Wireshark Cheatsheet

Essential operations for network packet analysis and troubleshooting

This cheatsheet provides a quick reference to fundamental Wireshark operations, filters, and analysis techniques, ideal for both beginners and experienced network administrators for efficient packet capture and analysis.

Packet Capture
Capture network traffic from interfaces

Display Filters
Filter and view specific packets

Protocol Analysis
Analyze network protocols in detail

Statistics & Analysis
Generate network traffic statistics

Export & Save
Save and export capture data

Capture Filters & Traffic Capture

Host Filtering

Capture traffic to/from specific hosts.

```
# Capture traffic from/to specific IP
host 192.168.1.100

# Capture traffic from specific source
src host 192.168.1.100

# Capture traffic to specific destination
dst host 192.168.1.100

# Capture traffic from subnet
net 192.168.1.0/24
```

Port Filtering

Capture traffic on specific ports.

```
# HTTP traffic (port 80)
port 80

# HTTPS traffic (port 443)
port 443

# SSH traffic (port 22)
port 22

# DNS traffic (port 53)
port 53

# Port range
portrange 1000-2000
```

Protocol Filtering

Capture specific protocol traffic.

```
# TCP traffic only
tcp

# UDP traffic only
udp

# ICMP traffic only
icmp

# ARP traffic only
arp
```

Advanced Capture Filters

Combine multiple conditions for precise capture.

```
# HTTP traffic to/from specific host
host 192.168.1.100 and port 80

# TCP traffic except SSH
tcp and not port 22

# Traffic between two hosts
host 192.168.1.100 and host 192.168.1.200

# HTTP or HTTPS traffic
port 80 or port 443
```

Interface Selection

Choose network interfaces for capture.

```
# List available interfaces
tshark -D

# Capture on specific interface
# Ethernet interface
eth0

# WiFi interface
wlan0

# Loopback interface
lo
```

Capture Options

Configure capture parameters.

```
# Limit capture file size (MB)
-a filesize:100

# Limit capture duration (seconds)
-a duration:300

# Ring buffer with 10 files
-b files:10

# Promiscuous mode (capture all traffic)
-p
```

Display Filters & Packet Analysis

Filter captured packets for detailed analysis.

01

Basic Display Filters

Essential filters for common protocols and traffic types.

```
# Show only HTTP traffic
http

# Show only HTTPS/TLS traffic
tls

# Show only DNS traffic
dns

# Show only TCP traffic
tcp

# Show only UDP traffic
udp

# Show only ICMP traffic
icmp
```

02

IP Address Filtering

Filter packets by source and destination IP addresses.

```
# Traffic from specific IP
ip.src == 192.168.1.100

# Traffic to specific IP
ip.dst == 192.168.1.200

# Traffic between two IPs
ip.addr == 192.168.1.100

# Traffic from subnet
ip.src_net == 192.168.1.0/24

# Exclude specific IP
not ip.addr == 192.168.1.1
```

03

Port & Protocol Filters

Filter by specific ports and protocol details.

```
# Traffic on specific port
tcp.port == 80

# Source port filter
tcp.srcport == 443

# Destination port filter
tcp.dstport == 22

# Port range
tcp.port >= 1000 and tcp.port <=
2000

# Multiple ports
tcp.port in {80 443 8080}
```

Protocol-Specific Analysis

HTTP Analysis

Analyze HTTP requests and responses.

```
# HTTP GET requests
http.request.method == "GET"

# HTTP POST requests
http.request.method == "POST"

# Specific HTTP status codes
http.response.code == 404

# HTTP requests to specific host
http.host == "example.com"

# HTTP requests containing string
http contains "login"
```

DNS Analysis

Examine DNS queries and responses.

```
# DNS queries only
dns.flags.response == 0

# DNS responses only
dns.flags.response == 1

# DNS queries for specific domain
dns.qry.name == "example.com"

# DNS A record queries
dns.qry.type == 1

# DNS errors/failures
dns.flags.rcode != 0
```

TCP Analysis

Analyze TCP connection details.

```
# TCP SYN packets (connection attempts)
tcp.flags.syn == 1

# TCP RST packets (connection resets)
tcp.flags.reset == 1

# TCP retransmissions
tcp.analysis.retransmission

# TCP window size issues
tcp.analysis.window_update

# TCP connection establishment
tcp.flags.syn == 1 and tcp.flags.ack == 0
```

TLS/SSL Analysis

Examine encrypted connection details.

```
# TLS handshake packets
tls.handshake

# TLS certificate information
tls.handshake.certificate

# TLS alerts and errors
tls.alert

# Specific TLS version
tls.handshake.version == 0x0303

# TLS Server Name Indication
tls.handshake.extensions_server_name
```

Network Troubleshooting

Identify common network issues.

```
# ICMP unreachable messages
icmp.type == 3

# ARP requests/responses
arp.opcode == 1 or arp.opcode == 2

# Broadcast traffic
eth.dst == ff:ff:ff:ff:ff:ff

# Fragmented packets
ip.flags.mf == 1

# Large packets (potential MTU issues)
frame.len > 1500
```

