

FOR ENGINEERS, SYSADMINS & SECURITY ANALYSTS

PRACTICAL WIRESHARK

From First Capture to Traffic Forensics



Capture, filter, and save like a pro

Decrypt TLS, QUIC and Wi-Fi in monitor mode

Spot scans, beacons, and leaks in real traces

Automate with tshark, pyshark, and Lua

GROUND TRUTH BOOKS

Practical Wireshark

Ground Truth Books

This book is available at <https://leanpub.com/practical-wireshark>

This version was published on 2026-10-08



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2026 Ground Truth Books

Contents

Part I: Foundations of Capture	1
Your First Capture: Installing Wireshark and Building Your Workspace	2
Installing Wireshark on Windows, macOS, and Linux	2
Choosing a capture interface	3
The three panes	4
Columns that show what matters	5
Filtering your first capture	5
Saving and reopening captures	6
Configuration profiles	7
Exercises	8
Where to Put the Sniffer: Hubs, SPAN, TAPs, Virtual Switches, and Remote Capture	10
The capture point decides what you see	10
Why a laptop on a switch port sees only its own traffic	10
SPAN and port mirroring	10
Network TAPs	10
TAP versus SPAN	10
Capturing in virtualized and cloud environments	10
Remote capture over SSH with sshdump	11
Remote capture with rpcapd	11
Permission, privacy, and the ethics of passive capture	11
Exercises	11
Capturing the Right Traffic and Saving It Well: Capture Filters, pcapng, dumpcap, and Ring Buffers	12
A Collection Plan Before a Single Frame	12
Capture filters vs. display filters	12
Berkeley Packet Filter syntax	12
Recipes and where to type them	12

CONTENTS

What pcapng adds	12
dumpcap for headless capture	12
Ring buffers and file-size limits	13
Performance and very large captures	13
Exercises	13
Managing Capture Files at Scale: capinfos, editcap, and mergecap . .	14
capinfos	14
editcap: extracting packet ranges and dropping packets	14
editcap: deduplication and timestamp adjustment	14
mergecap: combining captures and controlling the output order . .	14
A triage workflow: from a messy capture folder to a single clean file	14
Verifying that edits did what you intended	14
Exercises	15
 Part II: The Filter Language and the Command Line	 16
The Display Filter Language: Syntax, Operators, and the Field Refer-	
ence	17
Fields, protocols, and the reference	17
Comparison operators	17
Logical operators and parentheses	17
contains and matches	17
Values: addresses, ports, flags, strings	17
The GUI path and saved filters	17
Idioms: host, conversation, failures	18
Exercises	18
 Making Traffic Stand Out: Coloring Rules, Following Streams, and Time	
References	19
Reading the default coloring rules	19
Writing your own coloring rules and ordering their priority	19
Following streams to rebuild conversations	19
Using follow-stream views to spot and extract payloads	19
Time references	19
Bookmarks and navigation for long captures	20
Exercises	20
 tshark: The Terminal Wireshark	 21

CONTENTS

tshark basics	21
Capture and display filters	21
Printing fields with -T fields and -e	21
Reading only a range of packets	21
GUI features on the command line	21
Stream reconstruction and export from the command line	21
Real-world tshark one-liners for fast triage	22
Performance on very large captures	22
One pipeline, end to end	22
Exercises	22
Part III: Everyday Protocols	23
Getting Online: ARP, DHCP, and DNS	24
The boot packet story	24
ARP request/reply and variants	24
The DORA exchange and lease failures	24
DNS anatomy and correlation	24
Filters for ARP, DHCP, and DNS	24
Triage: “I can’t reach the website”	24
Exercises	25
TCP Under the Microscope: Handshakes, Retransmissions, and Sequence Analysis	26
The Three-Way Handshake as Captured	26
Sequence and Acknowledgment Numbers	28
Retransmissions: What They Mean and How to Count Them	29
Duplicate ACKs and Fast Retransmit	30
Zero Window and Window Updates	31
The Expert-Info View	32
Attributing the Cause: Client, Network, or Server	33
Exercises	34
Modern Web Protocols: HTTP/2 and HTTP/3	36
From HTTP/1.1 to HTTP/2	36
Reading the HTTP/2 dissection	36
Isolating a single stream	36
HTTP/3 on QUIC	36
QUIC dissection with and without keys	36

CONTENTS

When you need a key log	36
Exercises	37
Voice and Video: SIP, RTP, and VoIP Analysis	38
SIP call setup	38
Signaling and media in one capture	38
RTP basics	38
The RTP stream analysis window	38
VoIP Calls	38
A trace walk-through	38
Exercises	39
 Part IV: Encrypted, Wireless, and Slow Traffic	 40
Decrypting TLS and QUIC with Session Keys	41
What TLS 1.2 and 1.3 hide	41
How SSLKEYLOGFILE works	41
Generating a key log and pointing Wireshark at it	41
Decrypting TLS captures	41
Decrypting QUIC and HTTP/3	41
Verifying decryption	41
Exercises	42
 Capturing Wi-Fi in Monitor Mode	 43
Adapters, drivers, and monitor mode	43
Channel hopping	43
802.11 frame types	43
Beacons, probes, and association	43
Columns, filters, and expert info	43
Practice: Reading a join	43
Legal and privacy reality	44
Exercises	44
 Finding the Bottleneck: Expert Info, I/O Graphs, and TCP Stream Graphs	 45
The expert info window	45
From symptom to hypothesis	45
I/O graphs	45
TCP stream graphs	45

Network delay or application delay	45
Complete case: ‘the app is slow’, start to finish	45
Exercises	46
Part V: Security Use Cases	47
Detecting Reconnaissance and Port Scans	48
SYN scan anatomy	48
Incomplete SYN handshake	48
The conversations view	48
Recognizing nmap probes	48
False positives	48
From scan to attribution	48
Exercises	49
Malware Beacons and Call-Home Patterns	50
What a Beacon Is	50
Finding Beacon Candidates with tshark Summaries and Flow Statistics	50
Measuring Beacon Interval and Interval Jitter	50
DNS and HTTP Beacons versus HTTPS Beacons	50
Beacon False Positives	50
Practice: Hunting a Beacon in a Malware-Traffic-Analysis Capture	50
Exercises	51
Suspicious DNS and Credential Leaks in Real Traces	52
DNS as a channel: how tunneling works and what it looks like in queries	52
Measuring query entropy and spotting long, suspicious labels in captures	52
Finding plaintext credentials: follow-stream across HTTP, FTP, Telnet, and SMTP	52
Exporting objects to pull exfiltrated files out of a pcap	52
A forensics-style walkthrough: from odd DNS traffic to a remote shell	52
Reporting what you found and what it means	53
Exercises	53
Part VI: Automation and Extension	54
Extracting Fields and Automating Analysis with tshark	55

CONTENTS

The one-liner	55
-T fields, -e, and -E: building exactly the rows you need	55
Summarizing traffic with shell pipelines	55
Pulling fields from HTTP, DNS, and TLS conversations	55
Looping over many files and scheduling jobs	55
Why tshark fails in cron	55
A canned daily traffic report you can ship today	56
Exercises	56
Driving Captures and Analysis with Python: pyshark	57
Installing pyshark	57
Reading pcap and pcapng files	57
Layers and fields	57
Live capture	57
Scheduling pyshark captures with cron	57
Working example: flag suspicious DNS queries	57
Exercises	58
Writing Lua Dissectors for Custom Protocols	59
When you need a dissector	59
Dissector anatomy	59
Parsing the header	59
Subtrees, columns, and expert annotations	59
Port-based registration versus heuristic dissection	59
Testing against a real capture	59
Exercises	60
Appendix A: Display Filter and Capture Filter Quick Reference	61
The Two Filter Families	61
Display Filter Language	62
Ready-Made Display Filters	64
Capture Filters (BPF)	65
Converting Between Families	66
Appendix B: Command-Line Toolkit Reference	68
The Tools and Their Jobs	68
Common Behaviors	68
tshark: Capture and Read	68
dumppcap	68
capinfos	68

