IS – IA1 REPORT

Group members: 16010122051 - Mayur Dumbre

16010122055 – Hriday Gandhi

16010122065 – Farzan Irani

1. Introduction

Wireshark is a powerful and widely-used open-source network protocol analyzer that enables users to capture and inspect network traffic in real-time. Initially developed in 1998 as Ethereal, Wireshark has grown into a robust tool trusted by network administrators, cybersecurity professionals, and IT enthusiasts worldwide. It provides unparalleled visibility into network communications, allowing users to analyze data at a granular level. Whether you're troubleshooting a slow network, debugging applications, or investigating potential security threats, Wireshark is an indispensable tool for understanding what's happening on your network.

---

2. Features/Characteristics

Wireshark offers a comprehensive suite of features that make it a go-to tool for network analysis:

1. **Live Traffic Capture**: Capture packets from live network traffic for immediate inspection.
2. **Protocol Support**: Supports hundreds of network protocols, including TCP, UDP, HTTP, DNS, FTP, and more.
3. **Detailed Packet Analysis**: Allows users to drill down into packet details, including headers, payloads, and timestamps.
4. **Advanced Filtering**: Provides powerful filtering options to focus on specific packets or conversations.
5. **Color-Coding**: Highlights different types of traffic with customizable color rules to simplify analysis.
6. **Cross-Platform Compatibility**: Runs seamlessly on Windows, macOS, Linux, and other platforms.
7. **Export and Reporting**: Enables users to save captured data in multiple formats and generate detailed reports for further analysis.

These features make Wireshark not only versatile but also user-friendly for beginners and experts alike.

---

3. Methodology

Wireshark operates by capturing network packets as they traverse a selected network interface, such as Ethernet or Wi-Fi, and displaying them in a structured, human-readable format. This methodology outlines the key steps involved in using Wireshark for network analysis, including setup, packet capturing, filtering, analysis, and exporting data.

1. Setup and Installation

Before using Wireshark, it must be installed and configured properly:

- **Downloading and Installing:**
    - Wireshark can be downloaded from its official website (https://www.wireshark.org/) and installed on various operating systems, including Windows, macOS, and Linux.
    - During installation, users may need to install additional components such as **Npcap** (for Windows) or **libpcap** (for Linux/macOS) to enable packet capturing.
- **Selecting the Network Interface:**
    - Once installed, Wireshark allows users to choose from available network interfaces (e.g., Ethernet, Wi-Fi, Bluetooth) to capture traffic.
    - The correct interface must be selected based on the type of traffic to be analyzed (e.g., Wi-Fi for wireless traffic, Ethernet for wired connections).

2. Capturing Packets

Wireshark enables real-time network traffic capture to monitor and analyze ongoing data exchange.

- **Starting a Capture Session:**
    - After selecting the appropriate interface, the user starts capturing packets by clicking the "Start Capture" button.
    - Packets are collected continuously and displayed in a structured format in the Wireshark GUI.
- **Stopping and Saving the Capture:**
    - Once enough data has been captured, the session can be stopped to avoid unnecessary storage consumption.

  - The captured traffic can be saved in **.pcap** or **.pcapng** format for future analysis. These file formats retain all packet details, including timestamps and metadata.

3. Filtering and Analysis

To manage large volumes of captured packets, Wireshark provides filtering and in-depth analysis tools.

