BY DAILY RED TEAM ACADEMY

EXPLORING

WIRESHARK

FOR RED TEAMERS

FROM BASICS TO ADVANCED
NETWORK ANALYSIS TECHNIQUES

Disclaimer: This book is intended for **educational and research purposes only**. Unauthorized use of systems **without explicit permission** is illegal. Always conduct testing in a controlled and legal environment.

TABLE OF CONTENTS

CHAPTER 1 - INTRODUCTION TO WIRESHARK

Wireshark is the world's foremost network protocol analyzer. It is cross-platform, features both GUI and command-line interfaces, and supports various protocols. It is optimized to analyze networks and network traffic and adheres to the paradigm of being a "sniffing" tool. Wireshark captures network traffic from both Ethernet and IEEE 802.11 networks and presents collected data in a multi-colored "packet list" graph powered by exhaustive protocol records. With these features, Wireshark has become straightforward yet complex software capable of handling simple to advanced tasks.

Yet, with its wide range of capabilities and variety of options, Wireshark could prove both daunting and over-encumbering for users who are first exposed to its characteristics resulting from its complex protocol-driven structure. Hence, the objective of this chapter is to provide a comprehensive guide on Wireshark's salient features and uses, notably its graphical user interface and its command-line interface.

The chapter focuses on the usage and emphasizes "unpackaged" tools that underlie and coexist with Wireshark. This guide is divided into four distinct but equally important sections. In the first section, the Wireshark application is highlighted. The basic usage techniques are discussed, including how to acquire packets using real-time network traffic. Decoding the structure of Wi-Fi networks and how to capture them into a file are also documented. These fundamentals will generally allow users to capture without any trouble. Needed to continue efficiently, though, are the tools to set capturing filters for special pre-evacuation of selected traffic. Both GUI and command-line filtering tools are supported. The third part establishes advanced capturing operations obtained directly from the software through parameters utilized by the main experiments for network protocol content changes. With respect to network protocols, the final part provides in-depth coverage of a variety of protocols from Wireshark's required knowledge.

WHAT IS WIRESHARK?

Wireshark is a powerful and user-friendly tool for analyzing networks. Typically, network analysis is performed by reviewing the traffic either through a switch’s or a hub’s mirrored port, or by capturing traffic using a TAP or SPAN port to receive a copy of all traffic that is going into or going out of a specific port or multiple ports. As a packet analyzer, Wireshark is the right tool for providing all preliminary information on the contents of traffic running on your network.

Wireshark also has a set of advanced features that can provide additional analysis when needed. It provides a simple way to create filters and also has basic statistics features and the ability to perform packet modifications. In addition, Wireshark can be complemented with other tools to help with more detailed statistical analysis or with specific information that is not shared by packet inspection tools.

The community that supports Wireshark is very active. This allows the developers to come up with new, updated versions that can work with a wide range of communication protocols, as well as generating up-to-date protocol libraries. The community can also help when troubleshooting particular events on the network by sharing the captured packets and then analyzing them together when specific protocol oddities are observed or specific needs arise.

WHY USE WIRESHARK?

Wireshark offers a wide set of features. In addition to being popular, some strengths of Wireshark are:

- **Free:** As Wireshark remains one of the best free tools for network analysis, it is frequently recommended and used by network administrators, ISPs, governments, or home network administrators to help them understand network problems and the impact of changes.
- **Cross-platform support:** Wireshark is highly compatible with most types of operating system environments, including Windows, Linux, and macOS.
- **Well-documented:** Wireshark has built a large group of followers over time, all contributing to community-generated documentation. With numerous contributions, users can easily access comprehensive instructions for essential information they may need.
- **Pull-down menus and color coding:** Wireshark includes a built-in toolset that supports deep inspection of hundreds of protocols so that users can view and analyze thousands of packets. Moreover, it also contains powerful display filters to help users easily and quickly locate specific network traffic.
- **Support for a huge number of protocols:** Wireshark provides advanced detection capabilities for hundreds of different types of packets or protocols, covering not only commonly used protocols such as TCP, UDP, HTTP, or DNS but also a wide array of protocol family members.

Initially created as a solo project for an open-source initiative, the Wireshark codebase has also been contributed to by various individuals over the years. Because Wireshark is widely used by private and professional organizations at various scales, this tool has been improved in recent years, upgraded with enhanced decoding, and fortified with security scanning functions to cater to network analysts' stringent security and compliance demands. With support for a diversity of operating system platforms, detailed documentation, an active development community, and an astonishing spectrum of protocol analysis support, a user can hardly find any fault with using Wireshark as a tool for network analysis.

Wireshark is a powerful, open-source network protocol analyzer that enables users to capture and inspect network traffic in real-time or from saved capture files. It is designed to assist network administrators, security professionals, and developers in understanding and troubleshooting network communications, making it a versatile tool for various technical roles.

Key Features and Functionalities

1. **Packet Capture:** Wireshark can capture live network traffic from a variety of interfaces, including Ethernet, Wi-Fi, and Bluetooth. This functionality allows users to monitor data as it traverses the network, providing a real-time view of communications.
2. **Protocol Decoding:** It supports an extensive range of protocols, such as TCP, UDP, HTTP, and SMB, and can decode them to display human-readable information. This feature is crucial for analyzing the content and structure of network packets.
3. **Filtering:** Users can apply both display filters and capture filters to focus on specific types of traffic. For instance, a capture filter like tcp port 80 can be used to monitor HTTP traffic, enhancing efficiency in analysis.

