

Wireshark

Wireshark???
?GNU???

- [????](#)
- [Learning](#)
- [tcpdump](#)

????

????

?????

```
==  
!=  
>  
<  
>=  
<=  
in
```

?????

```
&& # AND  
||  # OR  
!   # NOT
```

?????

- type: host, port
- dir: src, dst
- proto: tcp, udp, ftp, http

SIP ??

tcpdump

```
timeout 6m tcpdump -i eth0 host <sip-trunk-ip> -n -s 0 -vvvv -w carrier.pcap
```

Wireshark

- ??? sip ??? filter ??? <https://www.wireshark.org/docs/dfref/s/sip.html>

??? REGISTER ??Filter: `sip.CSeq.method == REGISTER`

??

??????

```
ssh root@192.168.0.1 tcpdump -n -i any -w- 'not \( port 22 and host 192.168.0.1 \)' |etherape -r-
```

Filter ???

Wireshark Display Filter Cheat Sheet

www.cellstream.com www.netscionline.com

Operators and Logic

eq or ==	lt or <	and or && Logical AND	not or ! Logical NOT
ne or !=	ge or >=	or or Logical OR	[n] [] Substring operator
gt or >	le or <=	xor or ^^ Logical XOR	

LAYER 1

frame	frame.ignored	frame.number	frame.time_delta
frame.cap_len	frame.len	frame.p2p_dir	frame.time_delta_displayed
frame.coloring_rule.name	frame.link_nr	frame.protocols	frame.time_epoch
frame.coloring_rule.string	frame.marked	frame.ref_time	frame.time_invalid
frame.file_off	frame.md5_hash	frame.time	frame.time_relative

LAYER 2

Ethernet

eth.addr	eth.multicast
eth.dst	eth.src
eth.ig	eth.trailer
eth.len	eth.type
eth.lg	

ARP

arp.dst.hw_mac	arp.proto.size
arp.dst.proto_ipv4	arp.proto.type
arp.hw.size	arp.src.hw_mac
arp.hw.type	arp.src.proto_ipv4
arp.opcode	

802.1Q VLAN

vlan.cfi	vlan.len
vlan.etype	vlan.priority
vlan.id	vlan.trailer

PPP

ppp.address	ppp.direction
ppp.control	ppp.protocol

VLAN Trunking Protocol

vtp.code	vtp.version
vtp.conf_rev_num	vtp.vlan_info.802_10_index
vtp.followers	vtp.vlan_info.isl_vlan_id
vtp.md	vtp.vlan_info.len
vtp.md5_digest	vtp.vlan_info.mtu_size
vtp.md_len	vtp.vlan_info.status.vlan_susp
vtp.neighbor	vtp.vlan_info.tlv_len
vtp.seq_num	vtp.vlan_info.tlv_type
vtp.start_value	vtp.vlan_info.vlan_name
vtp.upd_id	vtp.vlan_info.vlan_name_len
vtp.upd_ts	vtp.vlan_info.vlan_type

DTP

dtp.neighbor	dtp.tlv_type
dtp.tlv_len	dtp.version

MPLS

mpls.bottom	mpls.oam.defect_location
mpls.cw.control	mpls.oam.defect_type
mpls.cw.res	mpls.oam.frequency
mpls.exp	mpls.oam.function_type
mpls.label	mpls.oam.ttsi
mpls.aom.bip16	mpls.ttl

Frame Relay

fr.becn	fr.control.p	fr.dlci	fr.snap.oui
fr.chdlctype	fr.control.s_ftype	fr.dlcore_control	fr.snap.pid
fr.control	fr.control.u_modifier_cmd	fr.ea	fr.snaptype
fr.control_f	fr.control.u_modifier_resp	fr.fecr	fr.third_dlci
fr.control.ftype	fr.cr	fr.lower_dlci	fr.upper_dlci
fr.control.n_r	fr.dc	fr.nlpid	
fr.control.n_s	fr.de	fr.second_dlci	

LAYER 3

IP v4

ip.addr	ip.fragment.overlap.conflict
ip.checksum	ip.fragments
ip.checksum_bad	ip.fragment.toolongfragment
ip.checksum_good	ip.hdr_len
ip.dsfield	ip.host

IP v6

ipv6.addr	ipv6.hop_opt
ipv6.class	ipv6.host
ipv6.dst	ipv6.mipv6_home_address
ipv6.dst_host	ipv6.mipv6_length
ipv6.dst_opt	ipv6.mipv6_type

Wireshark

- Frame number from the beginning of the packet capture — No.
- Seconds from the first frame — Time
- Source address, commonly an IPv4, IPv6 or Ethernet address — Source (src)
- Destination address — Destination (dst)
- Protocol used in the Ethernet frame, IP packet, or TCP segment — Protocol
- Length of the frame in bytes — Length