CONTENTS

editcap	68
mergecap	69
Companion Tools	69
File Locations and Automation Notes	69
Appendix C: Where to Find Practice Captures	70
Generate Your Own Traffic	70
Official Wireshark Sample Captures	70
Training Collections with Answer Keys	70
Malware and Intrusion Traffic	70
CTF and Competition Captures	70
Broad and Benchmark Captures	70
Finding Captures by Search	71
Verify Before You Analyze	71
Appendix D: Lab Captures and Answers to the Exercises	72
Chapter 1: Your First Capture: Installing Wireshark and Building Your Workspace	72
Chapter 2: Where to Put the Sniffer: Hubs, SPAN, TAPs, Virtual Switches, and Remote Capture	72
Chapter 3: Capturing the Right Traffic and Saving It Well: Capture Filters, pcapng, dumpcap, and Ring Buffers	72
Chapter 4: Managing Capture Files at Scale: capinfos, editcap, and mergecap	72
Chapter 5: The Display Filter Language: Syntax, Operators, and the Field Reference	72
Chapter 6: Making Traffic Stand Out: Coloring Rules, Following Streams, and Time References	73
Chapter 7: tshark: The Terminal Wireshark	73
Chapter 8: Getting Online: ARP, DHCP, and DNS	73
Chapter 9: TCP Under the Microscope: Handshakes, Retransmissions, and Sequence Analysis	73
Chapter 10: Modern Web Protocols: HTTP/2 and HTTP/3	73
Chapter 11: Voice and Video: SIP, RTP, and VoIP Analysis	73
Chapter 12: Decrypting TLS and QUIC with Session Keys	74
Chapter 13: Capturing Wi-Fi in Monitor Mode	74
Chapter 14: Finding the Bottleneck: Expert Info, I/O Graphs, and TCP Stream Graphs	74
Chapter 15: Detecting Reconnaissance and Port Scans	74

Chapter 16: Malware Beacons and Call-Home Patterns	74
Chapter 17: Suspicious DNS and Credential Leaks in Real Traces . .	74
Chapter 18: Extracting Fields and Automating Analysis with tshark .	75
Chapter 19: Driving Captures and Analysis with Python: pyshark . .	75
Chapter 20: Writing Lua Dissectors for Custom Protocols	75
Glossary	76
References	77

Part I: Foundations of Capture

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Your First Capture: Installing Wireshark and Building Your Workspace

Seeing your own traffic on the wire is the first reward of Wireshark, and it is closer than most people think: a download, a driver, and a double-click. Installs rarely fail on the installer itself. They fail on the capture driver, the interface choice, and the window layout. This chapter gets you from a clean machine to your own traffic in minutes, then builds the workspace (columns, filters, profiles) that turns browsing into analysis.

Installing Wireshark on Windows, macOS, and Linux

Installing the application follows the same pattern on all three platforms: download, run, accept defaults. But the capture driver differs per platform, and that is where installs fail. Windows needs Npcap, macOS needs ChmodBPF, and Linux needs a group membership. Get the driver right and the GUI part takes minutes.

On Windows, download the 64-bit installer from the Wireshark download page and run it [1]. The installer bundles Npcap, the driver that actually reads raw frames off your network card, and the bundle is selected by default, so accept it [1]. Accept the remaining defaults as well. The installer also offers USBPcap, which you do not need for anything in this book [2].

On macOS, Wireshark ships as a disk image signed by the Wireshark Foundation [3]. Open the .dmg, drag the Wireshark bundle into /Applications, and you are done, except for capturing [3]. Run `Install ChmodBPF.pkg` from inside the disk image. It installs a launch daemon that lets your account read raw packets without running as root [3, 4]. Skip it, and capturing requires root privileges [4]. If you prefer a package manager, `brew install wireshark` works [5].

On Linux, install from your distribution's packages. On Debian or Ubuntu:

```
1 sudo apt update
2 sudo apt install wireshark
3 sudo usermod -a -G wireshark $(whoami)
```

The installer asks whether non-superusers may capture packets. Answer yes, and it creates a `wireshark` group you must join with the `usermod` command above [6]. Some install guides also add your account to the `dialout` group with `sudo usermod -a -G dialout $(whoami)` [6]. Group membership applies at your next login, so log out and back in before testing. Distro packages are often older than the current official release, so if you need the latest build, take the installer from wireshark.org instead [7]. Binary packages are available for most Linux distributions and other Unices [8].

System requirements are modest. Wireshark needs a 64-bit processor, about 500 MB of disk, and a gigabyte of RAM to start [4]. Then the numbers scale with your workload. A busy network produces hundreds of megabytes of capture in minutes, and packet analysis is single-threaded, so a fast processor and lots of memory beat many cores [8]. Windows support tracks Microsoft's extended-support window (11, 10, Server 2022, 2019, 2016 at the time of writing). macOS needs version 11 or later [8].

Choosing a capture interface

The first window you see is the Welcome screen. It lists every interface on the machine: Wi-Fi, Ethernet, and any virtual adapters, each with a small sparkline showing live traffic [9]. Pick the interface whose sparkline shows live traffic. Skip VPN and container adapters unless they are the subject of your capture. Double-click the interface, and the capture starts [10, 11].

Stopping is just as direct: click the red square, or choose Capture > Stop [10, 9]. You will start and stop a lot, so learn the equivalents: the shark-fin toolbar button, Capture > Start, and Ctrl+E all begin a capture, and Ctrl+E also ends one [10, 5].

If you are unsure which interface carries your traffic, open Capture > Options. The dialog's interface table shows a traffic sparkline for each adapter plus the link-layer header type, snapshot length, kernel buffer size, promiscuous and monitor-mode toggles, and any capture filter attached to that interface [11]. Hover over an interface and Wireshark shows the IPv4 and IPv6 addresses bound to it, which settles most doubts [11]. One checkbox, promiscuous mode,

matters for your first capture. With it enabled, the NIC keeps every frame it receives instead of discarding the ones not addressed to your machine [11, 12]. On a hub or a mirrored port, that stream includes frames between other hosts [13]. On a switched port the NIC never receives those frames, so the setting changes nothing [14].

The command line offers the same choices. `wireshark -D` prints the interface list, `-i <interface>` selects one for capture, and with no interface given, Wireshark picks the first non-loopback adapter [15].



Monitor mode is a different animal from promiscuous mode. It captures raw Wi-Fi frames, including management and control frames, and requires an adapter that supports it plus, on Windows, an Npcap option you must enable at install time [16, 17]. You cannot fix missing hardware support with a checkbox [18].

The three panes

One packet, three views. The main window divides into a top packet list, a middle packet details tree, and a bottom packet bytes hexdump [19]. Select a packet in the list, and the other two panes fill with that packet's decoded structure and raw bytes [19, 5]. Select a field in the details, and the bytes pane highlights the exact bytes that field occupies [13]. That three-way linkage (summary, structure, bytes) is the interface's whole design [19].