4. Statistics and Analysis: Wireshark provides tools for analyzing traffic patterns, performance metrics, and protocol distributions, which are essential for identifying anomalies or performance bottlenecks.

Wireshark is cross-platform, supporting Windows, macOS, and Linux, ensuring accessibility for a wide range of users. Its open-source nature, as of March 1, 2025, allows for community contributions and customizations, further enhancing its utility.

RELEVANCE TO RED TEAMERS

Red teamers are security professionals engaged in offensive security operations, simulating cyberattacks to identify vulnerabilities and test an organization's defensive capabilities. Wireshark is highly relevant to their work for several specific reasons, each contributing to their ability to plan and execute effective attacks.

1. Understanding Network Protocols: By studying network traffic captured with Wireshark, red teamers can gain a deep understanding of how different protocols function. This knowledge is crucial for identifying potential vulnerabilities and crafting exploits. For example, understanding SMB protocol behavior can aid in credential extraction techniques, as discussed in later chapters.
2. Identifying Attack Vectors: Analyzing network traffic helps red teamers recognize potential entry points and weaknesses in the network's security posture. They can identify misconfigured services, unencrypted communications, or outdated protocols that can be exploited.
3. Analyzing Malicious Traffic: Wireshark can be used to understand the behavior of malware or other malicious activities, aiding in the development of detection evasion strategies. For instance, red teamers can analyze command-and-control (C2) traffic to ensure their implants are communicating effectively without detection.
4. Verifying Attack Success: Red teamers can capture and analyze traffic generated by their own attacks to confirm if they are achieving the desired outcomes without being detected by blue team defenses. This is particularly useful in verifying lateral movement or data exfiltration attempts.

An unexpected detail here is that while red teamers typically focus on being covert, Wireshark can also be used in controlled environments, such as during post-exploitation phases, to gather intelligence from compromised systems. This dual use—both for learning and operational analysis—makes Wireshark a versatile tool in their arsenal.

Setting Up Wireshark for Offensive Security

To effectively use Wireshark in red team operations, proper setup and configuration are essential, ensuring the tool is tailored to the specific needs of offensive security activities. The following steps outline the process, with considerations for red team-specific customizations.

Step 1: Installation

1. Download Wireshark: Visit the official Wireshark website (www.wireshark.org) and download the latest version suitable for your operating system. As of March 1, 2025, ensure you select the version compatible with your system's architecture (32-bit or 64-bit).

2. Install Wireshark: Follow the installation instructions provided for your OS. On Windows, this involves running the installer with administrative privileges; on Linux, it may require using package managers like apt or yum. Ensure that you have the necessary permissions to install the software, which is critical for red teamers working in controlled environments.

Step 2: Configuration

1. Capture Interfaces: Upon launching Wireshark, familiarize yourself with the network interfaces available on your system. Select the appropriate interface for capturing traffic, such as eth0 for Ethernet or a Wi-Fi interface. Red teamers may need to capture traffic from compromised systems, which could involve setting up network taps or using tools to forward traffic to their analysis machine.
2. Filters: Learn to use display and capture filters to narrow down the traffic you are interested in. For example, to capture HTTP traffic, use tcp port 80; for SMB, use port 445. This is particularly useful for focusing on specific attack-related traffic, such as credential exchanges or C2 communications.
3. Preferences: Customize Wireshark's preferences to suit your needs. This includes setting default capture settings, enabling or disabling certain protocols for analysis, and adjusting display options for better readability. For instance, enabling the "Reassemble fragmented IPv4 datagrams" option under Edit > Preferences > Protocols > IPv4 can help in analyzing fragmented attack traffic.

Step 3: Customizations for Red Teamers

1. Dissectors: If dealing with custom or proprietary protocols, you might need to create or use custom dissectors to properly decode the traffic. This is particularly relevant when analyzing traffic from bespoke applications used in target environments, enhancing the ability to identify vulnerabilities.
2. Scripting: Use Wireshark's built-in scripting capabilities, such as Lua, or external tools like tshark (the command-line version) to automate tasks. For example, scripting can be used to parse specific data from captures, such as extracting NTLM hashes from SMB traffic, as discussed in subsequent chapters.
3. Security Considerations: Always ensure that you are capturing traffic within the scope of your engagement and in compliance with legal and organizational policies. Red teamers must operate under authorized conditions, typically defined by a rules of engagement document, to avoid legal repercussions. This is crucial, as unauthorized network captures can lead to severe consequences.

Table: Comparison of Wireshark Features for Red and Blue Teams

Feature	Relevance to Red Teamers	Relevance to Blue Teamers
Packet Capture	Used to analyze attack traffic and verify success	Used to detect intrusions and anomalies
Protocol Decoding	Helps identify vulnerabilities for exploitation	Aids in understanding normal traffic patterns
Filtering	Focuses on specific attack-related traffic	Filters out noise to identify threats
Statistics and Analysis	Verifies attack effectiveness and network layout	Monitors network performance and security

This table highlights the dual utility of Wireshark, emphasizing its role in both offensive and defensive security operations, which is an important consideration for red teamers to understand blue team perspectives.

