tshark Packet analyzer CLI

START HERE

`tshark -D`
List interfaces

`tshark -i <iface> -c 20`
Capture 20 packets

`tshark -i <iface> -w <file.pcapng>`
Capture to a file

`tshark -r <file>`
Read a capture

`tshark -r <file> -Y "<filter>"`
Display filter

`tshark -r <file> -T fields -e <field>`
Extract fields

`tshark -r <file> -q -z conv,tcp`
Conversation stats

`tshark -r <file> -q -z follow,tcp,ascii,0`
Follow a stream

Capture -i, -f filter → `-w` **pcapng file** raw packets → `-r` **Display filter** -Y → `-T` / `-z` **Output** text, fields, JSON, stats

CONCEPTS

Only capture on networks and hosts you are authorised to monitor. Packet data can contain passwords and personal data; store and share captures with care.

Two filter languages: -f is a capture filter (BPF, same as tcpdump, applied while capturing); -Y is a display filter (Wireshark fields like ip.addr, applied after dissection).

Live capture usually needs root or the dumpcap capability; a trailing filter argument is a capture filter live, but a display filter with -r.

CAPTURE

`tshark -D`
List capture interfaces

`tshark -i <iface>`
Capture live on an interface

`tshark -i <iface> -c <n>`
Stop after n packets

`tshark -i <iface> -f "<filter>"`
Capture filter (BPF syntax)

`tshark -i any`
Capture on all interfaces
On Linux this uses cooked headers, so link-layer details are lost.

`tshark -i <iface> -p`
No promiscuous mode

`tshark -i <iface> -s <len>`
Limit bytes kept per packet

`tshark -i <iface> -B <MiB>`
Capture buffer size

`tshark -i <iface> -I`
Monitor mode (Wi-Fi)

FILES AND STOPPING

`tshark -i <iface> -w <file>`
Write raw packets to a file

`tshark -r <file>`
Read a capture file

`tshark -i <iface> -a duration:<s>`
Stop after n seconds

`tshark -i <iface> -a filesize:<kB>`
Stop at file size

`tshark -i <iface> -b filesize:<kB> -b files:<n>`
Ring buffer of files
Each -b takes one condition; repeat it for several.

FILES AND STOPPING (cont.)

`tshark -i <iface> -b duration:<s>`
Start a new file every n seconds

`tshark -r <file> -w <out> -Y "<filter>"`
Save a filtered subset

DISPLAY FILTERS

`-Y "ip.addr == <ip>"`
Packets to or from a host

`-Y "ip.src == <ip>"`
Packets from a source

`-Y "tcp.port == <n>"`
TCP port, either side

`-Y "udp.port == <n>"`
UDP port, either side

`-Y "http.request"`
HTTP requests

`-Y "http.request.method == \"POST\""`
HTTP POST requests

`-Y "dns.qry.name contains \"<name>\""`
DNS queries by name

`-Y "tcp.flags.syn == 1 && tcp.flags.ack == 0"`
New TCP connections

`-Y "tcp.analysis.retransmission"`
Retransmitted segments

`-Y "tcp.analysis.flags"`
Any TCP problem flagged

`-Y "tls.handshake.type == 1"`
TLS Client Hello

`-Y "<a> && !(<b>)"`
Combine with and, or, not
Use && || ! or and or not; quote the whole expression.

`-Y "ip.addr == <net>/24"`
Match a whole subnet

`-Y "frame contains \"<text>\""`
Search raw bytes for text

`tshark -G fields`
List all filter field names

OUTPUT

`tshark -r <file> -V`
Full packet detail

`tshark -r <file> -O <proto>`
Detail for one protocol only

`tshark -r <file> -x`
Hex and ASCII dump

`tshark -r <file> -n`
No name resolution

OUTPUT (cont.)

`tshark -r <file> -t ad`
Absolute date and time

`tshark -r <file> -T json`
Dump packets as JSON

`tshark -r <file> -q`
Suppress packet output

`tshark -r <file> -o "<pref>:<value>"`
Override a preference once

`tshark -d tcp.port==<n>,<proto>`
Decode a port as a protocol

`tshark -o tls.keylog_file:<file>`
Decrypt TLS with a key log
The key log comes from the client, e.g. a browser run with SSLKEYLOGFILE set.

EXTRACT FIELDS

`tshark -r <file> -T fields -e <field>`
Print chosen fields only

`-T fields -e ip.src -e ip.dst -e tcp.dstport`
Several fields, tab separated

`-T fields -E header=y`
Add a header row

`-T fields -E separator=,`
Comma separated output

`-T fields -E quote=d`
Quote values with double quotes

`-T fields -E occurrence=f`
Keep first value of repeated fields

`-T fields -e _ws.col.info`
Include the Info column text

`-T fields -e frame.time_epoch`
Packet timestamp in epoch seconds

`-T fields -e http.host -e http.request.uri`
HTTP host and path

`-T fields -e dns.qry.name`
DNS names queried

`-T fields -e tls.handshake.extensions_server_name`
TLS server names (SNI)

STATISTICS

`tshark -r <file> -q -z io,phs`
Protocol hierarchy

`tshark -r <file> -q -z conv,tcp`
TCP conversations

STATISTICS (cont.)

`tshark -r <file> -q -z conv,ip`
IP conversations

`tshark -r <file> -q -z endpoints,ip`
IP endpoints

`tshark -r <file> -q -z io,stat,<s>`
Packets and bytes per interval

`tshark -r <file> -q -z expert`
Expert info summary

`tshark -z help`
List all statistics

STREAMS AND OBJECTS

`tshark -r <file> -q -z follow,tcp,ascii,<n>`
Follow TCP stream n as text

`-z follow,tcp,hex,<n>`
Follow a stream as hex

`-z follow,http,ascii,<n>`
Follow an HTTP stream

`-Y "tcp.stream == <n>"`
All packets of one TCP stream

`tshark -r <file> -T fields -e tcp.stream | sort -nu`
List stream numbers present

`tshark -r <file> --export-objects http,<dir>`
Save files from HTTP traffic

RECIPES

Capture web traffic to a rotating set of files

```
# 5 files of about 10 MB each, oldest overwritten
tshark -i <iface> -f "tcp port 80 or tcp port 443" \
  -b filesize:10000 -b files:5 -w web.pcapng
```

Top talkers from a capture

```
tshark -r <file> -T fields -e ip.src | sort | uniq -c | sort -rn | head
# or with built-in stats
tshark -r <file> -q -z endpoints,ip
```

DNS queries to CSV

```
tshark -r <file> -Y "dns.flags.response == 0" -T fields \
  -E header=y -E separator=, \
  -e frame.time -e ip.src -e dns.qry.name > dns.csv
```

Triage a slow connection

```
tshark -r <file> -q -z expert
tshark -r <file> -Y "tcp.analysis.retransmission" \
  -T fields -e tcp.stream -e ip.dst | sort | uniq -c | sort -rn
```