The packet list is a table with one row per packet [20]. Each column holds what the highest-level protocol dissector wrote. In a frame carrying HTTP over TCP over IP over Ethernet, the Ethernet dissector writes first, the IP dissector overwrites its columns, the TCP dissector overwrites IP's, and the HTTP dissector overwrites TCP's, so the row shows HTTP information [21, 20]. To the left of the row, small symbols mark conversations (arrowheads for requests and responses, brackets for grouping), so related packets stand out at a glance [20, 22].

The packet details pane shows the selected packet as an expandable tree [19]. Each layer of encapsulation is a node, such as Frame, Ethernet II, IPv4, or TCP, and each leaf is one decoded field with its value [13]. Press Shift+Right to open a node and all its subtrees, and Ctrl+Right to expand every node in the tree [23].

The packet bytes pane shows the frame as a canonical hex dump, exactly as it crossed the wire. Each line of the dump carries a data offset, sixteen hexadecimal bytes, and the same sixteen bytes as printable ASCII, with non-printable characters shown as a period [24]. Sometimes the pane shows more than one page, reassembled data or decrypted content, and tabs at its bottom switch between them [24]. Clicking a field in the details pane scrolls the dump to the matching offset, which is how you trace a decoded value back to the raw frame [13].

Columns that show what matters

The default columns (No., Time, Source, Destination, Protocol, Length, Info) answer the question “what happened?” [5]. Analysis usually needs a sharper question than that, one about ports, response times, or slow conversations. Columns are how you ask it.

Right-click any column header and choose Column Preferences [2, 25]. The dialog lists the current columns. Pick one and edit its title, or change its type from the dropdown [12]. One useful pass renames Source to Source IP and changes its type to “Src addr (unresolved)”, then adds Src Port and Dest Port columns of the unresolved port types [2]. The “(unresolved)” variants stop Wireshark from reverse-resolving addresses, which speeds up big captures [19] and shows you what the packet actually contains rather than what DNS says about it [2]. Remove Protocol and Length if you rarely read them, and drag the remaining headers into your preferred order, in the preference dialog or directly on the column bar [2, 25]. Freeze a column so it stays visible while you scroll horizontally, and hide one by unchecking it in the header popup [26].

For a field with no matching column type, build a custom column by right-clicking the field in the Packet Details pane and choosing Apply as Column [27]. The new column’s type is custom, and you can fine-tune it in preferences by giving it a display-filter field name and a width [12].



With network name resolution enabled, a Source column shows the hostname Wireshark looked up for each address, so two packets to different IPs can display the same name [19, 12]. The unresolved types show the address the frame actually carries [2].

Filtering your first capture

Start a capture on the interface you chose, generate traffic (ping a host, load a web page, let background chatter accumulate), then stop. The capture now holds a jumble: your traffic, other hosts' traffic, broadcasts, and background noise. Your job is to isolate your own traffic, and the tool for that is a display filter.

A display filter hides packets after capture, while a capture filter drops them before capture [28]. The distinction matters because hidden packets are recoverable and dropped packets are gone forever [28]. Reach for a capture filter when volume or exposure is a real problem, such as a long capture window or a high-rate link [28]. You enter it in BPF syntax on the Welcome screen or in Capture > Options, where host 192.168.1.42 records only that host's traffic [28, 29]. For a first capture, though, capture broadly and filter later.

Type into the filter toolbar, and the toolbar turns green once your expression parses or red when it does not, so you can fix a mistake before pressing Enter [30]. Press Enter, and the list narrows instantly [12]. The workhorse filter is your own address:

```
1 ip.addr == 192.168.1.42
```

`ip.addr` matches the address as either source or destination, so you see traffic in both directions without writing both clauses [31, 32]. When direction matters, split the field: `ip.src == 192.168.1.42` for outgoing, `ip.dst == 192.168.1.42` for incoming [31, 5]. Combine conditions with the filter language's operators (and, or, not, and comparisons like `==`), so `ip.addr == 192.168.1.42 && tcp.port == 443` shows only your HTTPS traffic [19, 32]. The filter toolbar autocompletes field names as you type, which is the fastest way to learn what fields exist [12].

One useful drill filters to your own IP, then looks at the Protocol column. A quiet minute of web browsing should show DNS, TLS, and occasionally QUIC or ARP. Each is a question you can ask with one more filter, such as `dns` for the name resolutions or `tcp.flags.reset == 1` for connections being torn down [32]. Run one now, then keep the filtered view on screen. The next section saves exactly what a filter shows.

Saving and reopening captures

Save before you close anything. File > Save As opens a dialog that lets you save the whole capture, only the packets currently displayed by a filter, or a numbered packet range [19]. Saving the displayed subset is a clean way to hand someone a focused trace instead of the full firehose [19].

The default format is pcapng, and it is the right default. Wireshark has saved files as pcapng by default since version 1.8. The older pcap format is what libpcap-based tools like tcpdump use [33]. Dumpcap, the capture engine under Wireshark, also writes pcapng by default [34].

The format matters because pcapng keeps more than pcap can. A pcapng capture is a chain of blocks. An Interface Description Block records, per interface, the link-layer type, the operating system, and the hardware description. Each packet lives in an Enhanced Packet Block with its own timestamp [35]. Per-interface statistics (packets received, packets dropped by the interface, packets dropped by the OS) can ride along [35]. Packet comments survive, where pcap has no place to put them [26]. If you capture from more than one interface at once, Wireshark must write pcapng, because pcap stores a single link-layer type for the whole file [34, 11]. On the practical side, process info, which application sent each packet, is recorded in the pcapng file when you enable it [11].

Reopen a saved capture any of three ways: File > Open, double-clicking the file, or on the command line `wireshark -r capture.pcapng` [36, 9]. The raw frames and per-interface metadata reopen unchanged. Dissection settings and window layout come from your active profile, not from the file.

Configuration profiles

So far you have customized columns and learned to read the three panes. A configuration profile is how you keep all of that organized by job, so a DNS troubleshooting session and a security review can have different windows without you rebuilding either one. Each profile stores its own preferences, capture and display filters, filter macros, coloring rules, disabled protocols, decode-as assignments, and recent settings such as pane sizes and column widths [37]. Switch profiles and the whole layout changes with you.

Create one from `Edit > Configuration Profiles`, or press `Shift+Ctrl+A` (`Shift+Cmd+A` on macOS) [37]. Click the `+` button, type a name, and you have an empty profile [27, 2]. The active profile's name sits in the status bar at the bottom of the window. Click it to switch profiles on the fly [27, 37]. Changes you make while a profile is active (the columns from the previous section, coloring rules, saved filters) are saved into it [37, 38]. You are always working inside some profile, so customize once and the next session starts where you left off [38].

Delete a profile the same way you made one, by selecting it in the dialog and clicking the minus button [37]. Wireshark will not delete the Default profile or the global profiles that ship with the application. Asking to delete Default merely resets it [37]. Share a crafted layout by exporting the profile to a zip archive and importing it on the other machine. A team with one profile set starts each investigation from the same window [27, 37, 38].

Profiles are folders of plain-text configuration files, stored under a profiles directory in your personal configuration folder, such as `~/.config/wireshark/profiles` on Linux [39]. `tshark -G folders` prints the exact paths on any platform when you need to back them up [39]. You can also tell Wireshark to switch profiles automatically. Give a profile an auto-switch filter, and opening a capture whose packets match it loads that profile for you [37].

A DNS troubleshooting profile shows the idea. Create the profile, add a “Response In” column by right-clicking the field in the details pane, remove the columns you never read, then add two filter buttons: one showing DNS responses, one flagging response times over half a second [27]. Later DNS investigations start with that window and those buttons.

