, well-documented reports of findings
- Understanding of how to integrate Wireshark into incident response procedures
- Recognition of when to escalate findings to incident response teams
- Appreciation for network visibility as a critical security control

### 7.2 Challenges Faced and Solutions

Challenge 1: Packet Capture Permission Issues

Description:
On many systems, capturing packets requires administrative or root privileges. Without proper configuration, Wireshark refuses to capture or produces empty results.

Impact:

- Beginners may become frustrated when they cannot start a capture
- Projects may be delayed if environment setup is incomplete

Solutions Implemented:

- Windows: Installed Npcap with administrator privileges and ensured "WinPcap API-compatible Mode" was enabled
- Linux: Added user account to wireshark group with sudo usermod -aG wireshark $USER and logged out/in to apply changes
- macOS: Installed ChmodBPF to grant packet capture permissions to non-root users

Challenge 2: Privacy and Ethical Considerations

Description:
Packet capture can reveal sensitive information about users (passwords, medical data, financial information, browsing habits). Improper use of Wireshark could violate privacy or laws.

Impact:

- Legal and ethical concerns when capturing and analyzing traffic
- Risk of exposure to sensitive personal data
- Potential regulatory violations (GDPR, HIPAA, etc.)

Solutions:

- Only capture traffic on networks you own or have explicit permission to monitor
- Implement strong access controls and audit logs on captured data
- Immediately stop capturing and delete files if you accidentally capture sensitive data
- Use encryption and access controls on stored capture files
- In professional settings, work within established policies and with proper oversight

## 8. Conclusion and Future Work

### 8.1 Conclusion

Wireshark stands as one of the most essential and powerful tools in the cybersecurity professional's arsenal. Through this comprehensive exploration and implementation, we have demonstrated that mastery of Wireshark enables security teams to gain unprecedented visibility into network communications—the foundation of modern security operations.

Key Takeaways:

1. Comprehensive Protocol Visibility
Wireshark's support for over 1,000 network protocols and its deep packet inspection capabilities provide analysts with the ability to understand network communications at the most granular level. Whether examining a simple HTTP request or a complex encrypted TLS handshake, Wireshark reveals the structure and flow of network data.

2. Versatility in Cybersecurity Operations
This project demonstrated Wireshark's application across multiple security domains:

- Real-time threat detection through live traffic monitoring
- Forensic investigation of past security incidents
- Compliance verification and validation of security controls
- Network troubleshooting supporting overall infrastructure security
- Security research and education

3. Accessibility and Cost-Effectiveness
As free, open-source software available on all major platforms, Wireshark democratizes network analysis. Small organizations, educational institutions, and individual researchers can access the same powerful analysis capabilities as large enterprises—removing financial barriers to network security visibility.

4. Learning Tool for Network Fundamentals
Beyond security applications, Wireshark serves as an invaluable educational tool for understanding how networks actually work. By observing protocols in action, learners develop intuition about network behavior that theoretical study alone cannot provide.

### 8.2 Practical Impact

The skills and knowledge gained through this project directly translate to professional capability:

- Security analysts can more effectively investigate incidents and identify threats
- Network engineers can more rapidly troubleshoot connectivity and performance issues
- Security architects can validate that network segmentation and encryption policies are properly implemented
- Incident response teams have a critical tool for understanding the scope and impact of security breaches

### 8.3 Future Work and Advanced Topics

While this project provides a solid foundation, Wireshark's capabilities extend much further:

Integration with Other Security Tools

- Feeding Wireshark findings into Security Information and Event Management (SIEM) systems
- Correlating Wireshark findings with Intrusion Detection System (IDS) alerts
- Using Wireshark data to validate firewall rule effectiveness
- Integration with threat intelligence feeds for automated IP/domain reputation checking

Advanced Threat Detection

- Behavioral analysis to detect anomalous patterns that don't match known signatures
- Machine learning models trained on normal traffic to identify deviations
- Encrypted traffic analysis (ETA) to find threats hidden in encrypted channels
- DNS-based threat detection finding malware callback communications

8.5 Conclusion

Wireshark represents more than just a technical tool—it embodies the principle that visibility is the foundation of security. In an era where cyber threats continue to evolve and become more sophisticated, the ability to see and understand exactly what is happening on your network remains one of the most powerful defensive capabilities available.

The hands-on experience gained through this project—from installation and configuration through complex attack detection and forensic analysis—provides a strong foundation for a career in network security. Whether working as a security analyst, incident responder, network defender, or security architect, the skills demonstrated through Wireshark proficiency are universally valued and directly applicable to protecting organizational assets and responding to security incidents.