CORVASEC

Wireshark Display Filter Reference

A practical display-filter reference for protocol isolation, hosts, ports, TCP analysis, DNS, HTTP, TLS, Windows protocols, and troubleshooting workflows.

DOCUMENT CENTER

Compiled September 2026

NOTE: Display filters hide or show packets after capture; they do not reduce what was captured. Capture filters use a different syntax. Field names vary by dissector and Wireshark version; validate a filter in the installed version.

Use this guide as a practical reference. Verify commands, targets, permissions, paths, versions, and organizational policy before making changes. Examples with placeholders must be adapted to the environment.

Contents

#	Section	Entries
1	Display filter quick reference	162
2	Filter construction patterns	10

Total reference entries: 172

1. Display filter quick reference

Command	Purpose / Notes
`tcp`	Show TCP packets.
`udp`	Show UDP packets.
`icmp`	Show ICMP packets.
`icmpv6`	Show ICMPv6 packets.
`arp`	Show ARP packets.
`dns`	Show DNS packets.
`http`	Show HTTP packets.
`http2`	Show HTTP/2 packets.
`tls`	Show TLS packets.
`dhcp || bootp`	Show DHCP/BOOTP traffic.
`ntp`	Show NTP packets.
`ssh`	Show traffic dissected as SSH.
`smb2`	Show SMB2/SMB3 packets.
`kerberos`	Show Kerberos packets.
`ldap`	Show LDAP packets.
`rdp`	Show traffic dissected as RDP where available.
`ip.addr == 10.0.0.5`	Packets where source or destination IPv4 address matches.
`ip.src == 10.0.0.5`	Packets sourced by an IPv4 address.
`ip.dst == 10.0.0.5`	Packets destined to an IPv4 address.
`ipv6.addr == 2001:db8::1`	Packets involving an IPv6 address.
`eth.addr == aa:bb:cc:dd:ee:ff`	Packets involving a MAC address.
`tcp.port == 443`	TCP source or destination port 443.
`udp.port == 53`	UDP source or destination port 53.
`tcp.srcport == 443`	TCP source port 443.
`tcp.dstport == 443`	TCP destination port 443.
`tcp.flags.syn == 1 && tcp.flags.ack == 0`	Initial TCP SYN packets.
`tcp.flags.reset == 1`	TCP resets.
`tcp.analysis.retransmission`	TCP retransmissions identified by analysis.
`tcp.analysis.fast_retransmission`	Fast retransmissions.
`tcp.analysis.duplicate_ack`	Duplicate ACKs.
`tcp.analysis.lost_segment`	Segments Wireshark considers missing.
`tcp.analysis.out_of_order`	Out-of-order TCP segments.
`tcp.window_size_value == 0`	Zero-window advertisements.
`tcp.analysis.zero_window`	TCP zero-window analysis events.
`tcp.len > 0`	TCP segments carrying payload.
`tcp.stream eq 0`	Packets in TCP stream 0.
`udp.length > 512`	UDP datagrams larger than 512 bytes.
`dns.flags.response == 0`	DNS queries.
`dns.flags.response == 1`	DNS responses.
`dns.qry.name == "example.com"`	DNS queries for an exact name.
`dns.qry.name contains "example"`	DNS query names containing text.