So far you have captured on your own machine, which records only your own traffic. Chapter 2 asks where to capture when the problem lives between other machines: on a switch port, a mirror port, a TAP, or a remote host.

Exercises

1. Open `lab/http.cap`. The machine that recorded it is `145.254.160.237`. Use the filter toolbar to count how many frames that host sent, how many it received, and how many either way. Then add `tcp.port == 80` to the either-way filter with `&&`. How many frames disappear, and what were they?

2. In `lab/http.cap`, frames 4, 6, and 38 all belong to the same web page download, but the Protocol column shows HTTP for frame 4, TCP for frame 6, and HTTP/XML for frame 38. Select each one, expand the packet details tree, and explain why the column differs even though all three frames carry HTTP data.
3. Build the DNS troubleshooting view the chapter describes, using `lab/dns.cap`. Right-click the response time field (`dns.time`) in a response and choose Apply as Column, then apply the filter `dns.time > 0.5`. Which query was slow, what record type did it ask for, and which frame holds the query it answers?
4. A colleague wants only the ad-server part of `lab/http.cap`: the DNS lookup and the traffic to 216.239.59.99. Filter for it, then use File > Export Specified Packets (or Save As with “displayed” selected) to save just those packets as `pcapng`. Open the new file. Its frame numbers and times don’t match the original, and the HTTP connection in it has no SYN. Work out which of these differences come from saving a filtered subset and which were already true of the original capture.

Where to Put the Sniffer: Hubs, SPAN, TAPs, Virtual Switches, and Remote Capture

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

The capture point decides what you see

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Why a laptop on a switch port sees only its own traffic

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

SPAN and port mirroring

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Network TAPs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

TAP versus SPAN

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Capturing in virtualized and cloud environments

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Remote capture over SSH with sshdump

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Remote capture with rpcapd

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Permission, privacy, and the ethics of passive capture

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Capturing the Right Traffic and Saving It Well: Capture Filters, pcapng, dumpcap, and Ring Buffers

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

A Collection Plan Before a Single Frame

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Capture filters vs. display filters

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Berkeley Packet Filter syntax

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Recipes and where to type them

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

What pcapng adds

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

dumpcap for headless capture

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Ring buffers and file-size limits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Performance and very large captures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Managing Capture Files at Scale: capinfos, editcap, and mergecap

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

capinfos

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

editcap: extracting packet ranges and dropping packets

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

editcap: deduplication and timestamp adjustment

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

mergcap: combining captures and controlling the output order

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

A triage workflow: from a messy capture folder to a single clean file

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Verifying that edits did what you intended

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Part II: The Filter Language and the Command Line

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

The Display Filter Language: Syntax, Operators, and the Field Reference

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Fields, protocols, and the reference

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Comparison operators

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Logical operators and parentheses

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

contains and matches

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Values: addresses, ports, flags, strings

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

The GUI path and saved filters

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Idioms: host, conversation, failures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Making Traffic Stand Out: Coloring Rules, Following Streams, and Time References

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Reading the default coloring rules

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Writing your own coloring rules and ordering their priority

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Following streams to rebuild conversations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Using follow-stream views to spot and extract payloads

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Time references

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Bookmarks and navigation for long captures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

tshark: The Terminal Wireshark

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

tshark basics

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Capture and display filters

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Printing fields with -T fields and -e

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Reading only a range of packets

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

GUI features on the command line

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Stream reconstruction and export from the command line

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Real-world tshark one-liners for fast triage

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Performance on very large captures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

One pipeline, end to end

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Part III: Everyday Protocols

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Getting Online: ARP, DHCP, and DNS

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

The boot packet story

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

ARP request/reply and variants

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

The DORA exchange and lease failures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

DNS anatomy and correlation

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Filters for ARP, DHCP, and DNS

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Triage: “I can’t reach the website”

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

TCP Under the Microscope:

Handshakes, Retransmissions, and Sequence Analysis

Wireshark's expert analysis tags TCP packets with labels: [TCP Retransmission], [TCP Dup ACK], [TCP Out-Of-Order], [TCP ZeroWindow]. They read like verdicts, but each is an inference Wireshark draws from the sequence numbers it has seen so far in one stream. Misread one, and you blame a server for a network problem, or blame the network for a server problem. This chapter shows the sequence-number math behind each label and how to attribute the cause to the client, the network, or the server. It starts with the TCP handshake, because the handshake sets the sequence numbers every label is measured against.

The Three-Way Handshake as Captured

A completed handshake starts with a SYN, and the Info column names each of its three packets for you [29, 199]. The client opens with a SYN that proposes the connection and its own starting sequence number. The server answers SYN, ACK to accept and propose its own. The client closes the loop with a final ACK, and data begins to flow [29, 199]. The first three frames of `lab/http.cap`, a web download captured at the client, are a healthy handshake [155]:

```
1 $ tshark -r http.cap -c 3
2   1   0.000000 145.254.160.237 → 65.208.228.223 TCP 62 3372 → 80 [SYN] Seq=0
   → Win=8760 Len=0 MSS=1460 SACK_PERM
3   2   0.911310 65.208.228.223 → 145.254.160.237 TCP 62 80 → 3372 [SYN, ACK]
   → Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 SACK_PERM
4   3   0.911310 145.254.160.237 → 65.208.228.223 TCP 54 3372 → 80 [ACK] Seq=1
   → Ack=1 Win=9660 Len=0
```

Start with the numbers, because the labels only restate them. The SYN's Seq=0 is the client's proposed initial sequence number. The SYN-ACK's Ack=1

acknowledges that proposal by counting it as one consumed byte, and the ACK's Seq=1 Ack=1 closes the loop [19, 199]. Both SYN and FIN consume one sequence position each, so this bookkeeping applies to the connection's setup and teardown as much as to its data [199, 200].

Captured at the client, the SYN to SYN-ACK interval is a full round-trip to the server, a free RTT measurement that no ICMP ping can fake. Routers often deprioritize ICMP, so this timing reflects what the application's own traffic experiences [29, 148, 201]. Captured mid-path, that same interval covers only part of the journey, so measure the SYN through final-ACK span, which includes both halves [201]. Wireshark records the SYN-to-final-ACK interval as `tcp.analysis.initial_rtt` and sets it only once the handshake completes, so a conversation carrying that field finished its handshake [202]. In `http.cap` the SYN-ACK arrives 0.911 s after the SYN and the final ACK leaves in the same instant, so `tcp.analysis.initial_rtt` is 0.911 s as well [203]. To find half-open connections, SYN/SYN-ACK without the final ACK, query `(tcp.flags==0x12)` and not `tcp.analysis.initial_rtt` [202]. In tshark, add -2, because a single pass has not yet seen the final ACK when it reaches the SYN-ACK.

Three connection failures show up in the handshake itself. A SYN with no SYN-ACK means something drops it without answering, usually a firewall. A SYN answered by RST, ACK means the port is closed and actively refused. Repeated SYNs at growing intervals are the client retrying a path that will not answer [29]. Isolate all of them with `tcp.flags.syn == 1 || tcp.flags.fin == 1 || tcp.flags.reset == 1`, which shows where each connection in the file starts and ends [29]. `tcp.flags.syn == 1` also matches SYN-ACKs. For SYN-only traffic, use `tcp.flags == 0x02` as a load filter in the open-file dialog [204].