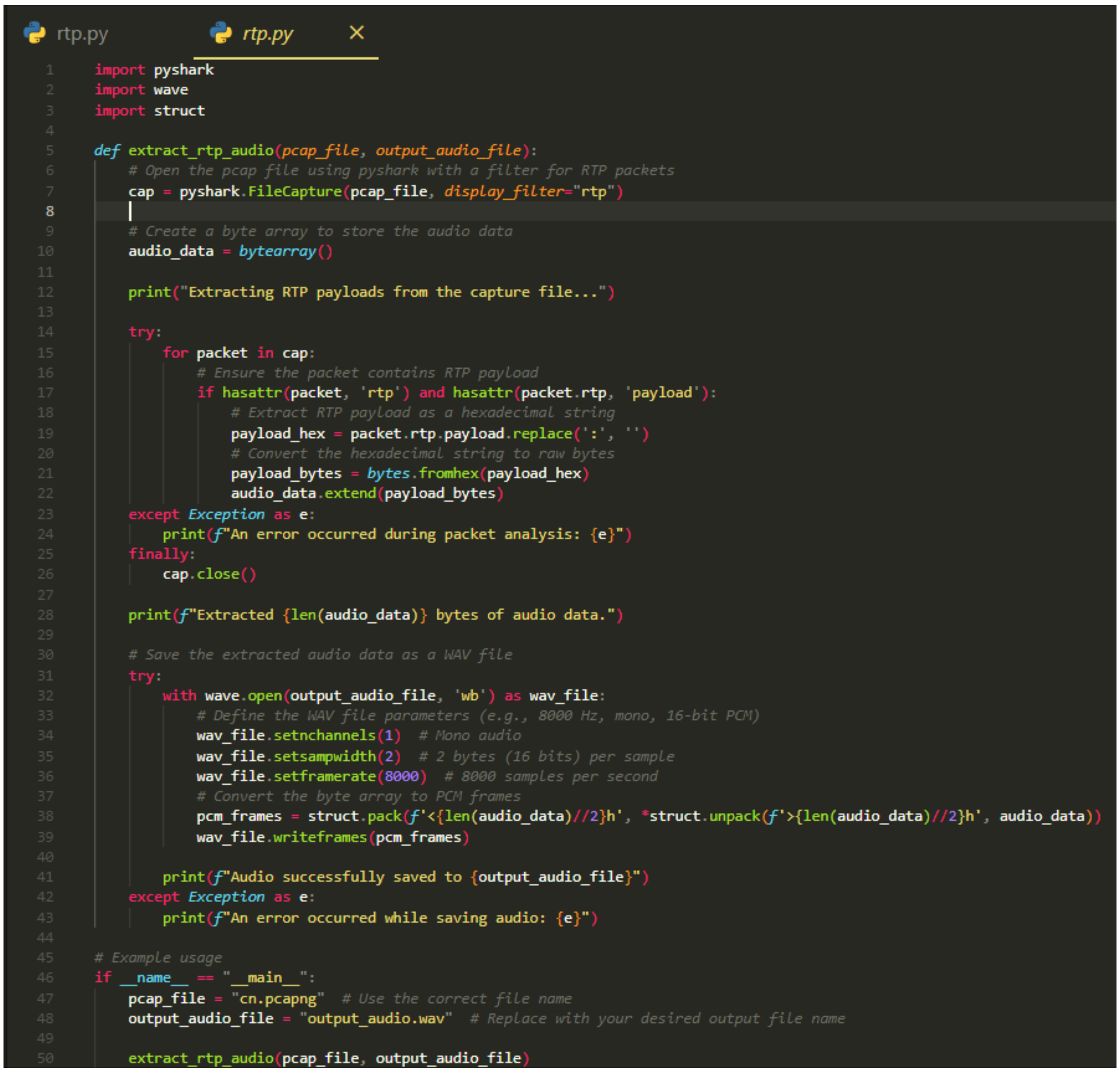
- **Applying Filters:**
  - **Capture Filters:** Used before starting a capture to collect only relevant traffic (e.g., capturing only TCP traffic using tcp filter).
  - **Display Filters:** Applied after capturing data to narrow down specific packets based on protocol, IP address, port number, or other parameters (e.g., filtering HTTP traffic with http).
- **Packet Inspection:**
  - Users can inspect individual packets by selecting them from the capture window.
  - Each packet contains three main sections:
    - **Packet List Pane:** Displays all captured packets with basic information.
    - **Packet Details Pane:** Provides in-depth details about the selected packet, including protocol layers.
    - **Packet Bytes Pane:** Shows the raw hexadecimal representation of the packet's data.
  - By analyzing packets, users can detect network anomalies, troubleshoot connectivity issues, or investigate security incidents such as unauthorized access attempts.