Wireshark is an open-source tool that lets you see and analyze network traffic, like emails or web browsing, by capturing data as it moves through a network. It can decode various protocols, making it easier to understand what's happening, and it works on Windows, macOS, and Linux as of March 1, 2025.

Packet Capturing Fundamentals

Packet capturing is the process of intercepting and storing network packets as they travel across a network. To start, launch Wireshark, choose the network interface (like Wi-Fi), and click "Start" to capture traffic. You can use capture filters, like `tcp port 80` for HTTP traffic, to focus on specific data. Save captures to files for later analysis using formats like *pcapng*.

Filters and Display Options

Once captured, use display filters to show only certain packets, like `http` HTTP traffic, entered in the filter field. Color packets by rules, such as coloring HTTP green, to spot patterns easily. Sort or group packets by columns like timestamp or source IP to organize data.

Decoding Protocols

Wireshark decodes protocols like TCP (for reliable connections), UDP (for faster, connectionless communication), and HTTP (for web traffic). Select a packet, expand the protocol layer in the details pane, and see readable data, like HTTP requests or responses. It also supports DNS, SMTP, and SSL/TLS, with options to update dissectors for custom protocols.

This chapter, "Wireshark Basics: Understanding Network Traffic," is the second in the book "Exploring Wireshark for Red Teamers: From Basics to Advanced Network Analysis Techniques." It aims to provide red teamers, who are security professionals simulating adversarial activities to test organizational defenses, with a foundational understanding of using Wireshark for packet capturing, filtering, display options, and protocol decoding. Given the current date, March 1, 2025, all information is aligned with the latest available resources and practices.

PACKET CAPTURING FUNDAMENTALS

Packet capturing is the process of intercepting and storing network packets as they travel across a network, forming the basis of network traffic analysis with Wireshark. This section details the steps and considerations for effective packet capture, which is particularly relevant for red teamers in offensive security operations.

STARTING A CAPTURE

To begin capturing network traffic with Wireshark:

1. Launch Wireshark: Open the Wireshark application, ensuring it is the latest version as of March 1, 2025, available at Wireshark Official Website.
2. Select Interface: Choose the network interface from which you want to capture traffic. This could be an Ethernet card (`eth0`), Wi-Fi interface, or any other network device. Red teamers may need to capture traffic from compromised systems, which could involve setting up network taps or using tools to forward traffic to their analysis machine.
3. Start Capture: Click the "Start" button to begin capturing packets. By default, Wireshark captures all traffic on the selected interface, which can generate a large amount of data, especially in busy networks.

An unexpected detail here is that capturing on high-traffic interfaces, such as those in data centers, can lead to performance issues, requiring red teamers to consider hardware capabilities and possibly use tools like *dumpcap* for high-volume captures.

CAPTURE FILTERS

Capture filters allow you to specify which packets to capture based on certain criteria, such as source or destination IP addresses, ports, or protocols. These filters are applied at the kernel level, making them efficient for reducing the amount of data captured and are particularly useful for red teamers focusing on specific attack-related traffic.

To set a capture filter in Wireshark:

1. Go to Capture Options: From the "Capture" menu, select "Options."
2. Set Filter: In the "Capture filter" field, enter your filter expression. For example, to capture only HTTP traffic, use `tcp port 80`; for SMB, use `port 445`. These examples are documented in the Wireshark User's Guide.
3. Start Capture: Click "Start" to begin capturing with the filter applied.

Common capture filter expressions include:

- `host 192.168.1.100` to capture traffic to or from a specific IP address.
- `port 443` to capture traffic on a specific port (HTTPS in this case).
- `tcp` to capture only TCP traffic.

This filtering reduces noise, allowing red teamers to focus on relevant traffic, such as command-and-control communications or credential exchanges.

SAVING CAPTURES

Captured data can be saved to a file for later analysis, which is crucial for red teamers reviewing traffic post-engagement. To save a capture:

1. Stop Capture: Click the "Stop" button to halt the capture.
2. Save File: From the "File" menu, select "Save As" and choose a location and filename for the capture file.

Wireshark supports various file formats, but the default is *pcapng*, which is recommended for its flexibility and compatibility with other tools. This format supports features like name resolution and multiple interfaces, enhancing analysis capabilities.

FILTERS AND DISPLAY OPTIONS

Once traffic is captured, Wireshark provides numerous ways to filter and display the data to make analysis more efficient, which is essential for red teamers to quickly identify patterns or anomalies.

Display Filters

Display filters are used to filter which packets are displayed in the packet list pane after capture. Unlike capture filters, they don't affect which packets are captured; they only show which ones, offering more flexibility for post-capture analysis.

To apply a display filter:

Enter Filter: In the "Filter" field at the top of the main window, type your filter expression. For example, `http` to show all HTTP packets, or `ip.addr == 192.168.1.100` to show packets where the IP address is 192.168.1.100, as detailed in the Wireshark Wiki.

1. Apply Filter: Press Enter or click the "Apply" button.

Display filters are more flexible and can be based on a wider range of packet attributes compared to capture filters, such as TCP flags (`tcp.flags == 0x02` for SYN packets) or specific protocol fields.

COLORING RULES

Wireshark allows you to color packets based on certain criteria, making it easier to identify specific types of traffic at a glance, which is particularly useful for red teamers spotting malicious patterns.