Command	Purpose / Notes
`dns.flags.rcode != 0`	DNS responses with non-zero response code.
`dns.flags.rcode == 3`	DNS NXDOMAIN responses.
`dns.a`	Packets containing DNS A records.
`dns.aaaa`	Packets containing DNS AAAA records.
`http.request`	HTTP requests.
`http.response`	HTTP responses.
`http.request.method == "GET"`	HTTP GET requests.
`http.request.method == "POST"`	HTTP POST requests.
`http.response.code >= 400`	HTTP error responses.
`http.host == "example.com"`	HTTP requests for a Host value.
`http.request.uri contains "login"`	HTTP request URIs containing login.
`tls.handshake`	TLS handshake packets.
`tls.handshake.type == 1`	TLS ClientHello messages.
`tls.handshake.type == 2`	TLS ServerHello messages.
`tls.handshake.type == 11`	TLS certificate messages where visible/dissected.
`tls.record.version`	TLS records with a version field.
`frame.len > 1500`	Frames larger than 1500 bytes.
`frame.time_delta > 1`	Packets with more than one second since previous frame.
`frame.number == 100`	Display one frame number.
`frame contains "password"`	Frames containing byte sequence text; use carefully because encrypted payloads will not match.
`ip.ttl < 5`	IPv4 packets with very low TTL.
`ip.flags.df == 1`	IPv4 packets with Dont Fragment set.
`ip.flags.mf == 1`	IPv4 packets with More Fragments set.
`ip.frag_offset > 0`	Non-initial IPv4 fragments.
`icmp.type == 8`	ICMP echo requests.
`icmp.type == 0`	ICMP echo replies.
`icmp.type == 3`	ICMP destination unreachable.
`icmp.type == 11`	ICMP time exceeded.
`arp.opcode == 1`	ARP requests.
`arp.opcode == 2`	ARP replies.
`arp.duplicate-address-detected`	ARP duplicate-address detection events when field is generated.
`dhcp.option.dhcp == 1`	DHCP Discover.
`dhcp.option.dhcp == 2`	DHCP Offer.
`dhcp.option.dhcp == 3`	DHCP Request.
`dhcp.option.dhcp == 5`	DHCP ACK.
`ntp.flags.mode == 3`	NTP client mode.
`ntp.flags.mode == 4`	NTP server mode.
`smb2.cmd == 5`	SMB2 CREATE operations.
`smb2.cmd == 8`	SMB2 READ operations.
`smb2.cmd == 9`	SMB2 WRITE operations.
`kerberos.msg_type == 10`	Kerberos AS-REQ.
`kerberos.msg_type == 11`	Kerberos AS-REP.
`kerberos.msg_type == 12`	Kerberos TGS-REQ.

Command	Purpose / Notes
`kerberos.msg_type == 13`	Kerberos TGS-REP.
`ldap.protocolOp`	LDAP protocol operations.
`tcp.port in {80 443 8080 8443}`	TCP traffic on a set of common web ports.
`ip.addr == 10.0.0.5 && tcp.port == 443`	HTTPS-related TCP traffic involving a host.
`dns && ip.src == 10.0.0.5`	DNS traffic sourced by a host.
`tcp.analysis.retransmission || tcp.analysis.fast_retransmission`	Show retransmission analysis events.
`tcp.flags.syn == 1 && tcp.flags.ack == 1`	TCP SYN/ACK packets.
`tcp.flags.fin == 1`	TCP FIN packets.
`tcp.flags.ack == 1 && tcp.len == 0`	Pure ACKs and other ACK-only segments.
`!(arp || ipv6 || icmp)`	Exclude selected common background protocols.
`tcp && !tcp.analysis.retransmission`	TCP excluding retransmissions.
`ip.addr == 10.0.0.5 && !(tcp.port == 22)`	Traffic involving host except SSH.
`tcp.port >= 49152`	TCP traffic using at least one high/ephemeral port.
`udp.port >= 49152`	UDP traffic using at least one high/ephemeral port.
`tcp.time_delta > 1`	TCP packets with stream-relative time delta over one second where field applies.
`tcp.analysis.ack_rtt > 0.2`	TCP ACK RTT over 200 ms where analysis field applies.
`tcp.window_size < 1024`	Small calculated TCP window sizes.
`dns.time > 0.5`	DNS responses taking over 500 ms where dns.time is available.
`http.time > 1`	HTTP response time over one second where http.time is available.
`tls.alert_message`	TLS alert messages.
`tls.record.content_type == 21`	TLS Alert records.
`tcp.port == 3389`	RDP-related TCP port traffic.
`tcp.port == 445`	SMB-related TCP port traffic.
`tcp.port == 22`	SSH-related TCP port traffic.
`udp.port == 123`	NTP-related UDP traffic.
`udp.port == 161`	SNMP query traffic.
`udp.port == 162`	SNMP trap traffic.
`tcp.port == 5985 || tcp.port == 5986`	WinRM traffic.
`tcp.port == 389 || tcp.port == 636`	LDAP/LDAPS traffic.
`tcp.port == 3268 || tcp.port == 3269`	Active Directory Global Catalog traffic.
`tcp.port == 88 || udp.port == 88`	Kerberos traffic.
`tcp.port == 53 || udp.port == 53`	DNS over TCP or UDP.
`tcp.port == 25 || tcp.port == 465 || tcp.port == 587`	Common SMTP/submission ports.
`tcp.port == 143 || tcp.port == 993`	IMAP/IMAPS traffic.
`tcp.port == 110 || tcp.port == 995`	POP3/POP3S traffic.
`tcp.port == 1433`	Microsoft SQL Server traffic on common port.
`tcp.port == 3306`	MySQL traffic on common port.
`tcp.port == 5432`	PostgreSQL traffic on common port.
`tcp.port == 6379`	Redis traffic on common port.
`tcp.port == 9200`	Elasticsearch HTTP API common-port traffic.
`udp.port == 5353`	mDNS traffic.
`udp.port == 5355`	LLMNR traffic.

