Wireshark: A Practical Guide with Examples

Compiled for Abdo

October 14, 2025

Contents

1 Introduction

Wireshark is a graphical network protocol analyzer that lets you capture and interactively browse the traffic running on a computer network. It is widely used for troubleshooting, analysis, software and protocol development, and education.

This document provides a comprehensive overview of Wireshark, practical examples (including capture and display filters), common analysis workflows, and command-line alternatives (`tshark`). The examples are suitable for running on Windows, macOS, or Linux installations.

2 Installation

2.1 Windows

1. Download the installer from `https://www.wireshark.org` and run it.
2. During installation you may be prompted to install Npcap — accept it to enable live capture.

2.2 macOS

1. Install via Homebrew: `brew install --cask wireshark` (you may need to allow permissions in System Preferences).
2. Alternatively download a macOS installer from the Wireshark website.

2.3 Linux

- Debian/Ubuntu: `sudo apt update`
  `sudo apt install wireshark`
- Fedora: `sudo dnf install wireshark-qt`
- Add your user to the `wireshark` group (if created) or use root privileges for capture.

3 Basic Concepts

Capture
: The process of recording packets on a network interface to a file (PCAP/PCAPNG).

Packet
: A single unit of data at a particular network layer (Ethernet frame, IP packet, TCP segment, etc.).

Capture filters
: Set before capturing; implemented by libpcap/Npcap. They determine which packets are recorded. (BPF syntax)

Display filters
: Applied after capture; powerful, Wireshark-specific expressions used to filter displayed packets.

4 Capture Filters (BPF) vs Display Filters

Capture filters reduce the amount of stored data. Display filters let you refine what you see.

4.1 Examples

```
# Capture only TCP traffic to or from host 10.0.0.5
host 10.0.0.5 and tcp

# Capture only traffic on port 53 (DNS)
port 53

# Capture only packets on interface eth0 with source IP 192.168.1.10
src host 192.168.1.10 and interface eth0
```

Display filter examples (Wireshark syntax):

```
# Show HTTP requests
http.request

# Show DNS responses
dns.flags.response == 1

# Show TCP retransmissions
tcp.analysis.retransmission

# Show packets from IP 192.168.1.2 to 8.8.8.8
ip.src == 192.168.1.2 and ip.dst == 8.8.8.8
```

5 Practical Examples

5.1 Example 1: Capturing HTTP traffic

1. Start Wireshark and select the interface connected to the network.
2. Set a capture filter to reduce noise: `port 80 or port 443`
3. Visit a web page on the machine being monitored.
4. Stop capture and apply display filter: `http`

You can then *Follow TCP Stream* (right-click a TCP packet) to see the HTTP request/response body in sequence.

5.2 Example 2: Inspecting a TCP three-way handshake

Use display filters to find the initial packets of a TCP connection.

```
# Packets showing SYN, SYN/ACK, ACK for connections involving host
   10.0.0.5
tcp.flags.syn == 1 and tcp.flags.ack == 0
tcp.flags.syn == 1 and tcp.flags.ack == 1
tcp.flags.ack == 1 and tcp.flags.syn == 0
```

A small table describing the handshake:

Step	Packet	Meaning
SYN	Flags: S	Client requests connection, initial seq = x
SYN/ACK	Flags: S, A	Server acknowledges and sends its own SYN, seq = y, ack = x+1

ACK	Flags: A	Client acknowledges server's SYN, ack = y+1

5.3 Example 3: DNS query analysis

Apply display filter: `dns`

- Look at `dns.qry.name` to see the domain queried.
- Check `dns.flags.response` to differentiate queries and responses.

6 Using tshark (Command-line)

```
# Capture 100 packets to file capture.pcap
tshark -c 100 -w capture.pcap

# Read a pcap file and show HTTP requests only
tshark -r capture.pcap -Y http.request -T fields -e ip.src -e http.host
    -e http.request.uri

# Live capture with a capture filter
tshark -i eth0 -f "port 53" -w dns_capture.pcap
```

7 Common Workflows and Analysis Techniques

- **Identify high-level issues:** Use `Statistics > Protocol Hierarchy` to see which protocols dominate traffic.
- **Performance problems:** Use `Statistics > TCP Stream Graphs > Round Trip Time` and `IO Graphs`.
- **Dropped packets / retransmissions:** Display filter: `tcp.analysis.retransmission or tcp.analysis.fast`$_r$*etransmission*
- `Latency: Inspect timestamps and TCP timestamps (if available) or use Round Trip Time graphs.`

8 Advanced Features

8.1 Coloring Rules

Wireshark supports coloring rules (View ¿ Coloring Rules) to highlight packets matching display filters. This is useful to quickly spot retransmissions, errors, or specific protocols.

8.2 Decryption of TLS

If you have server private keys (RSA) or client pre-master secrets, you can configure Wireshark to decrypt TLS traffic (Preferences ¿ Protocols ¿ TLS). For modern TLS with ephemeral key exchange (ECDHE), you generally need TLS key logging (SSLKEYLOGFILE) from the client.

8.3 Custom dissectors and plugins

Wireshark supports writing plugins in C and Lua dissectors for custom protocols.

9 Security and Privacy

- Captures can contain sensitive data (credentials, private information). Store and share pcap files carefully.
- Remove or anonymize sensitive fields before sharing: `editcap -E` can edit capture timestamps; use `tshark` and scripting to redact payloads.

10 Exporting and Reporting

Wireshark can export packet bytes, packet summaries, and flow data. Use File ¿ Export Packet Dissections to create CSV, JSON, or plain text reports. `tshark` can produce custom field output (see examples above).

11 Cheat Sheet: Useful Display Filters

Filter	Meaning
`http`	Any HTTP protocol field (requests or responses)
`dns`	DNS-related packets
`tcp.port == 22`	Packets where TCP port 22 is source or destination (SSH)
`arp`	ARP packets
`icmp`	ICMP packets (ping etc.)
`tcp.analysis.retransmission`	TCP retransmissions detected by Wireshark
`frame.len > 1500`	Frames larger than 1500 bytes
`ip.addr == 192.168.1.10`	Packets where IP is either src or dst

12 Appendix: Example Capture Scenario

12.1 Scenario

You want to debug why an application cannot reach `api.example.com` from host `192.168.1.20`.

1. Start capture with capture filter: `host 192.168.1.20`
2. Reproduce the failure on the application.
3. Stop capture and apply display filters to inspect layers:
   - `dns  ip.src == 192.168.1.20` – check name resolution.
   - `tcp  ip.addr == 192.168.1.20  tcp.port == 443` – check TCP handshake.
   - `tls  ip.addr == 192.168.1.20` – inspect TLS handshake if any.
4. Use *Statistics ¿ Conversations* to see if packets are being sent/received and identify potential one-sided traffic.

13 Further Reading and Resources

- Official Wireshark documentation: `https://www.wireshark.org/docs/`
- Wireshark User's Guide (PDF) and sample captures on the Wireshark website.

- "Practical Packet Analysis" (book) by Chris Sanders — a good hands-on book.

This LaTeX file is formatted for Overleaf. To use: create a new project, paste this file as `main.tex`, and compile. For images or PCAP file inclusion, upload the PCAPs to Overleaf if needed and reference them in a local analysis (note Overleaf cannot run Wireshark — use your local machine to capture and then upload results for reporting).