Time-Based Filtering

Filter packets by timestamp and timing.

```
# Packets within time range
frame.time >= "2024-01-01 10:00:00"

# Packets from last hour
frame.time_relative >= -3600

# Response time analysis
tcp.time_delta > 1.0

# Inter-arrival time
frame.time_delta > 0.1
```

Statistics & Analysis Tools

Protocol Hierarchy

View protocol distribution in capture.

```
# Access via: Statistics > Protocol Hierarchy
- Shows percentage of each protocol
- Identifies most common protocols
- Useful for traffic overview

# Command line equivalent
tshark -r capture.pcap -q -z io,phs
```

Conversations

Analyze communication between endpoints.

```
# Access via: Statistics > Conversations
- Ethernet conversations
- IPv4/IPv6 conversations
- TCP/UDP conversations
- Shows bytes transferred, packets count

# Command line equivalent
tshark -r capture.pcap -q -z conv,tcp
```

I/O Graphs

Visualize traffic patterns over time.

```
# Access via: Statistics > I/O Graphs
- Traffic volume over time
- Packets per second
- Bytes per second
- Apply filters for specific traffic

# Useful for identifying traffic spikes
```

Expert Information

Identify potential network problems.

```
# Access via: Analyze > Expert Info
- Warnings about network issues
- Errors in packet transmission
- Performance problems
- Security concerns

# Filter by expert info severity
tcp.analysis.flags
```

Flow Graphs

Visualize packet flow between endpoints.

```
# Access via: Statistics > Flow Graph
- Shows packet sequence
- Time-based visualization
- Useful for troubleshooting
- Identifies communication patterns
```

Response Time Analysis

Measure application response times.

```
# HTTP response times
# Statistics > HTTP > Requests

# DNS response times
# Statistics > DNS

# TCP service response time
# Statistics > TCP Stream Graphs > Time Sequence
```

File Operations & Export

Save, load, and export capture data.

Save & Load Captures

Manage capture files in various formats.

```
# Save capture file
File > Save As > capture.pcap

# Load capture file
File > Open > existing.pcap

# Merge multiple capture files
File > Merge > select files

# Save filtered packets only
File > Export Specified Packets
```

Export Options

Export specific data or packet subsets.

```
# Export selected packets
File > Export Specified Packets

# Export packet dissections
File > Export Packet Dissections

# Export objects from HTTP
File > Export Objects > HTTP

# Export SSL/TLS keys
Edit > Preferences > Protocols > TLS
```

Command Line Capture

Use tshark for automated capture and analysis.

```
# Capture to file
tshark -i eth0 -w capture.pcap

# Capture with filter
tshark -i eth0 -f "port 80" -w http.pcap

# Read and display packets
tshark -r capture.pcap

# Apply display filter to file
tshark -r capture.pcap -Y "tcp.port == 80"
```

Batch Processing

Process multiple capture files automatically.

```
# Merge multiple files
mergecap -w merged.pcap file1.pcap file2.pcap

# Split large capture files
editcap -c 1000 large.pcap split.pcap

# Extract time range
editcap -A "2024-01-01 10:00:00" \
    -B "2024-01-01 11:00:00" \
    input.pcap output.pcap
```

Performance & Optimization

Optimize Wireshark for better performance with large captures.

Memory Management

Handle large capture files efficiently.

```
# Use ring buffer for continuous capture
-b filesize:100 -b files:10

# Limit packet capture size
-s 96  # Capture only first 96 bytes

# Use capture filters to reduce data
host 192.168.1.100 and port 80

# Disable protocol dissection for speed
-d tcp.port==80,http
```

Display Optimization

Improve GUI performance with large datasets.

```
# Preferences to adjust:
# Edit > Preferences > Appearance
- Limit recent files list
- Reduce font size if needed

# Edit > Preferences > Protocols
- Disable unnecessary protocol dissectors
- Reduce TCP reassembly

# Use tshark for large file analysis
tshark -r large.pcap -q -z conv,tcp
```

Efficient Analysis Workflow

Best practices for analyzing network traffic.

```
# 1. Start with capture filters
# Capture only relevant traffic

# 2. Use display filters progressively
# Start broad, then narrow down

# 3. Use statistics first
# Get overview before detailed analysis

# 4. Focus on specific flows
# Right-click packet > Follow > TCP Stream
```

Automation & Scripting

Automate common analysis tasks.

```
# Create custom display filter buttons
# View > Display Filter Expression

# Use profiles for different scenarios
# Edit > Configuration Profiles

# Script with tshark
#!/bin/bash
tshark -r $1 -q -z endpoints,tcp | \
grep -v "Filter:" | head -20
```

Installation & Setup

Install and configure Wireshark for your operating system.