To set coloring rules:

1. Go to Preferences: From the "Edit" menu, select "Preferences."
2. Coloring Rules: Navigate to "Coloring Rules" under "User Interface."
3. Add Rule: Click "New" to create a new rule, specify the filter and color, and click "OK."

For example, you can color all HTTP packets green or all TCP SYN packets red, enhancing visual analysis. This feature can be customized to highlight attack-related traffic, such as coloring all SMB session setup requests for credential extraction analysis.

SORTING AND GROUPING

Packets can be sorted by various columns in the packet list pane, such as timestamp, source IP, or protocol. Grouping can also be applied to organize packets by certain attributes, aiding in pattern recognition.

To sort or group packets:

1. Right-click Column Header: Select "Sort Ascending," "Sort Descending," or "Group By" for the desired column.

This feature is useful for quickly identifying patterns, such as grouping by source IP to see all traffic from a specific compromised host, which is valuable for red teamers analyzing lateral movement.

DECODING PROTOCOLS (TCP, UDP, HTTP, ETC.)

Wireshark supports decoding a vast number of network protocols, allowing you to see the contents of packets in a human-readable format, which is crucial for red teamers to understand and exploit network communications.

TCP AND UDP

- TCP (Transmission Control Protocol): Used for reliable, connection-oriented communication. Wireshark displays TCP flags, sequence numbers, acknowledgment numbers, and other control information. For red teamers, analyzing TCP flags can help identify connection initiation (SYN) or termination (FIN, RST), valid for mapping network topology or detecting firewall rules.
- UDP (User Datagram Protocol): Used for connectionless communication, suitable for applications like DNS or streaming. It shows source and destination ports and the data payload, which can be analyzed for unencrypted data exfiltration.

HTTP

HTTP (Hypertext Transfer Protocol) is used for web traffic, a common target for red teamers. Wireshark can decode HTTP requests and responses, showing methods (GET, POST), URLs, headers, and payloads.

To view HTTP details:

1. Select Packet: Choose a packet with HTTP traffic in the packet list.
2. Expand HTTP Layer: In the packet details pane, expand the "HTTP" section to see request or response information, such as credentials in cleartext POST requests, which is a potential vulnerability for exploitation.

OTHER COMMON PROTOCOLS

- DNS (Domain Name System): Used for name resolution. Wireshark shows query types (A, MX, etc.), responses, and resource records, which red teamers can analyze for domain enumeration or detecting C2 domains.
- SMTP (Simple Mail Transfer Protocol): Used for email transmission. It displays commands (HELO, MAIL FROM, RCPT TO) and responses related to mail delivery, useful for analyzing phishing campaign traffic.
- SSL/TLS (Secure Sockets Layer/Transport Layer Security): Used for encrypted communications. With proper configuration, such as providing session keys, Wireshark can decrypt and display the contents of SSL/TLS sessions, revealing sensitive data like API keys or session tokens, as discussed in later chapters.

PROTOCOL DISSECTORS

Wireshark uses dissectors to decode specific protocols. These are pieces of code that parse the packet data according to the protocol's specification, ensuring accurate analysis.

If a protocol isn't recognized or decoded correctly, you can:

1. Check Preferences: Ensure that the protocol is enabled in Wireshark's preferences under Edit > Preferences > Protocols.
2. Update Wireshark: Make sure you're using the latest version, as new dissectors are added over time, available at Wireshark Official Website.
3. Custom Dissectors: For custom or proprietary protocols, you might need to write or obtain a custom dissector, which is particularly relevant for red teamers analyzing bespoke applications in target environments.

Table: Comparison of TCP and UDP for Red Team Analysis

Feature	TCP Analysis	UDP Analysis
Connection Type	Connection-oriented, reliable	Connectionless, faster
Use Case for Red Team	Analyze session establishment, flags	Monitor unencrypted data, DNS queries
Example Filter	`tcp.flags == 0x02` (SYN packets)	`udp.port == 53` (DNS traffic)
Vulnerability	Session hijacking, RST injection	Data leakage, amplification attacks

This table highlights the differences, aiding red teamers in choosing the appropriate protocol for analysis based on their objectives.

CHAPTER 3 - INTERMEDIATE TECHNIQUES: TRAFFIC ANALYSIS FOR RECONNAISSANCE

Reconnaissance is a critical phase in red team operations, where the team gathers information about the target network to identify potential vulnerabilities and plan their attacks. Traffic analysis using Wireshark plays a pivotal role in this phase by allowing red teamers to intercept and analyze network traffic to extract valuable intelligence. This chapter will delve into techniques such as identifying network devices and services, extracting credentials from cleartext protocols, and mapping network topology via packet analysis, each with practical proofs-of-concept (POCs) for application in engagements.

IDENTIFYING NETWORK DEVICES AND SERVICES

To effectively plan and execute attacks, red teamers need to understand the types of devices and services present in the target network. This information can help identify potential entry points, vulnerable services, and the overall network structure. Wireshark provides several methods to achieve this through traffic analysis.

USING MAC ADDRESSES TO IDENTIFY DEVICE TYPES

Every network interface has a unique MAC address, which includes an Organizationally Unique Identifier (OUI) that can indicate the manufacturer of the device. By analyzing MAC addresses from network traffic, red teamers can identify the types of devices present, such as routers, switches, or endpoints.

