



Wireshark Cheat Sheet 2026

Quick Reference Guide by PassITExams

The essential capture and display filters every network engineer and cybersecurity analyst needs at their fingertips. Print it, pin it, and start analyzing packets faster.

Capture Filters vs Display Filters

Capture Filters (BPF syntax)

- Applied **before** capture begins
- Set in Capture Options / interface dialog
- Reduces file size & memory usage
- Cannot be changed once capture starts
- Syntax: `host, port, net, tcp`

Display Filters (Wireshark syntax)

- Applied **after** packets are captured
- Typed into the top filter bar
- Can be changed anytime, non-destructive
- Supports rich logical & field expressions
- Syntax: `ip.addr, tcp.port, http`

Essential Capture Filters

FILTER	DESCRIPTION
<code>host 192.168.1.1</code>	Capture traffic to/from a specific host
<code>net 192.168.1.0/24</code>	Capture traffic on a subnet/network
<code>port 80</code>	Capture traffic on a specific port
<code>portrange 1-1024</code>	Capture a range of ports
<code>tcp</code>	Capture only TCP traffic
<code>udp</code>	Capture only UDP traffic
<code>src host 10.0.0.5</code>	Capture traffic from a source host only
<code>dst host 10.0.0.5</code>	Capture traffic to a destination host only
<code>ether host aa:bb:cc:dd:ee:ff</code>	Capture traffic for a specific MAC address
<code>not broadcast and not multicast</code>	Exclude broadcast/multicast noise
<code>host 10.0.0.1 and port 443</code>	Combine host and port conditions
<code>vlan 100</code>	Capture traffic on a specific VLAN tag

Most Useful Display Filters

IP Address & Subnet

FILTER	DESCRIPTION
<code>ip.addr == 10.0.0.1</code>	Traffic to/from an IP
<code>ip.src == 10.0.0.1</code>	Traffic from source IP
<code>ip.dst == 10.0.0.1</code>	Traffic to destination IP
<code>ip.addr == 10.0.0.0/24</code>	Traffic within a subnet
<code>!(ip.addr == 10.0.0.1)</code>	Exclude an IP address
<code>ip.dst == 224.0.0.0/4</code>	Multicast destination traffic

Port-Based Filters

FILTER	DESCRIPTION
<code>tcp.port == 443</code>	TCP traffic on port 443
<code>udp.port == 53</code>	UDP traffic on port 53 (DNS)
<code>tcp.srcport == 80</code>	TCP traffic from source port 80
<code>tcp.dstport == 22</code>	TCP traffic to destination port 22
<code>tcp.port in {80,443,8080}</code>	Match a set of ports
<code>!(tcp.port == 3389)</code>	Exclude a specific port

Tip: Combine capture filters at collection time to keep files small, then refine with display filters for deep analysis — the two work together, not against each other.



Protocol Filters

FILTER	DESCRIPTION
arp	Show all ARP traffic
dns	Show all DNS queries/responses
icmp	Show ICMP (ping) traffic
dhcp	Show DHCP traffic (BOOTP)
ftp	Show FTP control traffic
smtp pop imap	Show common email protocols
ssh	Show SSH handshake / traffic
tls.handshake.type == 1	Show TLS Client Hello packets

TCP Flags & Analysis

FILTER	DESCRIPTION
tcp.flags.syn == 1	SYN packets (connection start)
tcp.flags.syn==1 && tcp.flags.ack==0	SYN only — new connection attempts
tcp.flags.reset == 1	RST packets — reset connections
tcp.flags.fin == 1	FIN packets — graceful close
tcp.analysis.retransmission	TCP retransmissions
tcp.analysis.duplicate_ack	Duplicate ACKs (possible loss)
tcp.analysis.zero_window	Zero window — receiver buffer full
tcp.analysis.lost_segment	Detected lost TCP segments

HTTP / HTTPS Specific

FILTER	DESCRIPTION
http	Show all HTTP traffic
http.request	Show only HTTP requests
http.response.code == 404	Show 404 Not Found responses
http.request.method == "POST"	Show HTTP POST requests
http contains "password"	Search HTTP payload for text
tls	Show all TLS/SSL traffic
tls.record.content_type == 22	TLS handshake records only
http.host == "example.com"	Requests to a specific host header