Command	Purpose / Notes
`udp.port == 1900`	SSDP traffic.
`tcp.port == 502`	Modbus/TCP common-port traffic.
`udp.port == 47808`	BACnet/IP common-port traffic.
`tcp.port == 44818 || udp.port == 44818`	EtherNet/IP common-port traffic.
`eth.dst == ff:ff:ff:ff:ff:ff`	Ethernet broadcasts.
`eth.dst[0] & 1`	Ethernet multicast/broadcast destination bit set.
`ip.dst == 255.255.255.255`	Limited IPv4 broadcast.
`ipv6.dst == ff02::1`	IPv6 all-nodes multicast.
`tcp.flags.syn == 1 && tcp.flags.ack == 0 && tcp.window_size_value == 0`	Initial SYNs advertising zero window; unusual and context dependent.
`tcp.analysis.spurious_retransmission`	Packets Wireshark flags as spurious retransmissions.
`tcp.analysis.keep_alive`	TCP keep-alive analysis packets.
`tcp.analysis.keep_alive_ack`	TCP keep-alive ACK analysis packets.
`tcp.analysis.window_update`	TCP window update analysis events.
`tcp.analysis.window_full`	TCP window full analysis events.
`tcp.analysis.bytes_in_flight > 1000000`	Streams with more than 1 MB in flight where analysis field is available.
`frame.time_relative < 10`	Packets in first ten seconds of capture.
`frame.time_relative >= 60 && frame.time_relative < 120`	Packets between one and two minutes.
`tcp.stream eq 5 && http`	HTTP traffic in a selected TCP stream.
`udp.stream eq 3`	Packets in a selected UDP stream where stream index exists.
`dns.qry.type == 1`	DNS A queries.
`dns.qry.type == 28`	DNS AAAA queries.
`dns.qry.type == 15`	DNS MX queries.
`dns.qry.type == 16`	DNS TXT queries.
`dns.qry.type == 33`	DNS SRV queries.
`dns.qry.type == 12`	DNS PTR queries.
`dns.flags.truncated == 1`	Truncated DNS messages.
`http.user_agent`	HTTP messages containing User-Agent field.
`http.authorization`	HTTP messages with Authorization header when visible.
`http.cookie`	HTTP messages with Cookie header when visible.
`http.set_cookie`	HTTP responses setting cookies when visible.
`tls.handshake.extensions_server_name`	TLS handshake messages containing SNI extension.
`x509af.validity_not_after`	Packets containing parsed certificate expiration field where available.
`tcp.analysis.reused_ports`	TCP 4-tuples/ports Wireshark flags as reused.
`frame.interface_id == 0`	Packets from capture interface 0 in multi-interface captures.
`tcp.checksum.status == 2`	TCP packets marked with bad checksum status; offload can create apparent false positives.
`udp.checksum.status == 2`	UDP packets marked with bad checksum status; offload can create apparent false positives.

2. Filter construction patterns

Pattern	Example	Use
`Protocol presence`	dns	Display packets containing a protocol.
`Field presence`	http.request	Display packets where a field exists.
`Equality`	tcp.port == 443	Match exact numeric/string values.
`Comparison`	frame.len > 1500	Compare numeric fields.
`Boolean AND`	ip.addr == 10.0.0.5 && tcp	Require both conditions.
`Boolean OR`	tcp.port == 80 || tcp.port == 443	Accept either condition.
`Negation`	!arp	Exclude matching packets.
`Set membership`	tcp.port in {80 443 8080}	Match any value in a set.
`Contains`	dns.qry.name contains "example"	Match substring in a supported field.
`Stream focus`	tcp.stream eq 0	Focus on one reconstructed stream.

Authoritative references

Use vendor and standards documentation for exact syntax, current behavior, version-specific details, licensing, and policy requirements.

Wireshark display filter manual

https://www.wireshark.org/docs/man-pages/wireshark-filter.html

Wireshark User Guide

https://www.wireshark.org/docs/wsug_html/