Teardown: FIN versus RST

Graceful close mirrors the handshake. The side with nothing left to send transmits FIN, the other side ACKs it, then the reverse happens, four packets total. Each FIN consumes one sequence position, just as the SYN did [19, 205]. Filter with `tcp.flags.fin == 1`. An RST is different. It aborts the connection immediately with no close handshake, appearing after a crash, an application abort, or a refused connection [205, 146]. `tcp.flags.reset == 1` isolates the resets, including the [RST, ACK] refusals that punctuate

your captures. To classify whole conversations rather than packets, use the `tcp.completeness` field, whose bits are 1=SYN, 2=SYN-ACK, 4=ACK, 8=data, 16=FIN, and 32=RST; `tcp.completeness==7` (SYN + SYN-ACK + ACK) finds handshake-only connections [206].

Sequence and Acknowledgment Numbers

The handshake looks trivial because Wireshark shows you **relative** sequence numbers by default. It subtracts the stream's initial sequence number so the first segment reads `Seq=0` [19, 205]. The raw 32-bit values are still there. Uncheck "Relative sequence numbers" under Edit > Preferences > Protocols > TCP to see them [9, 205]. Keep relative numbers on for everyday reading, and keep the preference path in mind when two engineers disagree about a trace.

TCP numbers bytes, and it numbers each direction independently [200]. The sequence number of a segment names the first byte it carries. The acknowledgement number names the next byte the receiver expects, and it is cumulative. One value confirms all earlier bytes, which is why losing a pure ACK is cheap [200]. That two of the three handshake packets show `Seq=0 Ack=1` and the third `Seq=1 Ack=1` is just this rule applied to the SYN's one consumed byte [19, 199].

Once data flows, the numbers advance by segment size. These are frames 4 to 12 of `http.cap`, with the columns frame, source, relative sequence number, acknowledgement number and payload length [207]:

```

1  4  145.254.160.237 1    1    479 GET /download.html HTTP/1.1
2  5  65.208.228.223 1    480 0    80 → 3372 [ACK] Seq=1 Ack=480 Win=6432 Len=0
3  6  65.208.228.223 1    480 1380  HTTP/1.1 200 OK
4  7  145.254.160.237 480 1381    0    3372 → 80 [ACK] Seq=480 Ack=1381 Win=9660
   → Len=0
5  8  65.208.228.223 1381    480 1380    80 → 3372 [ACK] Seq=1381 Ack=480
   → Win=6432 Len=1380
6  9  145.254.160.237 480 2761    0    3372 → 80 [ACK] Seq=480 Ack=2761 Win=9660
   → Len=0
7 10  65.208.228.223 2761    480 1380    80 → 3372 [ACK] Seq=2761 Ack=480
   → Win=6432 Len=1380
8 11  65.208.228.223 4141    480 1380    80 → 3372 [PSH, ACK] Seq=4141 Ack=480
   → Win=6432 Len=1380
9 12  145.254.160.237 480 5521    0    3372 → 80 [ACK] Seq=480 Ack=5521 Win=9660
   → Len=0

```

The GET carries bytes 1 to 479, so the server's ACK in frame 5 says 480, the next byte it expects. Each server segment carries 1380 bytes and moves the sequence number up by that much: 1, 1381, 2761, 4141. Frame 12's Ack=5521 covers frames 10 and 11 at once, because one cumulative ACK confirms all the bytes below it. When you suspect a byte is lost, find the ACK that stops advancing. The retransmission's sequence number will equal that stalled ACK number, and that equality pins the exact missing segment [19].

Retransmissions: What They Mean and How to Count Them

TCP has no header flag that says “this is a retransmission.” Wireshark infers the label by tracking each stream's state (comparing a segment's sequence number, IP ID, addresses, ports, and arrival time against what it has already seen) and prepends [TCP Retransmission] to the Info column when a segment repeats [208, 209]. The dissector runs this analysis once per packet when the file opens, and the “Analyze TCP sequence numbers” toggle in Preferences > Protocols > TCP turns it on or off for the whole file [206].

A **retransmission** follows a timeout, when the sender's retransmission timer (RTO) expires before an ACK arrives. The sender re-sends and doubles the wait next time, in exponential backoff [209, 19]. A **fast retransmission** happens reactively, before the timer fires, after the sender collects three duplicate ACKs [209, 138, 200]. Fast recovery after duplicate ACKs is ordinary loss handling. Waiting out a full RTO means no feedback path, which points to severe loss, a broken return path, or an endpoint problem [210].

A retransmission proves only that TCP's sender decided to resend bytes. It does not prove the packet was dropped on the wire. The copy may have arrived late, a **spurious retransmission** after an over-aggressive timeout, or the capture itself may have dropped a frame, leaving a gap that later flags a segment as a retransmission even though the network delivered it [208, 138, 200].

Counting and the Rate

A rate based on bytes is more precise than one based on packets, but packet counts are a serviceable estimate. One way to get both numbers from the command line:

```
1 tshark -r capture.pcap -q -z io,stat,1,"tcp.analysis.retransmission"  
2 TOTAL=$(tshark -r capture.pcap | wc -l)  
3 RETRANS=$(tshark -r capture.pcap -Y 'tcp.analysis.retransmission' | wc -l)
```

The `-z io,stat,1` tap prints packet counts per one-second interval restricted to the filter. Dividing `RETRANS / TOTAL` gives a rough rate, and the denominator must count every packet in the file [138]. Filter the retransmits by direction, `tcp.analysis.retransmission` and `ip.src == 10.0.0.50`, before you judge either side [138].

Then open each one before you quote the rate. In `http.cap` the filter matches one frame of 43, frame 36, a naive rate of 2.3% [211, 212]. Frame 36 also carries the spurious-retransmission flag, and frame 37 is the client's duplicate ACK for data it already held [211, 213]. Nothing was lost, so the true loss rate of this capture is zero.

For timing, learn what a normal backoff looks like. A Windows sender retransmits up to five times with doubling gaps, roughly 9.6 seconds total, before giving up [19]. A VNC thin-client capture shows that a fast retransmit does not guarantee recovery. The server resent the segment after the third duplicate ACK, but the client still did not acknowledge it, so the sender fell back to slow timeout retransmissions. Closer analysis found that the server's stack mishandled ACKs that split a segment and used a fixed RTO of about 500 ms. The slow recovery was sender behavior, made worse by the client's tiny receive window [214].

Duplicate ACKs and Fast Retransmit

A duplicate ACK is the receiver repeating the same acknowledgment value because it detected a gap. The ACK number is lower than the last byte the sender transmitted, so the receiver re-acknowledges it once for every segment that arrives past the hole [209, 215]. The Info column annotates the count. [TCP Dup ACK 8796#16] means the receiver sent the identical ACK seventeen times [215]. Occasional duplicates are normal on any path with slight reordering. A long run means the receiver is parked at a gap [215, 216].

Three consecutive duplicate ACKs, with no intervening progress, is the sender's evidence that the segment at the ACK number is probably lost, so it fast-retransmits that single segment instead of waiting out the RTO [209, 216, 200]. This three-ACK threshold is deliberate. One or two duplicates are

consistent with reordering, and three-plus strongly indicate real loss [217]. Distinguish the two outcomes by what follows:

- **Loss:** the dup-ACK sequence ends in a fast retransmission (`tcp.analysis.fast_retransmission`).
- **Reordering:** the expected segment finally arrives and the gap closes without any retransmission [216].

Wireshark exposes the count as a filterable field, so highlight only ACKs past the threshold with `tcp.analysis.duplicate_ack_num >= 3` [216].