Advanced & Troubleshooting Filters

FILTER	DESCRIPTION
frame contains "error"	Search raw frame bytes for text
tcp.window_size == 0	Sender buffer full — flow control issue
expert.severity == error	Show packets Wireshark flags as errors
tcp.time_delta > 1	Packets with response time over 1s
ip.ttl < 10	Low TTL — possible traceroute/loop
icmp.type == 3	ICMP Destination Unreachable
tcp.stream eq 5	Isolate one TCP conversation by stream ID
not arp and not dns	Hide noisy ARP & DNS chatter

Essential Keyboard Shortcuts

Windows / Linux

SHORTCUT	ACTION
Ctrl+E	Start / stop capture
Ctrl+F	Find packet
Ctrl+G	Go to packet number
Ctrl+R	Restart current capture
Ctrl+W	Close capture file
Ctrl+Alt+Shift+T	Follow TCP stream
Ctrl+M	Mark / unmark packet

macOS

SHORTCUT	ACTION
⌘+E	Start / stop capture
⌘+F	Find packet
⌘+G	Go to packet number
⌘+R	Restart current capture
⌘+W	Close capture file
⌘+⌘+⇧+T	Follow TCP stream
⌘+M	Mark / unmark packet



Useful tshark Commands

COMMAND	DESCRIPTION
<code>tshark -D</code>	List available capture interfaces
<code>tshark -i eth0</code>	Capture live traffic on interface eth0
<code>tshark -i eth0 -w out.pcap</code>	Capture and save to a pcap file
<code>tshark -r file.pcap</code>	Read and print packets from a file
<code>tshark -r file.pcap -Y "http"</code>	Apply a display filter while reading
<code>tshark -i eth0 -f "port 443"</code>	Apply a capture filter (BPF) live
<code>tshark -r file.pcap -T fields -e ip.src -e ip.dst</code>	Export specific fields as columns
<code>tshark -r file.pcap -q -z io,stat,1</code>	Show I/O statistics in 1-second intervals
<code>tshark -r file.pcap -z conv,tcp</code>	Summarize TCP conversations
<code>tshark -c 100 -i eth0</code>	Capture a fixed number of packets (100)

Real-World Filter Examples

1. Find slow web responses

```
http.time > 2
```

Isolates HTTP transactions that took longer than 2 seconds to complete — great for spotting slow backend services.

2. Detect a possible port scan

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

A high volume of SYN-only packets from one source to many ports is a classic port scan signature.

3. Troubleshoot DNS failures

```
dns.flags.rcode != 0
```

Shows DNS responses with a non-zero return code, i.e. failed lookups (NXDOMAIN, SERVFAIL, etc.).

4. Isolate one user's conversation

```
ip.addr==10.0.0.15 && tcp.port==443
```

Narrows a busy capture down to a single workstation's HTTPS traffic for focused analysis.

5. Spot packet loss on a link

```
tcp.analysis.retransmission
```

Frequent retransmissions along a path point to congestion, a bad cable, or an overloaded device.

6. Catch credentials sent in the clear

```
http.request && (http contains "pass" || http contains "pwd")
```

A quick security check for plaintext login forms that should be using HTTPS instead of HTTP.

Quick Pro Tips

- 1 Use `Statistics > Conversations` to quickly find the top talkers in any capture before writing filters by hand.
- 2 Save frequently-used filters as Filter Buttons (Analyze > Display Filter Buttons) for one-click access.
- 3 Colorize traffic with custom coloring rules so errors, retransmissions, and specific hosts stand out visually.
- 4 Always apply capture filters on high-traffic links to avoid dropped packets and oversized files.
- 5 Use `Follow > TCP Stream` to read an entire conversation in plain, readable order.
- 6 Export filtered results via File > Export Specified Packets to share a clean, focused pcap with colleagues.

Full interactive version + more guides available at passitexams.com

Master Wireshark, networking, and cybersecurity certifications with PassITExams.

Updated: August 2026