POC: Identifying Device Manufacturers from MAC Addresses

1. Capture ARP Traffic: ARP (Address Resolution Protocol) packets contain MAC addresses. Use a capture filter `arp` capture ARP traffic, focusing on traffic that resolves IP addresses to MAC addresses.

2. Extract MAC Addresses: In Wireshark, select an ARP packet and look at the "Source MAC Address" and "Destination MAC Address" fields in the packet details pane. These fields are typically found under the "Ethernet II" layer.

3. Identify Manufacturers: Wireshark has a built-in feature to resolve OUIs to vendor names. To enable this, go to `Edit > Preferences > Name Resolution > MAC Address`, and ensure "Resolve MAC addresses using a local table" is checked. This uses Wireshark's internal manufacturer database, which is compiled from sources like the IEEE Registration Authority.

Alternatively, for manual lookup, use online tools such as Wireshark's OUI Lookup or IEEE OUI and Company IDs to determine the manufacturer based on the MAC address's OUI (the first three bytes, e.g., `00:1A:2F` for Intel).

An unexpected detail here is that while OUIs denote the vendor, they don't always indicate the chipset manufacturer, as some vendors package third-party chipsets registered under their own OUI, adding complexity to device identification.

IDENTIFYING SERVICES VIA PORT NUMBERS

Services typically run on standard port numbers, which can be used to identify them. For example, port 80 is commonly associated with HTTP, port 443 with HTTPS, and port 22 with SSH. By observing which ports are in use, red teamers can infer which services are running on the network.

POC: Listing Services Based on Observed Ports

1. Capture Network Traffic: Capture traffic without any filters to see all active ports, ensuring a comprehensive view of network activity. Use the "Start Capturing" button in Wireshark to begin.
2. Analyze TCP and UDP Port Numbers: In Wireshark, look at the "Destination Port" column for TCP and UDP packets in the packet list pane. Use the "Protocol" column to see which protocol is being used, as Wireshark automatically decodes this information.
3. Map Ports to Services: Use the well-known port numbers to identify services, as listed in resources like IANA Service Name and Transport Protocol Port Number Registry.
    - Port 80: HTTP
    - Port 443: HTTPS
    - Port 22: SSH
    - Port 21: FTP
    - Port 23: Telnet

For non-standard ports, look for patterns or known service behaviors, such as unusual ports for HTTP (e.g., 8080), and verify by inspecting packet contents.

A tip for red teamers: Some services might use non-standard ports due to security policies, so relying solely on port numbers might not always be accurate. Cross-verify with protocol signatures for confirmation.

RECOGNIZING PROTOCOL SIGNATURES

Some protocols have unique signatures or patterns in their traffic that can be identified, such as specific strings or headers. For example, HTTP traffic often starts with "GET" or "POST," while DNS traffic has query and response formats with specific record types.

POC: Identifying Specific Protocols or Applications

1. Capture Traffic: Capture traffic from a known application or protocol using filters like `tcp port 80` for HTTP or `udp port 53` for DNS to focus on specific traffic.
2. Analyze Packet Contents: Look for specific strings or patterns in the packet data. For example, HTTP traffic starts with "GET", "POST", etc., visible in the "HTTP" layer of the packet details pane. DNS traffic shows queries like "A" records or responses with IP addresses under the "DNS" layer.
3. Use Wireshark's Protocol Decoders: Wireshark automatically decodes many protocols, making it easier to identify them. If a packet is decoded as a specific protocol (e.g., "HTTP" or "DNS"), that's likely what it is. Expand the protocol layers in the packet details pane to see readable data.

This method is particularly useful for identifying custom or proprietary protocols that might not follow standard port assignments, enhancing reconnaissance depth.

EXTRACTING CREDENTIALS FROM CLEARTEXT PROTOCOLS

Many protocols transmit data in cleartext, making it possible to capture and read sensitive information like credentials, which is a goldmine for red teamers during reconnaissance. This section covers capturing HTTP, FTP, and Telnet traffic, where credentials are often visible.

CAPTURING HTTP TRAFFIC

HTTP traffic is often unencrypted, especially in internal networks or for certain applications, making it vulnerable to credential extraction. Modern websites increasingly use HTTPS, but misconfigurations or legacy systems might still use HTTP.

POC: Extracting Credentials from HTTP Requests

1. Capture HTTP Traffic

Use a capture filter: `tcp port 80` to focus on HTTP traffic, capturing packets where credentials might be sent in plain text.

2. Identify Login Requests

Look for HTTP POST requests to login pages, identified by URLs like `/login.php`, `/authenticate`, etc., in the packet list. Use a display filter `http.request.method == "POST"` to narrow down to POST requests.

3. Extract Form Data

    - In the packet details, expand the "HTTP" section and look for the "Form Item" fields, which typically include username and password fields. For example, you might see:

```
username: admin
password: P@ssw0rd123
```

    - Sometimes, credentials might be sent in the URL as query parameters in GET requests, though this is less common for sensitive data.

4. Save or Note the Credentials

    - Record the extracted credentials for further use, such as attempting to log in to the target system or escalating access.

An unexpected detail is that while HTTPS encrypts traffic, some internal applications or misconfigured systems might still use HTTP, providing opportunities for credential capture in otherwise secure environments.