Packet Columns

Logical Operators

- and or && — Logical AND — All the conditions should match
- or or || — Logical OR — Either all or one of the conditions should match
- xor or ^ — Logical XOR — Exclusive alterations - only one of the two conditions should match not both
- not or ! — Not (Negation) — Not equal to
- [n] [...] — Substring operator — Filter a specific word or text

Packet Filter

- ip.dest == 192.168.1.1 — Equal — eq or ==
- ip.dest != 192.168.1.1 — Not equal — ne or !=
- frame.len > 10 — Greater than — gt or >
- frame.len < 10 — less than — lt or <
- frame.len >= 10 — Greater than or equal — ge or >=
- frame.len <= 10 — Less than or equal — le or <=

Filter Types

- Capture Filter — Filter packets during capture
- Display Filter — Hide packets from a capture display

Capturing Modes

- Sets interface to capture all packets on a network segment to which it is associated to — Promiscuous mode
- Setup the wireless interface to capture all traffic it can receive (Unix/Linux only) — Monitor Mode

Miscellaneous

- Slice Operator — [...] Range of values
- Membership Operator — {} - In
- CTRL + E — Start/Stop Capturing

Capture Filter Syntax

tcp src 192.168.1.1 80 and tcp dst 202.164.30.1

Display Filter Syntax

http dest dest == 192.168.1.1 and tcp port

Keyboard Shortcuts

- Move to the next packet in the selection history. — Atl + → or option + →
- In the packet detail, opens the selected tree item. — →
- In the packet detail, open the selected tree items and all of its subtrees. — Shift + →
- In the packet detail, opens all tree items. — Ctrl + →
- In the packet detail, close all the tree items. — Ctrl + ←
- In the packet detail, jumps to the parent node. — Backspace
- In the packet detail, toggles the selected tree item. — Return or Enter

- Tab or Shift + Tab — Move between screen elements, e.g. from the toolbars to the packet list to the packet detail.
- ↓ — Move to the next packet or detail item.
- ↑ — Move to the previous packet or detail item.
- Ctrl + ↓ or F8 — Move to the next packet, even if the packet list isn't found
- Ctrl + ↑ or F7 — Move to the previous packet, even if the packet list isn't found
- Ctrl + . — Move to next packet of the conversation (TCP, UDP or IP).
- Ctrl + , — Move to the previous packet of the conversation (TCP, UDP or IP).

Protocols

- lat
- sca
- moprc
- mopdl
- tcp
- udp
- ether
- fdci
- ip
- arp
- rarp
- decnet

Common Filtering Commands

- http.host == "host name" — Filter by URL
- frame.time >= "Feb 02, 2023 18:10:00" — Filter by time stamp
- tcp.flags.syn == 1 and tcp.flags.ack == 0 — Filter SYN Flag
- wlan.fc.type_subtype == 0x08 — Wireshark Beacon Filter
- eth.dst == ffffffff — Wireshark Broadcast Filter
- (eth.dst[0] & 1) — Wireshark Multicast Filter
- ip.host = hostname — Host name Filter
- eth.addr == 00:80:d3:24:52:c4 — MAC address Filter
- tcp.flag.reset == 1 — RST flag filter

- Wireshark Filter by IP — ip.addr == 192.168.0.2
- Filter by Destination IP — ip.dest == 192.168.0.2
- Filter by Source IP — ip.src == 192.168.0.2
- Filter by IP range — ip.addr >= 192.168.0.2 and ip.addr <= 192.168.0.200
- Filter by Multiple IPs — ip.addr == 192.168.0.2 and ip.addr == 192.168.0.20
- Filter out IP address — !ip.addr == 192.168.0.2
- Filter subnet — ip.addr == 192.168.0.2/24
- Filter by port — tcp.port == 25
- Filter by destination port — tcp.dstport == 23
- Filter by ip address and port — ip.addr == 192.168.0.2 and Tcp.port == 25

Main Toolbar Items

- Jump forward in the packet history — Go → Go forward — Go Forward
- Go to specific packet — Go → Go to Packet... — Go to Packet...
- Jump to first packet of the capture file — Go → Go to First Packet — Go to First Packet
- Jump to last packet of the capture file — Go → Go to Last Packet — Go to Last Packet
- Auto scroll packet list during live capture — View → Auto Scroll in Live Capture — Auto Scroll in Live Capture
- Colorize the packet list (or not) — View → Colorize — Colorize
- Zoom into the packet data (increase the font size) — View → Zoom In — Zoom In
- Zoom out of the packet data (decrease the font size) — View → Zoom Out — Zoom Out
- Set zoom level back to 100% — View → Normal Size — Normal Size
- Resize columns, so the content fits the width — View → Resize Columns — Resize Columns