Out-of-order is the label for the segment that arrives with a sequence number *lower* than the previously seen packet on that connection [209]. It is not itself loss. The duplicate ACKs may just be your only view of it. Possible causes span network reordering, parallel paths, load-balanced links, capture behavior, and loss where the original arrives late [140]. The related **Previous segment not captured** label (`tcp.analysis.lost_segment`) is stronger. It marks the packet whose sequence number jumped past the next expected value, meaning one or more earlier segments were not observed, usually followed by a retransmission of that range [209, 19].

`lab/smtp.pcap` shows how far an out-of-order label can be from reordering [218]. The client sends four 1460-byte segments, frames 22 to 25. A router answers with four ICMP *Fragmentation needed* messages, frames 26, 28, 29 and 30, because those segments are too big for the next link. The client then resends the same data in 1452-byte segments. Wireshark labels the first resend, frame 27, a retransmission, and the next four, frames 32, 33, 35 and 36, out-of-order, because their sequence numbers are lower than the highest already seen. Nothing was reordered and nothing was lost on the wire. This is Path MTU Discovery doing its job, and only the ICMP messages beside the labels show it.

Interpret these labels relative to your capture point. The analysis runs once per packet in capture order, so the same loss looks different from client side, server side, or mid-path. Wireshark even offers a preference to let fast retransmission supersede the out-of-order interpretation in ambiguous cases [206].

Zero Window and Window Updates

The 16-bit window field in every TCP header advertises how many bytes the receiver can still accept, the free space in its receive buffer, once the field's value is multiplied by the scale factor negotiated in the handshake [206, 210]. **Zero window** is the receiver advertising zero, telling the sender “do not send, my buffer is full.” The sender must pause until a window update reopens [209, 206]. Some zero windows are normal. A printer may zero its window while reversing a sheet, but in general a zero window means the receiving application is not draining the socket buffer fast enough, a resource or capacity problem on that host [206, 219].

Because the zero window can evaporate while the pause lasts, TCP probes it. The sender transmits one byte, a **ZeroWindowProbe** (look for `tcp.analysis.zero_window_probe`), to elicit an ACK, doubling its persist timer if the window stays closed [209, 220]. Receiving data anyway after advertising zero is a **ZeroWindowViolation**, and a useful accusation of a misbehaving sender [209]. When the application finally consumes bytes, the receiver sends a pure **WindowUpdate** advertising the reopened space, typically the packet you see just before the data resumes [209].

The mirror-image label is **WindowFull**, set on a segment whose payload exactly fills the receiver's advertised window. After that segment, the sender must stop until ACKs advance. Unlike a zero window, this occurs without any receiver complaint, whenever the window is too small for the round-trip path, the classic small-buffer-on-high-latency case [209, 221]. Filter both conditions together with `tcp.analysis.window_full` or `tcp.analysis.zero_window`.

Then separate them with `tcp.window_size == 0`, which isolates the zero-window advertisements, a receiver-side verdict. What remains are Window Full segments from the sender [222, 210]. Window scaling deserves its own check. When it was not negotiated in the SYN exchange, both sides are stuck with a 64 KB ceiling regardless of bandwidth-delay product [210]. The TCP Stream Graphs > Window Scaling plot makes the story visual. A flat bytes-out line under a falling receive-window line is the receiver throttling the sender [223].

The Expert-Info View

Analyze > Expert Information condenses the file's anomalies into one table grouped by severity (chat, notes, warnings, and errors) alongside the packet numbers, and clicking an entry jumps your packet list to the frame [224, 225]. Sort by severity, Errors first, for a health summary in one screen. Out-of-order segments and resets warn, retransmissions and duplicate ACKs note [225, 138]. On the command line, `tshark -r capture.pcap -q -z expert` prints the same grouping with occurrence counts [225, 159].

Two cautions, both from Wireshark's own documentation. Expert information is a hint, not a diagnosis. Its presence does not prove a problem, and its absence does not prove health [224]. The counts are only as good as the capture, because the same analysis that mislabels a duplicated capture frame as a retransmission also inflates the expert table [208]. Use the dialog to find candidate packets. Right-click an item to Prepare a Filter from it, which yields `tcp.analysis.*` conditions you can reuse. `tcp.analysis.flags` alone selects every packet with any TCP-analysis annotation, and it is the basis of Wireshark's "Bad TCP" coloring rule [225, 138].

Attributing the Cause: Client, Network, or Server

Now connect the labels into a verdict. A methodical sequence works. State the complaint in plain language, map it to a host and time, check Conversations and Endpoints for the busy flows, review Expert Information, form one hypothesis, and test it against the packets before moving on [53]. The filter combinations below are the hypotheses.

- **Network-path loss.** When `tcp.analysis.retransmission` appears without `tcp.analysis.window_full`, the receiver has buffer headroom and keeps up, yet segments must be resent. Look for dup-ACK/fast-retransmit pairs, then suspect congestion, a lossy link, or duplex mismatch on the path [210].
- **Receiver/application bottleneck.** When `tcp.analysis.zero_window` or `tcp.analysis.window_full` shows up, the receiver cannot drain its socket. Correlate with that host's CPU, disk, and `recv()` activity. Increasing buffers merely hides the slowness [219, 226].

- **Sender- or path-severity problem.** When `tcp.analysis.retransmission` appears without `tcp.analysis.fast_retransmission`, the sender waited out full RTOs. Too few segments may have followed the loss to generate three duplicate ACKs (tail loss, small windows). Otherwise, suspect ACK loss, a broken return path, or very severe loss [210].

Always test both directions. Loss is often asymmetric, and capturing only one side of a conversation hides which ACKs actually arrived [140]. Also check where you captured. “Previous segment not captured” may mean your monitor port is oversubscribed rather than that the network lost bytes [227, 140]. One capture rarely proves where a packet was lost. Sender- and receiver-side captures (or captures from both ends of the conversation) together separate an on-wire drop from a late or duplicated copy [200]. These labels are not TCP header fields but Wireshark’s reading of the stream [206], so filter `tcp.analysis.retransmission` and `tcp.analysis.zero_window` in each direction and confirm each label against the sequence numbers before you name the cause.

Exercises

1. Open `lab/tcp-ecn-sample.pcap` and read the handshake and the first request. What is the connection’s initial RTT as Wireshark records it? The GET in frame 4 carries 161 bytes starting at relative sequence 1: predict the acknowledgement number the server’s next segment will carry, then predict the client’s ACK for the server’s first 256-byte reply. Check both against frames 5 and 6.
2. In `lab/dns-remoteshell.pcap`, filter on `tcp.port == 21`. You’ll see SYNs labelled [TCP Retransmission] and [TCP Port numbers reused], and every answer is [RST, ACK]. How many connection attempts did the client really make, what did the server say to each, and why do neither of those two labels mean what they seem to? Use `tcp.completeness` to classify the connections.
3. `lab/http2-data-reassembly.pcap` is a fast HTTPS download with exactly one [TCP Retransmission], at frame 38. Was any data lost? Read the sequence numbers of frames 37 and 38, then the acknowledgement and SACK fields of frame 39, and explain what they prove. Also say what the [RST, ACK] in the last frame means for this connection.

4. `lab/tcp-ecn-sample.pcap` moves about 83 kB in about 94 seconds, and the expert summary shows no retransmissions, no zero windows, and no Window Full. Attribute the slowness to the client, the network, or the server. Compare the most data the server ever had in flight with the smallest window the client advertised, and look for TCP and IP flags beyond the ones this chapter names.