CAPTURING FTP TRAFFIC

FTP transmits credentials in cleartext during the login process, making it another target for credential extraction, though modern systems often use encrypted alternatives like SFTP or FTPS.

POC: Extracting Credentials from FTP Login

1. Capture FTP Traffic

Use a capture filter: `tcp port 21` to capture FTP control channel traffic, where login commands are sent.

2. Identify USER and PASS Commands

    - In FTP, the username is sent with the "USER" command and the password with the "PASS" command. Look for packets containing these commands in the packet list, visible under the "FTP" layer.

3. Extract Credentials
    - Look for packets containing "USER" followed by the username and "PASS" followed by the password. For example:

```
USER admin
PASS P@ssw0rd123
```

4. Note the Credentials
    - Record the username and password for further exploitation, such as logging into the FTP server.

A tip for red teamers: Ensure capture is done during active FTP sessions, as credentials are only visible during login, and modern networks might block or encrypt FTP traffic.

CAPTURING TELNET TRAFFIC

Telnet also sends credentials in cleartext; though it's primarily deprecated due to security concerns, it might still be found in legacy systems.

POC: Extracting Credentials from Telnet Login

1. Capture Telnet Traffic

Use a capture filter: `tcp port 23` to capture Telnet traffic, where login prompts and responses are sent.

2. Identify Login Sequence
    - Telnet login typically involves the server prompting for "login:" and "password:", followed by the user input. This might require assembling multiple packets to get the complete input, using "Follow TCP Stream" in Wireshark to see the conversation.
3. Extract Credentials
    - Look for the user's response to the login prompt and the password prompt in the stream, which is visible as plain text under the "Data" section of the packet details.
4. Note the Credentials
    - Record the username and password for further use, noting that Telnet is rare in modern networks but might be present in isolated legacy environments.

MAPPING NETWORK TOPOLOGY VIA PACKET ANALYSIS

By analyzing network traffic, red teamers can map out the network's structure, including IP addresses, subnets, and connections between devices, providing a blueprint for further exploitation.

DETERMINING IP ADDRESS RANGES AND SUBNETS

Identifying the range of IP addresses used can help understand the network's structure, including subnet ranges and potential segmentation.

POC: Analyzing IP Addresses to Map Subnets

1. Capture Network Traffic
    - Capture traffic to see all IP addresses involved, using no filter initially to get a broad view, then save as a *pcapng* file for analysis.
2. List Unique IP Addresses
    - Use Wireshark's "Statistics > Conversations" to list all IP addresses, focusing on the IPv4 or IPv6 addresses in the "Address A" and "Address B" columns.
    - Alternatively, use *tshark* to extract IP addresses with:

```
tshark -r capture.pcap -T fields -e ip.src -e ip.dst | sort | uniq
```

This lists all unique source and destination IP addresses.

3. Determine Subnets
    - Look for patterns in IP addresses to identify subnets. For example, multiple IP addresses starting with 192.168.1.x suggest a subnet of 192.168.1.0/24. Use subnet calculators online, such as Subnet Calculator, to confirm.
4. Map Subnets to Functions
    - If possible, infer the function of each subnet (e.g., server subnet, user subnet, etc.) by analyzing traffic types. For instance, a subnet with many HTTP requests might be a web server subnet.

A tip for red teamers: Capture traffic over a sufficient period to get a comprehensive view, as short captures might miss devices that are intermittently active.

IDENTIFYING KEY NETWORK DEVICES

Devices like routers and switches can be identified through their behavior in network traffic, such as handling multiple IP addresses or being central to traffic flow.

POC: Using ARP Traffic to Identify Network Infrastructure

1. Capture ARP Traffic

Use a capture filter: `arp` to focus on ARP requests and replies, which map IP addresses to MAC addresses.

2. Identify ARP Replies from Routers and Switches
    - Routers often have multiple IP addresses (one for each interface) and might appear in ARP replies for different subnets. Switches might have a single IP address but are involved in ARP traffic for multiple devices, often with vendor-specific MAC addresses (e.g., Cisco or HP).
    - Look at the "Sender MAC Address" and "Sender IP Address" fields in ARP replies to note these devices.
3. Note Device IPs and MACs
    - Record the IP and MAC addresses of these devices for further analysis using Wireshark's export feature or manual notes to build a list of network infrastructure.

UNDERSTANDING NETWORK TRAFFIC FLOW TO MAP CONNECTIONS

By analyzing the flow of traffic, red teamers can understand how devices are connected, identifying central hubs and potential segmentation.

POC: Visualizing Network Connections Using Wireshark's Features

1. Use Wireshark's "Statistics > Conversations"
    - This feature, accessible under "Statistics > Conversations," shows which IP addresses are communicating with each other, listing source and destination IPs, ports, and packet counts.
2. Analyze the Conversations
    - Identify central nodes (likely servers or routers) that communicate with many other IPs, and peripheral nodes (likely clients) with fewer connections. Look at the "Packets" and "Bytes" columns to gauge traffic volume.
    - Use the "Follow TCP Stream" or "Follow UDP Stream" for specific conversations to see detailed interactions.

3. Interpret the Map
    - Infer network topology by noting which IPs are talking to each other frequently, suggesting direct connections, and identifying any isolated parts of the network with minimal traffic, indicating segmentation.