- Start — Capture → Start — Uses the same packet capturing options as the previous session, or uses defaults if no options were set
- Stop — Capture → Stop — Stops currently active capture
- Restart — Capture → Restart — Restart active capture session
- Options... — Capture → Options... — Opens "Capture Options" dialog box
- Open... — File → Open... — Opens "File Open" dialog box to load a capture for viewing
- Save as... — File → Save as... — Save current capture file
- Close — File → Close — Close current capture file
- Reload — File → Reload — Reload current capture file
- Find Packet... — Edit → Find Packet... — Find packet based on different criteria
- Go Back — Go → Go back — Jump back in the packet history



@hackingarticles



<https://in.linkedin.com/company/hackingarticles>



<https://github.com/ignitetechnologies>

For Cybersecurity



Useful Wireshark Filters



- ✓ `ip.addr == 10.0.0.1` Show all traffic with 10.0.0.1 as either source or destination
- ✓ `ip.addr == 10.0.0.0/24` Show all traffic to and from any address in 10.0.0.0/24
- ✓ `ip.src == 10.0.0.1 && ip.dst == 10.0.0.2` Show all traffic from 10.0.0.1 to 10.0.0.2
- ✓ `!(ip.addr == 10.0.0.1)` Exclude all traffic to or from 10.0.0.1
- ✓ `icmp.type == 3` Show ICMP "destination unreachable" packets
- ✓ `tcp or udp` Show TCP or UDP traffic
- ✓ `tcp.port == 80` Show TCP traffic with port 80
- ✓ `tcp.srcport < 1000` Show TCP traffic with src port range
- ✓ `http or dns` Show all HTTP or DNS traffic
- ✓ `tcp.flags.syn == 1` Show TCP packets with SYN flag set
- ✓ `tcp.flags == 0x012` Show TCP packets with both SYN and ACK flags set
- ✓ `tcp.analysis.retransmission` Show all retransmitted TCP packets
- ✓ `http.request.method == "GET"` Show TCP packets associated with HTTP GET
- ✓ `http.response.code == 404` Show packets associated with HTTP 404 response
- ✓ `http.host == "www.test.com"` Show HTTP traffic matching the Host header field
- ✓ `tls.handshake` Show only TLS handshake packets
- ✓ `tls.handshake.type == 1` Show client Hello packet during TLS handshake
- ✓ `dhcp and ip.addr == 10.0.0.0/24` Show DHCP traffic for 10.0.0.0/24 subnet
- ✓ `dhcp.hw.mac_addr == 00:11:22:33:44:55` Show DHCP packets for client MAC addr
- ✓ `dns.resp.name == cnn.com` Show DNS responses with name field of "cnn.com"
- ✓ `frame contains keyword` Show all packets that contain the word "keyword"
- ✓ `frame.len > 1000` Show all packets with total length larger than 1000 bytes
- ✓ `eth.addr == 00:11:22:33:44:55` Show all traffic to or from the specified MAC addr
- ✓ `eth[0x47:2] == 01:80` Match Ethernet frames with 2 bytes at offset 0x47 == 01:80
- ✓ `!(arp or icmp or stp)` Filter out background traffic from ARP, ICMP and STP
- ✓ `vlan.id == 100` Show packets with VLAN ID 100



Created by
@dan_nanni
on Instagram

Learning

- [Network Forensics, Wireshark Basics, Part 1](#)
- [10 Tips On How to Use Wireshark to Analyze Network Packets \(tecmint.com\)](#)

tcpdump

List the interfaces

```
sudo tcpdump -D
```

Capture All traffic

```
tcpdump -i eth0  
tcpdump -i wlan0
```

To a File

```
tcpdump -i eth0 -w capture.pcap  
tcpdump -i any -w capture.pcap -nn 'ip and port 80'  
  
# Set Timeout  
timeout 6m tcpdump -i eth0 -w capture.pcap
```

Read a file (.pcap)

- `-nn` : Disable port and protocol name lookup.
- `-r` : Read capture data from the named file.
- `-v` : Display detailed packet data.
- `-X` : Display the hexadecimal and ASCII output format packet data. Security analysts can analyze hexadecimal and ASCII output to detect patterns or anomalies during malware analysis or forensic analysis.

```
tcpdump -r capture.pcap  
tcpdump -r capture.pcap -nn -v 'ip and (port 80 or port 443)'  
tcpdump -nn -r capture.pcap -X
```