Windows Installation

Download and install from official website.

```
# Download from wireshark.org
# Run installer as Administrator
# Include WinPcap/Npcap
during installation

# Command line installation
(chocolatey)
choco install wireshark

# Verify installation
wireshark --version
```

Linux Installation

Install via package manager or from source.

```
# Ubuntu/Debian
sudo apt update
sudo apt install wireshark

# Red Hat/CentOS/Fedora
sudo yum install wireshark
# or
sudo dnf install wireshark

# Add user to wireshark group
sudo usermod -a -G wireshark
$USER
```

macOS Installation

Install using Homebrew or official installer.

```
# Using Homebrew
brew install --cask wireshark

# Download from wireshark.org
# Install .dmg package

# Command line tools
brew install wireshark
```

Configuration & Preferences

Customize Wireshark for optimal workflow and security.

Interface Preferences

Configure capture interfaces and options.

```
# Edit > Preferences > Capture
- Default capture interface
- Promiscuous mode settings
- Buffer size configuration
- Auto-scroll in live capture

# Interface-specific settings
# Capture > Options > Interface Details
```

Protocol Settings

Configure protocol dissectors and decoding.

```
# Edit > Preferences > Protocols
- Enable/disable protocol dissectors
- Configure port assignments
- Set decryption keys (TLS, WEP, etc.)
- TCP reassembly options

# Decode As functionality
# Analyze > Decode As
```

Display Preferences

Customize the user interface and display options.

```
# Edit > Preferences > Appearance
- Color scheme selection
- Font size and type
- Column display options
- Time format settings

# View > Time Display Format
- Seconds since capture start
- Time of day
- UTC time
```

Security Settings

Configure security-related options and decryption.

```
# TLS decryption setup
# Edit > Preferences > Protocols > TLS
- RSA keys list
- Pre-shared keys
- Key log file location

# Disable potentially dangerous features
- Lua scripts execution
- External resolvers
```

Advanced Filtering Techniques

Master complex filtering for detailed packet analysis.

Logical Operators

Combine multiple filter conditions.

```
# AND operator
tcp.port == 80 and ip.src == 192.168.1.100

# OR operator
tcp.port == 80 or tcp.port == 443

# NOT operator
not icmp

# Parentheses for grouping
(tcp.port == 80 or tcp.port == 443) and ip.src ==
192.168.1.0/24
```

String Matching

Search for specific content in packets.

```
# Contains string (case-sensitive)
tcp contains "password"

# Contains string (case-insensitive)
tcp matches "(?i)login"

# Regular expressions
http.request.uri matches "\.php$"

# Byte sequences
eth.src[0:3] == 00:11:22
```

Field Comparisons

Compare packet fields with values and ranges.

```
# Equality
tcp.srcport == 80

# Greater than/less than
frame.len > 1000

# Range checks
tcp.port >= 1024 and tcp.port <= 65535

# Set membership
tcp.port in {80 443 8080 8443}

# Field existence
tcp.options
```

Advanced Packet Analysis

Identify specific packet characteristics and anomalies.

```
# Malformed packets
_ws.malformed

# Duplicate packets
frame.number == tcp.analysis.duplicate_ack_num

# Out of order packets
tcp.analysis.out_of_order

# TCP window scaling issues
tcp.analysis.window_full
```

Common Use Cases

Practical examples for network troubleshooting and security analysis.

Network Troubleshooting

Identify and resolve network connectivity issues.

```
# Find connection timeouts
tcp.analysis.retransmission and tcp.analysis.rto

# Identify slow connections
tcp.time_delta > 1.0

# Find network congestion
tcp.analysis.window_full

# DNS resolution problems
dns.flags.rcode != 0

# MTU discovery issues
icmp.type == 3 and icmp.code == 4
```

Security Analysis

Detect potential security threats and suspicious activity.

```
# Port scanning detection
tcp.flags.syn == 1 and tcp.flags.ack == 0

# Large number of connections from single IP
# Use Statistics > Conversations

# Suspicious DNS queries
dns.qry.name contains "dga" or dns.qry.name matches
"^[a-z]{8,}\.com$"

# HTTP POST to suspicious URLs
http.request.method == "POST" and http.request.uri
contains "/upload"

# Unusual traffic patterns
# Check I/O Graphs for spikes
```

Application Performance

Monitor and analyze application response times.

```
# Web application analysis
http.time > 2.0

# Database connection monitoring
tcp.port == 3306 and tcp.analysis.initial_rtt > 0.1

# File transfer performance
tcp.stream eq X and tcp.analysis.bytes_in_flight

# VoIP quality analysis
rtp.jitter > 30 or rtp.marker == 1
```

Protocol Investigation

Deep dive into specific protocols and their behavior.

```
# Email traffic analysis
tcp.port == 25 or tcp.port == 587 or tcp.port == 993

# FTP file transfers
ftp-data or ftp.request.command == "RETR"

# SMB/CIFS file sharing
smb2 or smb

# DHCP lease analysis
bootp.option.dhcp == 1 or bootp.option.dhcp == 2
```

Reference: This cheatsheet covers essential Wireshark commands and techniques for effective network packet analysis, troubleshooting, and security monitoring in modern network environments.