An unexpected detail is that while Wireshark doesn't have a graphical map feature in recent versions (as of March 1, 2025, the "Map" button was removed in version 2.6.0 due to dependency on deprecated GeoIP databases), the "Conversations" feature provides a tabular view that, with manual interpretation, can effectively map connections.

AUTOMATING WITH TSHARK

While the graphical interface of Wireshark is powerful, red teamers can also use tshark, the command-line version, to automate certain tasks, enhancing efficiency in reconnaissance.

Example: Listing All Unique IP Addresses

```
tshark -r capture.pcap -T fields -e ip.src -e ip.dst | sort | uniq
```

This command extracts all source and destination IP addresses from the capture file and lists them uniquely, aiding in subnet mapping.

Example: Extracting HTTP Credentials

To extract potential credentials from HTTP POST requests:

```
tshark -r capture.pcap -Y "http && http.request.method == \"POST\"" -T fields -e http.http
```

This command filters for HTTP POST requests and searches for lines containing "username" or "password," streamlining credential extraction.

This is just a basic example; more sophisticated parsing, such as using Python with pyshark, might be needed for accurate and automated extraction in larger engagements.

Table: Comparison of Cleartext Protocols for Credential Extraction

Protocol	Port Number	Credential Visibility	Common Use Case	Security Note
HTTP	80	Cleartext in POST/GET	Web login forms, internal apps	Vulnerable, use HTTPS instead
FTP	21	Cleartext USER/PASS	File transfers, legacy systems	Deprecated, use SFTP/FTPS
Telnet	23	Cleartext login/pwd	Remote terminal access, legacy	Highly insecure, use SSH

This table highlights the protocols, aiding red teamers in prioritizing targets for credential capture based on visibility and common usage.

CHAPTER 4 - ADVANCED RED TEAM TECHNIQUES WITH WIRESHARK

This chapter aims to provide red teamers—security professionals simulating adversarial activities to test organizational defenses—with practical, hands-on proof-of-concepts (POCs) demonstrating Wireshark's advanced capabilities in real-world scenarios. The chapter covers extracting data from SMB sessions, reverse engineering command-and-control (C2) communications, and bypassing network security with fragmented packets.

These scenarios assume authorized access to networks or systems under a legal engagement framework, emphasizing the ethical and responsible use of these techniques.

EXTRACTING DATA FROM SMB SESSIONS

The Server Message Block (SMB) protocol is widely used in Windows environments for file sharing and resource access, making it a prime target for red teamers to extract credentials or sensitive data. This POC demonstrates how to use Wireshark to capture and analyze SMB traffic, focusing on extracting NTLM hashes during session setup for cracking or relay attacks.

Prerequisites

- Wireshark installed.
- Access to a network with SMB traffic (e.g., via ARP spoofing or a compromised host).
- Tools like John the Ripper or Hashcat for cracking NTLM hashes.

POC: Extracting NTLM Hashes from SMB Traffic

1. Capture SMB Traffic

- Launch Wireshark and select the network interface (e.g., `eth0`).
- Apply a capture filter: `port 445` to focus on SMB traffic (TCP port 445 is standard for SMB).
- Start the capture during active SMB sessions, such as a client accessing a shared folder, and save it as `smb_session.pcap`

2. Filter for Session Setup Requests

- Open `smb_session.pcap` in Wireshark.
- Apply a display filter `smb2.command == 0x01` to isolate SMB2 session setup requests, where authentication data resides.
- Look for packets with "NTLMSSP_AUTH" in the "Security Buffer" field under the SMB2 layer in the packet details pane.

3. Extract NTLM Hashes

    - Expand the "NTLMSSP_AUTH" section to view fields like:

Username: e.g., `user123`

Domain: e.g., `DOMAIN`

NTLMv2 Response: e.g., `cafebabe12345678...`

- Manually note these values or use Wireshark's export feature `File > Export Objects > SMB` if files are transferred, though this POC focuses on credentials.
- Format the extracted data into a hash file for cracking tools, e.g., for John the Ripper:

```
user123:DOMAIN::cafebabe12345678...
```

Note: LM hashes are often absent in NTLMv2, replaced with dummy values like `AAD3B435B51404EE`

4. Crack the Hashes

- Use John the Ripper: `john --format=netntlmv2 smb_hashes.txt` with a wordlist (e.g., rockyou.txt) to attempt password recovery.
- Alternatively, use Hashcat with mode 5600: `hashcat -m 5600 -a 0 smb_hashes.txt wordlist.txt`

5. Red Team Application

- If the password is cracked (e.g., `P@ssw0rd123`), use it to authenticate to other systems in the network, facilitating lateral movement.
- Alternatively, relay the NTLM hash using tools like Responder or Impacket's `ntlmrelayx.py` to authenticate to other SMB services without cracking, assuming the relay is viable in the environment.

REVERSE ENGINEERING C2 COMMUNICATIONS

Command-and-control (C2) communications are essential for malware and implants to communicate with attackers. Red teamers can use Wireshark to capture and reverse engineer C2 traffic from their tools or real malware, improving their own implants' stealth and effectiveness.

Prerequisites

- Wireshark installed.
- A test environment with a C2 implant (e.g., a custom tool or open-source like Metasploit's Meterpreter).
- Basic scripting knowledge (e.g., Python for payload analysis).

