Capture and Analyze Network Traffic with Wireshark for Diagnostics

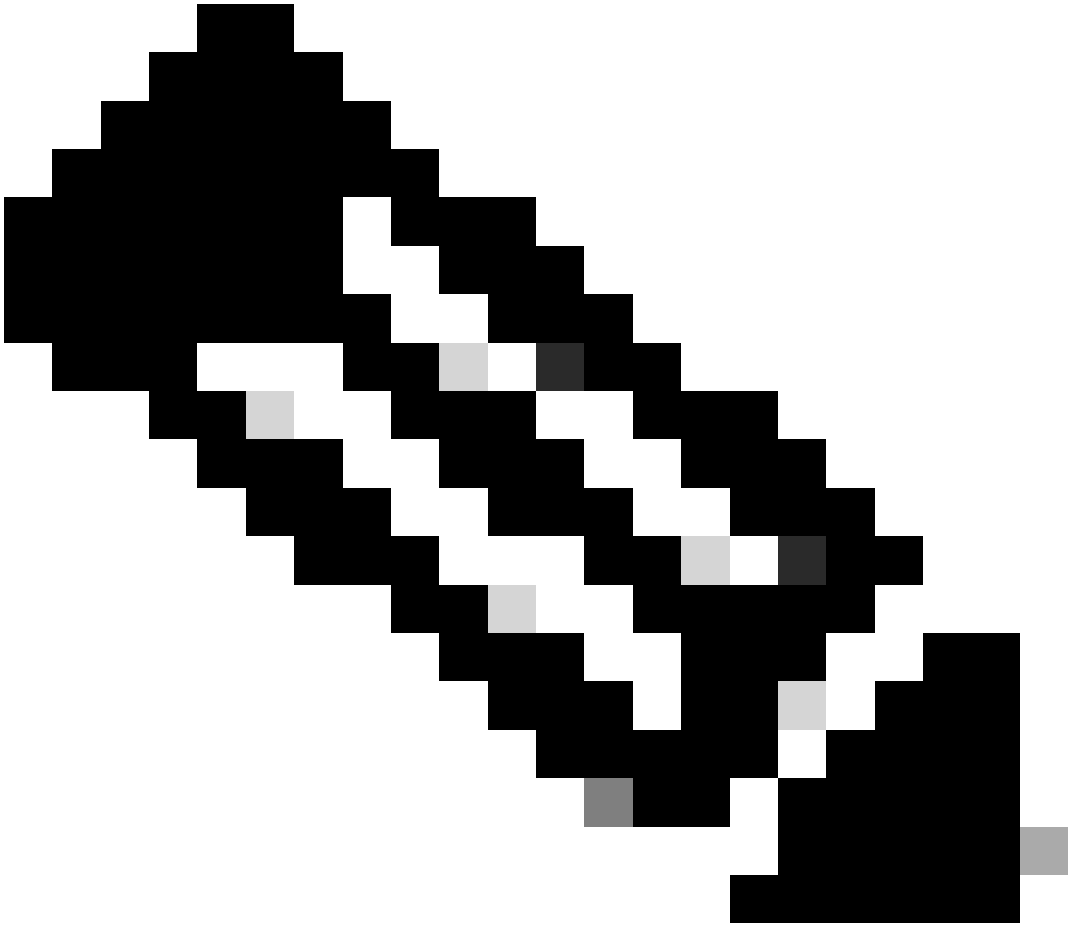
Contents

Introduction

This document describes how to use Wireshark to capture and analyze network traffic for diagnostic purposes.

Overview

Wireshark is a free application you can use to read and analyze packet captures (also called "TCP dumps"). Packet captures reveal all communications through a network adapter at the packet level, making it possible to view DNS, HTTP, ping, and other traffic types. Packet captures are especially valuable as a diagnostic step for deep troubleshooting and, with the introduction of SIG, are now a fundamental part of the diagnostic process.



Note: Wireshark captures all traffic on the selected adapter. Because packet captures often contain personally identifiable information (PII), always use a secure method, such as a Box link, to share capture files with support.

Get Wireshark

You can download Wireshark for Windows, macOS, or Linux at: https://www.wireshark.org/

Gather a Packet Capture

1. Choose the network adapter connected to the internet and start the capture in Wireshark.
2. While capturing, reproduce the issue you want to diagnose.
3. Stop the capture when finished and save the file as a `.pcap`.

Basic Ports and Protocols

- Most packets communicate on the transport layer protocols TCP or UDP

    - For example, "DNS" runs "on top of" UDP by default. It switchs to UDP if TCP fails.
- HTTP and DNS are common protocols that run on a combination of transport protocol + ports.

Transport Layer Protocol	Port	Protocol Name	Usage
TCP	22	SSH	Remote VA Access
TCP	25	SMTP	VA Monitoring
IP	50	ESP (Encapsulating Security Payload)	Confidentiality, data integrity, origin authentication
IP	51	AH (Authentication Header)	Data integrity, origin authentication
UDP	53	DNS	DNS Default
TCP	53	DNS	DNS Fail-Over
TCP	80	HTTP	Web Traffic (unencrypted), APIs
UDP	123	NTP	VA Time Sync
TCP	443	HTTPS	Encrypted web traffic, APIs, AD Connectors to VAs
UDP	443	HTTPS	RC Encrypted DNS queries
UDP	500	IKE	IPsec tunnel negotiations
UDP	4500	NAT-T	NAT traversal for IPsec tunnels
TCP	8080	HTTP	AD connectors to VAs communications

Knowing protocol names, ports, and their uses helps you identify and analyze relevant traffic in Wireshark.

Basic Operators

When building filter strings in Wireshark, use these operators:

- `==`: Equals (Example:`ip.dst==1.2.3.4`)
- `!=`: Not Equal (Example:`ip.dst!=1.2.3.4`)
- `&&`: And (Example:`ip.dst==1.2.3.4 && ip.src==208.67.222.222`)
- `||`: Or (Example:`ip.dst==1.2.3.4 || ip.dst==1.2.3.5`)

For advanced filter options, refer to the Wireshark documentation: [6.4. Building Display Filter Expressions]

Filters

Packet captures can contain thousands of packets. Filters help you focus on specific traffic types:

- By protocol:
    - `dns`— Show only DNS traffic
    - `http || dns`— Show HTTP or DNS traffic
- By IP address:
    - `ip.addr==<IP>`— All traffic to/from`<IP>`
    - `ip.src==<IP>`— All traffic from`<IP>`
    - `ip.dst==<IP>`— All traffic to`<IP>`

- Miscellaneous:
    - `tcp.flags.reset==1`— Check for TCP resets (timeouts)
    - `dns.qry.name contains "[domain]"`— DNS queries matching a domain
    - `tcp.port==80 || udp.port==80`— TCP or UDP traffic on port 80

Viewing and Analyzing Packets

After locating a packet, expand the segments within Wireshark to analyze details. Familiarity with protocol structure helps you interpret these details and even reconstruct data if needed.

Following a Data Stream

Use the packet list to locate request and response pairs. Right-click on a packet and select **Follow > TCP Stream**, **UDP Stream**, **TLS Stream**, or **HTTP Stream** to view the related request and response sequence.

- This is more useful with protocols that have multiple exchanges (for example, HTTP) than with single-request protocols (for example, DNS).