Modern Web Protocols: HTTP/2 and HTTP/3

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

From HTTP/1.1 to HTTP/2

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Reading the HTTP/2 dissection

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Isolating a single stream

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

HTTP/3 on QUIC

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

QUIC dissection with and without keys

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

When you need a key log

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Voice and Video: SIP, RTP, and VoIP Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

SIP call setup

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Signaling and media in one capture

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

RTP basics

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

The RTP stream analysis window

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

VoIP Calls

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

A trace walk-through

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Part IV: Encrypted, Wireless, and Slow Traffic

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Decrypting TLS and QUIC with Session Keys

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

What TLS 1.2 and 1.3 hide

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

How SSLKEYLOGFILE works

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Generating a key log and pointing Wireshark at it

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Decrypting TLS captures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Decrypting QUIC and HTTP/3

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Verifying decryption

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Capturing Wi-Fi in Monitor Mode

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Adapters, drivers, and monitor mode

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Channel hopping

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

802.11 frame types

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Beacons, probes, and association

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Columns, filters, and expert info

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Practice: Reading a join

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Legal and privacy reality

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Finding the Bottleneck: Expert Info, I/O Graphs, and TCP Stream Graphs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

The expert info window

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

From symptom to hypothesis

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

I/O graphs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

TCP stream graphs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Network delay or application delay

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Complete case: ‘the app is slow’, start to finish

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Part V: Security Use Cases

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Detecting Reconnaissance and Port Scans

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

SYN scan anatomy

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Incomplete SYN handshake

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

The conversations view

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Recognizing nmap probes

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

False positives

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

From scan to attribution

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Malware Beacons and Call-Home Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

What a Beacon Is

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Finding Beacon Candidates with tshark Summaries and Flow Statistics

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Measuring Beacon Interval and Interval Jitter

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

DNS and HTTP Beacons versus HTTPS Beacons

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Beacon False Positives

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Practice: Hunting a Beacon in a Malware-Traffic-Analysis Capture

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Suspicious DNS and Credential Leaks in Real Traces

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

DNS as a channel: how tunneling works and what it looks like in queries

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Measuring query entropy and spotting long, suspicious labels in captures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Finding plaintext credentials: follow-stream across HTTP, FTP, Telnet, and SMTP

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exporting objects to pull exfiltrated files out of a pcap

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

A forensics-style walkthrough: from odd DNS traffic to a remote shell

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Reporting what you found and what it means

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Part VI: Automation and Extension

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Extracting Fields and Automating Analysis with tshark

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

The one-liner

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

-T fields, -e, and -E: building exactly the rows you need

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Summarizing traffic with shell pipelines

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Pulling fields from HTTP, DNS, and TLS conversations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Looping over many files and scheduling jobs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Why tshark fails in cron

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

PATH

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Capture privilege

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Home and configuration

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Output

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Determinism

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

A canned daily traffic report you can ship today

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Driving Captures and Analysis with Python: pyshark

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Installing pyshark

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Reading pcap and pcapng files

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Layers and fields

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Live capture

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Scheduling pyshark captures with cron

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Working example: flag suspicious DNS queries

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Writing Lua Dissectors for Custom Protocols

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

When you need a dissector

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Dissector anatomy

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Parsing the header

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Subtrees, columns, and expert annotations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Port-based registration versus heuristic dissection

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Testing against a real capture

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Appendix A: Display Filter and Capture Filter Quick Reference

The Two Filter Families

Wireshark has two filter languages with different jobs. Note which box you are in: **capture filters** (BPF syntax) run before packets are stored and **delete** what they reject; **display filters** run after capture and only hide packets from view [29, 160]. Capture filters cannot be changed while a capture runs; display filters can [160].

	Capture filter	Display filter
Language	BPF (libpcap/tcpdump syntax) [75, 78]	Wireshark's display-filter language [492]
Tests	Raw packet bytes in the kernel, before user space [78]	Fields in the decoded protocol tree [28]
Applies	While capturing; permanent; rejected packets are gone [29]	Anytime, reversible; applied to files too [160]
In Wireshark	Filter box on the Welcome screen, or Capture > Options [29]	Filter toolbar, colored green/red/yellow [78, 29]
In tshark	-f "filter" [72, 493, 75]	-Y "filter" [89, 492, 160]
On a saved file	No: operates on live bytes only [75]	Yes, with -r file -Y ... [160]
Cost	Single pass, kernel-level, faster [89, 75]	Per-packet dissection; risky on busy live links because you can lose packets [89]

The display filter can also express things a capture filter cannot, because its keep/drop decision can use knowledge from other packets: for example `icmp.resp_not_found` needs a multipass analysis, which a BPF filter has no access to [75].

Display Filter Language

A bare protocol or field name is an **existence test**: `dns` matches any packet containing DNS, and the implicit “exists” test has the highest priority of any operator [492, 114, 111].

Operators

English	C-like	Meaning
<code>eq</code>	<code>==</code>	Equal; any instance must match (<code>any_eq</code>) [111, 113, 114]
<code>ne</code>	<code>!=</code>	Not equal; all instances must differ (<code>all_ne</code>) [113]
	<code>===</code>	Equal; all instances must match (<code>all_eq</code>) [111, 113]
	<code>!==</code>	Not equal; any instance may differ (<code>any_ne</code>) [111, 113]
<code>gt / lt</code>	<code>> / <</code>	Greater / less than [113, 114]
<code>ge / le</code>	<code>>= / <=</code>	Greater-or-equal / less-or-equal [113, 114]
<code>contains</code>		Field or slice contains a string or byte sequence; not usable on atomic fields such as numbers or IPs [492, 114, 111]
<code>matches</code>	<code>~</code>	PCRE2 regular expression; case-insensitive by default, (<code>?-i</code>) forces case [114, 111]
<code>in {...}</code>		Set membership, with ranges: <code>tcp.port in {80, 443, 4430..4434}</code> [113]

English	C-like	Meaning
&	bitwise_and	Bit mask test on integer fields and slices: tcp.flags & 0x02 matches SYN [111, 122]

Logical operators, highest to lowest precedence: not (!), then and (&&), then xor (^), then or (||), all evaluated left to right; group with parentheses [111, 19]. The any/all modifiers (any ip.addr != 1.1.1.1, all tcp.port > 1024) override the equality operators [111, 114].

Slices take a sub-range of a string or byte field: [i:j] is offset+length, [i-j] is inclusive, [i] one byte, [:j] and [i:] from the ends, negative offsets count from the end (frame[-4:] == 0.1.2.3) [114, 492]. Useful functions: upper(), lower(), len(), count(), string(), max(), min(), abs() [114, 492].

Field value types

Type	Syntax	Example
Integer	Decimal, octal (0), hex (0x), binary (0b) [111, 113]	frame.len > 0x5dc [113]
Boolean	1/true or 0/false [111, 113]	tcp.flags.syn == 1 [113]
IPv4	Dotted decimal, hostname, or CIDR [111, 113]	ip.addr == 10.0.0.0/24 [113]
MAC / bytes	Hex digits split by :, ., or - [111, 113]	eth.dst == ff:ff:ff:ff:ff:ff [113]
String	Double quotes; \x and octal escapes [114, 494]	http.request.method == "POST" [114]
Date/time	Quoted string compared with >=, <= [495]	frame.time >= "2025-01-22 10:00:00" [495]

A Boolean **field is present whether its value is 0 or 1**. tcp.flags.syn alone matches every TCP packet that carries the flag, set or clear. Test tcp.flags.syn == 1 instead [113]. Since Wireshark 3.6, ip.addr !=

10.0.0.1 means “no address equals 10.0.0.1”, so it hides every packet to or from that host [113]. Before 3.6 it meant “some address differs” and let those packets through, which is why older guides tell you to write `not ip.addr eq 10.0.0.1` [122, 114]. Both forms now hide that host’s packets. They still differ on frames with no IP layer, such as ARP: `!=` hides those too, and `not ip.addr eq 10.0.0.1` keeps them. And because every comparison carries an implicit existence test, `ip.dst ne 224.1.2.3` hides every non-IP frame. Use `not ip.dst eq 224.1.2.3` to keep them [111, 114].

