Tcpdump Essentials: Command-Line Network Traffic Analysis

Capturing, Filtering, and Analyzing Network Packets

Introduction

Understanding network protocols requires visibility into the actual conversations happening beneath polished user interfaces. While we interact with networks daily through seamless experiences, the underlying protocol exchanges remain invisible. Tcpdump provides the window into this hidden world, enabling analysts to capture and examine network traffic at the packet level.

Written in C and C++ for Unix-like systems in the late 1980s or early 1990s, tcpdump remains highly relevant today. Its libpcap library forms the foundation for numerous networking tools and was ported to Windows as winpcap. This stability and optimal performance make tcpdump the go-to tool for command-line packet analysis.

Learning Objectives

This guide covers the essential tcpdump techniques you need:

- Capture packets and save them to files for later analysis
- Apply precise filters to focus on specific traffic
- Control packet display formats for different analysis needs

Core Command-Line Options

While tcpdump can run without arguments, real-world scenarios require specificity about what to capture, where to save it, and how to display results. The following options form the foundation of effective packet capture.

Network Interface Selection

The first decision is choosing which network interface to monitor. Use `-i INTERFACE` to specify a particular interface, or `-i any` to listen on all available interfaces simultaneously.

Identify available interfaces using `ip address show` or `ip a s`. This displays all network cards along with the loopback interface.

Saving Captured Packets

For later analysis, save captured packets to a file using `-w FILE`. The standard extension is .pcap. When using this option, packets won't scroll on screen but are written directly to the file, which can then be opened in Wireshark or other analysis tools.

Reading Saved Captures

Read previously captured packets using `-r FILE`. This proves invaluable for protocol learning, attack analysis, and applying different filters to the same data set without recapturing traffic.

Limiting Capture Count

Control the number of packets captured with `-c COUNT`. Without this option, capture continues until interrupted with CTRL-C. Limiting packet count is useful when you only need a sample of traffic or want to avoid filling disk space.

Disabling Name Resolution

By default, tcpdump resolves IP addresses to hostnames via DNS lookups, which introduces delays. The `-n` flag prevents IP address resolution, while `-nn` disables both IP and port number resolution, showing raw IP addresses and port numbers instead of service names like 'http' for port 80.

Verbose Output Levels

Increase output detail with verbosity flags. The `-v` option prints additional packet details including TTL, identification, total length, and IP options. Use `-vv` for more verbose output, or `-vvv` for maximum verbosity.

Command-Line Options Summary

Command	Explanation
`tcpdump -i INTERFACE`	Captures packets on a specific network interface
`tcpdump -w FILE`	Writes captured packets to a file
`tcpdump -r FILE`	Reads captured packets from a file
`tcpdump -c COUNT`	Captures a specific number of packets
`tcpdump -n`	Disables IP address resolution
`tcpdump -nn`	Disables IP and port number resolution
`tcpdump -v`	Verbose display; can be increased with -vv and -vvv

Practical Examples

- `tcpdump -i eth0 -c 50 -v` - Captures and displays 50 packets on the wired Ethernet interface with verbose output
- `tcpdump -i wlo1 -w data.pcap` - Captures WiFi traffic and writes to data.pcap until interrupted
- `tcpdump -i any -nn` - Captures on all interfaces without DNS or port resolution

Packet Filtering Techniques

Running tcpdump without filters captures everything, which quickly becomes overwhelming. Just as you wouldn't try to follow every conversation in a crowded room simultaneously, effective packet analysis requires focusing on specific traffic patterns. Tcpdump's filtering expressions enable this precision.

Filtering by Host

Limit captures to traffic involving specific hosts using `host IP` or `host HOSTNAME`. This proves essential when monitoring communication with particular servers, devices, or network endpoints.

For more granular control, specify traffic direction:

- `src host IP` - Captures only packets originating from the specified host
- `dst host IP` - Captures only packets destined for the specified host

Filtering by Port

Isolate traffic on specific ports using `port PORT_NUMBER`. This technique is invaluable for analyzing particular services. For example, `port 53` captures all DNS queries and responses, since DNS uses both UDP and TCP port 53.

Directional port filtering works similarly to host filtering:

- `src port PORT_NUMBER` - Filters packets from a specific source port
- `dst port PORT_NUMBER` - Filters packets to a specific destination port

Filtering by Protocol

Focus on specific protocols by name: `ip, ip6, udp, tcp, icmp`, and others. For instance, filtering for ICMP traffic reveals ping commands (echo requests and replies) and traceroute activity (time exceeded messages).