Filter

```
# Filter by Source IP  
tcpdump src 192.168.0.1
```



```
# Filter by Destination IP
```

```
tcpdump dst 192.168.0.1
```

```
# Filter by Port
```

```
tcpdump port 80
```

```
# Filter by Protocol
```

```
tcpdump icmp
```

```
# Protocol and Port
```

```
tcpdump tcp port 443
```

```
# Source and Destination
```

```
tcpdump src 192.168.0.1 and dst 192.168.0.2
```

```
tcpdump -i any -w capture.pcap -n 'ip and port 80'
```

Display in ASCII

```
# Display in ASCII
```

```
tcpdump -A
```

```
# Display in Hexadecimal
```

```
tcpdump -X
```

Specific Number of Packets

```
tcpdump -c 100
```

Display

```
# Capture and Display IPv6 Traffic
```

```
tcpdump -6
```

```
# Capture and Display Traffic in Timestamp Format
```

```
tcpdump -tttt
```

SSH Connections

```
# -l: real-time
# -e: including ethernet headers
tcpdump -i eth0 'tcp port 22' -l -e
```

HTTP Request and Response

```
tcpdump -i eth0 -s 0 -A -n 'tcp dst port 80'
```

IP Range and Protocol

```
tcpdump -i eth0 'net 192.168.0.0/24 and (tcp port 22 or icmp)'
```

DNS Traffic

```
tcpdump -i eth0 'udp port 53' -nnvvv
```

FTP Traffic

```
tcpdump -i eth0 -s 0 'tcp port 21'
```

?? DDos ??????????

```
interface=ens1
dumpdir=/home/user/automatic-tcp-dump/
while /bin/true; do
    pkt_old=`grep $interface: /proc/net/dev | cut -d : -f2 | awk '{ print $2 }'`
    sleep 1
    pkt_new=`grep $interface: /proc/net/dev | cut -d : -f2 | awk '{ print $2 }'`
    pkt=$(( $pkt_new - $pkt_old ))
    echo -ne "\r$pkt packets/s\033[0K"
    if [ $pkt -gt 30000 ]; then
        echo -e "\n`date` Under Attack. Capturing Packets..."
        sudo tcpdump -n -i $interface -s0 -c 20000 -w $dumpdir/dump.`date +%Y%m%d-%H%M%S`.pcap
        echo "`date` Packets Captured."
        sleep 300 && pkill -HUP -f /usr/sbin/tcpdump
    else
        sleep 1
    fi
done
```



Tcpdump Command Examples

- ✓ **tcpdump** listen on the first non-loopback interface detected
- ✓ **tcpdump -i eth0** capture packets on eth0 and display their content
- ✓ **tcpdump -i eth0 -w my.pcap** save packets received on eth0 to my.pcap
- ✓ **tcpdump -i any** capture packets from all available interfaces
- ✓ **tcpdump arp|tcp|udp|icmp** capture only a specific protocol
- ✓ **tcpdump src 10.0.0.1** capture traffic from 10.0.0.1
- ✓ **tcpdump port 80** capture traffic with ether src/dst port 80
- ✓ **tcpdump dst net 10.1.1.0/24** capture traffic for specific subnet
- ✓ **tcpdump tcp and src 10.0.0.1 and port 80** combine multiple filters
- ✓ **tcpdump tcp dst portrange 22-1023** capture packets with port range
- ✓ **tcpdump -vvv** show protocol-specific info with full verbosity
- ✓ **tcpdump -tt** use UNIX timestamp as packet timestamp format
- ✓ **tcpdump not port 22** capture all traffic except ssh traffic
- ✓ **tcpdump -c 1000** capture the first 1000 packets only
- ✓ **tcpdump -n** do not convert IP addresses/ports to names
- ✓ **tcpdump -e** display layer-2 info such as MAC addresses
- ✓ **tcpdump -X** show payload content in hex/ASCII format
- ✓ **tcpdump ip6** capture IPv6 packets only
- ✓ **tcpdump 'tcp port 80 or udp port 67'** use complex filters
- ✓ **tcpdump greater 200** capture packets whose length > 200
- ✓ **tcpdump ether dst ff:ff:ff:ff:ff:ff** capture layer-2 broadcast packets
- ✓ **tcpdump 'tcp[tcpflags] == tcp-syn'** capture TCP SYN packets
- ✓ **tcpdump 'tcp[tcpflags] & (tcp-syn|tcp-fin) != 0'** match TCP SYN or FIN
- ✓ **tcpdump -e vlan 10** capture traffic with VLAN tag 10
- ✓ **tcpdump 'icmp[0] = 8'** capture ICMP echo request packets (ping)
- ✓ **tcpdump outbound** capture only outbound traffic



Created by
Dan Nanni
study-notes.org