Ready-Made Display Filters

Task	Display filter
One host, either direction	<code>ip.addr == 10.0.0.5</code> [113]
Source or destination TCP port	<code>tcp.port == 443</code> [115, 31]
Source port only	<code>tcp.srcport == 68</code> [115]
Whole subnet	<code>ip.addr == 10.0.0.0/24</code> [113]
New connection (SYN)	<code>tcp.flags.syn == 1 && tcp.flags.ack == 0</code> [205, 28]
SYN-ACK with no completed handshake	<code>tcp.flags == 0x12 && !tcp.analysis.initial_rtt</code> [202]
Any retransmission	<code>tcp.analysis.retransmission or tcp.analysis.fast_retransmission</code> [138]
Duplicate ACKs	<code>tcp.analysis.duplicate_ack</code> [138]
All TCP analysis flags	<code>tcp.analysis.flags</code> [222, 138]
DNS failures	<code>dns.flags.rcode != 0</code> [496, 171]
HTTP requests	<code>http.request.method == "GET"</code> [144, 114]

Task	Display filter
TLS ClientHello / SNI	tls.handshake.type == 1 [31, 29] / tls.handshake.extensions_ server_name contains "example.com" [31]
HTTP/2 frame on a stream	http2.streamid == 1 [497, 89]
One whole TCP conversation	tcp.stream == 5 [375, 138]
Any expert note	expert [225]
Text anywhere in the frame	frame contains "password" [440]
Hide keep-alives	!tcp.analysis.keep_alive
Hide pure ACKs (no payload)	!(tcp.flags.ack == 1 && tcp.len == 0) [495]

Analysis fields such as `tcp.analysis.*` are inferences Wireshark draws from sequence numbers, not header values. Treat them as hints, and check the packet's `tcp.seq` against the stall point before blaming the network [138, 205].

Capture Filters (BPF)

A BPF expression is one or more **primitives**: an id preceded by qualifiers of three kinds [77, 76]:

Kind	Qualifiers
Type	host, net, port, portrange [77, 76, 272]
Direction	src, dst, src or dst, src and dst [77, 76]
Protocol	ether, ip, ip6, arp, rarp, tcp, udp (fddi, tr, wlan alias ether) [76, 77]

Defaults: host if no type, src or dst if no direction, and all protocols consistent with the type. So port 53 means (tcp or udp) port 53 [77, 76]. Combine primitives with and, or, not, and parentheses [77, 28].

Intent	Capture filter
One host	host 192.168.1.1 [78]
Source host only	src host 10.0.0.5 [78]
Subnet	net 10.0.0.0/24 [78]
TCP service	tcp port 443 [78, 28]
Well-known ports	portrange 1-1024 [78]
ICMP only	icmp [78]
ARP for one MAC	arp and ether host 00:11:22:33:44:55 [28]
Size bounds	less 100 / greater 1000 [272]
Broadcast/multicast	ether broadcast, ip multicast [272]
Byte arithmetic	ip and (not ip[1] & 0xfc == 0x0) [69]

Keep the capture filter broad: a habit like `not port 22` preserves evidence for later display filtering, and rejected packets cannot be recovered [29]. Two limits matter: `port 443` is a **port number**, not HTTPS, and includes UDP 443 [28]; and a host filter will not collect the client’s ARP resolution, because ARP frames carry no IP header. Collect ARP, DNS, and ICMP explicitly if adjacency evidence matters [28].

Converting Between Families

Intent	Capture filter	Display filter
One host	host 10.10.10.10	ip.addr == 10.10.10.10
Source host	src host 10.10.10.10	ip.src == 10.10.10.10
TCP service	tcp port 443	tcp.port == 443
Port on either protocol	port 53	tcp.port == 53 or udp.port == 53
Subnet	net 10.0.0.0/24	ip.addr == 10.0.0.0/24
ICMP	icmp	icmp

Sources: capture-side syntax [78, 77, 28]; display-side equivalents [113, 115, 498]. The pairs match for IP traffic; BPF speaks bytes, so the capture side cannot express decoded-tree conditions such as `tcp.analysis.*`, and it sees only what the capture point delivers [75, 28].

```
1 tshark -i eth0 -f "tcp port 443" -w https.pcapng # capture filter
2 tshark -r https.pcapng -Y "tls.handshake.type == 1" # display filter
```

Source: [29, 159]. Pass the whole filter as one quoted argument; in tshark, a filter given as trailing arguments is a capture filter when capturing and a display filter when reading with `-r` [159, 89].



The classic failure is mixing the languages: `ip.addr == 10.0.0.1` in the capture box errors, and `host 10.0.0.1` in the display bar turns red [29]. The filter bar tells you: green is valid, red is invalid, yellow is valid but comes with a warning, which the status bar spells out [78].

Appendix B: Command-Line Toolkit

Reference

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

The Tools and Their Jobs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Common Behaviors

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

tshark: Capture and Read

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

dumpcap

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

capinfos

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

editcap

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

mergcap

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Companion Tools

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

File Locations and Automation Notes

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Appendix C: Where to Find Practice Captures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Generate Your Own Traffic

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Official Wireshark Sample Captures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Training Collections with Answer Keys

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Malware and Intrusion Traffic

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

CTF and Competition Captures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Broad and Benchmark Captures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Finding Captures by Search

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Verify Before You Analyze

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Appendix D: Lab Captures and Answers to the Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 1: Your First Capture: Installing Wireshark and Building Your Workspace

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 2: Where to Put the Sniffer: Hubs, SPAN, TAPs, Virtual Switches, and Remote Capture

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 3: Capturing the Right Traffic and Saving It Well: Capture Filters, pcapng, dumpcap, and Ring Buffers

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 4: Managing Capture Files at Scale: capinfos, editcap, and mergecap

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 5: The Display Filter Language: Syntax, Operators, and the Field Reference

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 6: Making Traffic Stand Out: Coloring Rules, Following Streams, and Time References

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 7: tshark: The Terminal Wireshark

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 8: Getting Online: ARP, DHCP, and DNS

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 9: TCP Under the Microscope: Handshakes, Retransmissions, and Sequence Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 10: Modern Web Protocols: HTTP/2 and HTTP/3

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 11: Voice and Video: SIP, RTP, and VoIP Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 12: Decrypting TLS and QUIC with Session Keys

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 13: Capturing Wi-Fi in Monitor Mode

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 14: Finding the Bottleneck: Expert Info, I/O Graphs, and TCP Stream Graphs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 15: Detecting Reconnaissance and Port Scans

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 16: Malware Beacons and Call-Home Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 17: Suspicious DNS and Credential Leaks in Real Traces

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 18: Extracting Fields and Automating Analysis with tshark

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 19: Driving Captures and Analysis with Python: pyshark

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Chapter 20: Writing Lua Dissectors for Custom Protocols

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

Glossary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.

References

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/practical-wireshark>.