Logical Operators

Combine filters using logical operators for complex queries:

- **and** - Requires both conditions to be true (e.g., `host 1.1.1.1 and tcp`)
- **or** - Requires either condition to be true (e.g., `udp or icmp`)
- **not** - Inverts the condition (e.g., `not tcp` captures all non-TCP traffic)

Filter Commands Summary

Command	Explanation
`host IP or host HOSTNAME`	Filters packets by IP address or hostname
`src host IP`	Filters packets from a specific source host
`dst host IP`	Filters packets to a specific destination host
`port PORT_NUMBER`	Filters packets by port number
`src port PORT_NUMBER`	Filters by specified source port number
`dst port PORT_NUMBER`	Filters by specified destination port
`PROTOCOL (ip, tcp, udp, icmp)`	Filters packets by protocol

Complex Filter Examples

- `tcpdump -i any tcp port 22` - Captures SSH traffic on all interfaces
- `tcpdump -i wlo1 udp port 123` - Monitors Network Time Protocol traffic on WiFi
- `tcpdump -i eth0 host example.com and tcp port 443 -w https.pcap` - Captures HTTPS traffic with example.com

Advanced Filtering

Packet Length Filters

Filter packets based on size to isolate specific traffic patterns:

- `greater LENGTH` - Captures packets with length greater than or equal to specified value
- `less LENGTH` - Captures packets with length less than or equal to specified value

TCP Flag Filtering

Advanced filtering includes examining TCP flags to identify specific connection behaviors. Use `tcp[tcpflags]` to reference the TCP flags field, with available flags including:

- `tcp-syn` - TCP SYN (Synchronize) flag
- `tcp-ack` - TCP ACK (Acknowledge) flag
- `tcp-fin` - TCP FIN (Finish) flag
- `tcp-rst` - TCP RST (Reset) flag
- `tcp-push` - TCP Push flag

TCP Flag Filter Examples

- `tcpdump "tcp[tcpflags] == tcp-syn"` - Captures packets with only SYN flag set
- `tcpdump "tcp[tcpflags] & tcp-syn != 0"` - Captures packets with at least SYN flag set
- `tcpdump "tcp[tcpflags] & (tcp-syn|tcp-ack) != 0"` - Captures packets with SYN or ACK flags set

Display Customization

Tcpdump offers multiple output formats to suit different analysis needs. The following options control how captured packets appear on screen.

Brief Output

The `-q` flag produces quick, concise output showing only essential information: timestamp, source and destination addresses, and port numbers. This works well when you need to scan through large numbers of packets quickly.

Link-Level Headers

Include MAC addresses and Ethernet details with `-e`. This proves valuable when learning protocols like ARP and DHCP, or when tracking the physical source of unusual traffic on your network.

ASCII Format

Display packet contents as ASCII text using `-A`. This format works well for plain-text protocols and unencrypted traffic, mapping bytes to readable English characters, numbers, and symbols.

Hexadecimal Format

For encrypted, compressed, or non-English content, hexadecimal format provides universal readability. The `-xx` option displays each byte as two hexadecimal digits, enabling octet-by-octet inspection of IP and TCP headers along with packet contents.

Combined Hex and ASCII

Get the best of both worlds with `-X`, which displays packets in both hexadecimal and ASCII formats side by side. This comprehensive view facilitates thorough packet analysis.

Display Options Summary

Command	Explanation
`tcpdump -q`	Quick and quiet: brief packet information
`tcpdump -e`	Include MAC addresses and link-level headers
`tcpdump -A`	Print packets as ASCII encoding
`tcpdump -xx`	Display packets in hexadecimal format
`tcpdump -X`	Show packets in both hex and ASCII formats

Key Takeaways

- Tcpdump provides command-line packet capture with optimal performance and stability
- Interface selection, file operations, and count limits form the foundation of packet capture
- Filtering by host, port, and protocol enables focused analysis on specific traffic
- Logical operators combine multiple filter criteria for complex queries
- Advanced filtering includes TCP flag analysis and packet length criteria
- Multiple display formats accommodate different analysis requirements

Conclusion

Tcpdump transforms the invisible world of network protocols into visible, analyzable data. By combining capture options, filtering expressions, and display formats, analysts can precisely target and examine the traffic patterns that matter. Whether troubleshooting connectivity issues, investigating security incidents, or learning protocol behavior, tcpdump provides the foundational toolkit for command-line packet analysis.

This guide synthesizes essential tcpdump techniques into a practical reference. The command-line interface might seem daunting initially, but its power and flexibility make it indispensable for serious network analysis. Master these fundamentals, and you'll have the skills to dissect any network conversation.

Continue building your packet analysis skills through hands-on practice and real-world application.