POC: Analyzing and Mimicking HTTP-Based C2 Traffic

1. Set Up the C2 Environment
    - Deploy a simple HTTP-based C2 implant on a test machine (e.g., a Python script sending periodic GET requests to a C2 server at `192.168.1.200:8080`).
    - Start the C2 server to receive and respond to requests.

2. Capture C2 Traffic
    - In Wireshark, capture traffic on the test machine's interface, filtering for `ip.addr == 192.168.1.200 and tcp.port == 8080`
    - Save the capture `c2_traffic.pcap` after a few minutes of activity.

3. Analyze Traffic Patterns
    - Open `c2_traffic.pcap` and apply a display filter: `http` to focus on HTTP traffic.
    - Observe:
        - Request frequency (e.g., every 30 seconds).
        - Packet sizes (e.g., 200-300 bytes for requests, 100 bytes for responses).
        - Headers (e.g., `User-Agent`, `Host`).
        - Payloads (e.g., encoded commands like `cmd=whoami` in the URL).

Use "Follow TCP Stream" `right-click packet > Follow > TCP Stream` to see the full conversation, noting any encryption or encoding (e.g., base64).

4. Reverse Engineer the Protocol
    - Extract the payload format from the stream, e.g., `GET /data?cmd=base64encodedcommand`
    - Decode payloads manually or script it in Python:

```python
import base64
encoded = "d2hvYW1p"  # Example from capture
print(base64.b64decode(encoded).decode())  # Outputs: whoami
```

    - Identify the server's response format, e.g., plain text output or JSON.

5. Red Team Application
    - Adapt your implant to mimic legitimate traffic:
        - Match packet sizes and frequencies to typical HTTP traffic (e.g., browsing to a news site).
        - Use common headers like `User-Agent: Mozilla/5.0`
    - Recapture and analyze with Wireshark to verify stealth, ensuring it blends with normal traffic (e.g., compare with `Statistics > Conversations`).

BYPASSING NETWORK SECURITY WITH FRAGMENTED PACKETS

Network security devices like IDS/IPS often struggle to detect threats in fragmented traffic if they don't reassemble packets effectively. This POC shows how to craft and verify fragmented malicious traffic using Wireshark and Scapy to bypass such defenses.

Prerequisites

- Wireshark and Scapy installed (`pip install scapy`).
- A test network with an IDS/IPS (e.g., Snort) or a target environment under authorized testing.

POC: Crafting and Verifying Fragmented Malicious Traffic

1. Craft Fragmented Traffic with Scapy
    - Create a malicious HTTP request with an exploit payload

```
from scapy.all import *
payload = "GET /exploit?data=malicious_payload HTTP/1.1\r\nHost: target.com\r\n\r\r
packets = fragment(IP(dst="192.168.1.100")/TCP(dport=80)/payload, fragsize=32)
send(packets)
```

    - This splits the payload into 32-byte fragments, likely breaking any IDS signature across multiple packets.

2. Capture Traffic with Wireshark
    - Capture on the sending interface with no filter, then save as `fragmented_exploit.pcap`
    - Alternatively, capture on the target network to verify receipt.

3. Analyze Without Reassembly
    - Open `fragmented_exploit.pcap` in Wireshark.
    - Disable reassembly (`Edit > Preferences > Protocols > IPv4 > Uncheck "Reassemble fragmented IPv4 datagrams`).
    - Filter for `ip.dst == 192.168.1.100`, and check each fragment's "Data" field—ensure no single fragment contains the full "malicious_payload" string, reducing IDS detection likelihood.

4. Verify Functionality with Reassembly
    - Reenable reassembly and use "Follow TCP Stream" to confirm the complete HTTP request `GET /exploit?data=malicious_payload...` is intact, ensuring the target receives the exploit.
    - Compare packet counts and sizes in `Statistics > Conversations` to validate delivery.

5. Red Team Application

- Deploy this technique in a real engagement to deliver exploits or C2 traffic past IDS/IPS, confirming evasion by monitoring target logs or Wireshark captures for lack of alerts.
- Test against specific IDS/IPS (e.g., Snort) in a lab to refine fragment size based on its reassembly capabilities.

Table: Comparison of POC Techniques

Technique	Objective	Wireshark Role	External Tools
SMB Data Extraction	Extract NTLM hashes/files	Traffic capture, protocol decode	John/Hashcat
Reverse Engineering C2	Improve implant stealth	Pattern analysis, payload decode	Python for decoding
Fragmented Packet Bypass	Evade IDS/IPS detection	Verify fragmentation, reassembly	Scapy for crafting

Table: Comparison of Red Team Techniques in Wireshark

Technique	Objective	Wireshark Role	Example Command/Filter
Credential Extraction (HTTP)	Extract usernames/passwords	Decode HTTP, follow streams	`http.request.method == "POST"`
NTLM Hash Extraction (SMB)	Extract hashes for cracking	Filter session setup, decode	`smb2.command == 0x01`
SSL/TLS Decryption	Analyze encrypted traffic	Configure with keys, view data	Edit > Preferences > SSL
Evasion via Fragmentation	Bypass IDS/IPS detection	Disable reassembly, verify	Uncheck "Reassemble IPv4"
C2 Traffic Analysis	Reverse engineer communications	Analyze patterns, decode	`ip.addr == 192.168.1.200`