4. Exporting and Reporting

After analyzing network traffic, Wireshark allows users to export findings for documentation and further investigation.

- **Saving Analyzed Data:**
  - Users can save the filtered packets separately in **.pcap** or **.csv** formats for further analysis or sharing with others.
- **Generating Reports:**
  - Wireshark enables users to generate reports summarizing captured data, including key statistics like protocol distribution, IP conversations, and packet loss analysis.

- These reports are useful for network troubleshooting, forensic investigations, and performance optimization.



```python
import pyshark
import wave
import struct

def extract_rtp_audio(pcap_file, output_audio_file):
    # Open the pcap file using pyshark with a filter for RTP packets
    cap = pyshark.FileCapture(pcap_file, display_filter="rtp")

    # Create a byte array to store the audio data
    audio_data = bytearray()

    print("Extracting RTP payloads from the capture file...")

    try:
        for packet in cap:
            # Ensure the packet contains RTP payload
            if hasattr(packet, 'rtp') and hasattr(packet.rtp, 'payload'):
                # Extract RTP payload as a hexadecimal string
                payload_hex = packet.rtp.payload.replace(':', '')
                # Convert the hexadecimal string to raw bytes
                payload_bytes = bytes.fromhex(payload_hex)
                audio_data.extend(payload_bytes)
    except Exception as e:
        print(f"An error occurred during packet analysis: {e}")
    finally:
        cap.close()

    print(f"Extracted {len(audio_data)} bytes of audio data.")

    # Save the extracted audio data as a WAV file
    try:
        with wave.open(output_audio_file, 'wb') as wav_file:
            # Define the WAV file parameters (e.g., 8000 Hz, mono, 16-bit PCM)
            wav_file.setnchannels(1)  # Mono audio
            wav_file.setsampwidth(2)  # 2 bytes (16 bits) per sample
            wav_file.setframerate(8000)  # 8000 samples per second
            # Convert the byte array to PCM frames
            pcm_frames = struct.pack(f'<{len(audio_data)//2}h', *struct.unpack(f'>{len(audio_data)//2}h', audio_data))
            wav_file.writeframes(pcm_frames)

        print(f"Audio successfully saved to {output_audio_file}")
    except Exception as e:
        print(f"An error occurred while saving audio: {e}")

# Example usage
if __name__ == "__main__":
    pcap_file = "cn.pcapng"  # Use the correct file name
    output_audio_file = "output_audio.wav"  # Replace with your desired output file name

    extract_rtp_audio(pcap_file, output_audio_file)
```

---

4. Results

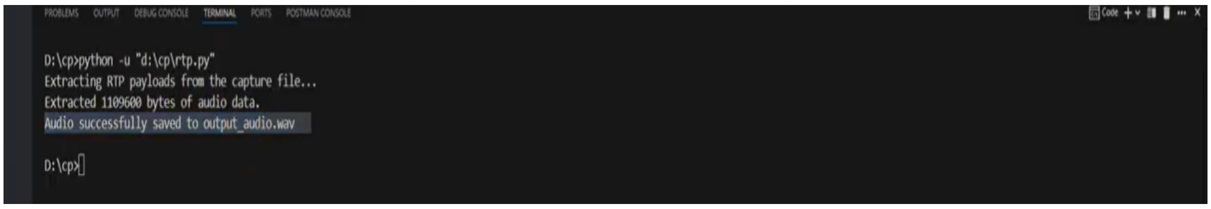
The use of Wireshark yields actionable insights into network traffic. Common results achieved with Wireshark include:

- **Troubleshooting**: Identifying causes of network slowdowns, packet loss, or congestion.
- **Security**: Detecting suspicious activity, such as unauthorized access or data exfiltration.
- **Protocol Analysis**: Understanding how various protocols operate in real-time communication.
- **Performance Optimization**: Pinpointing inefficiencies and bottlenecks in network performance.

- **Educational Value**: Providing hands-on experience with networking concepts and protocols for students and professionals.

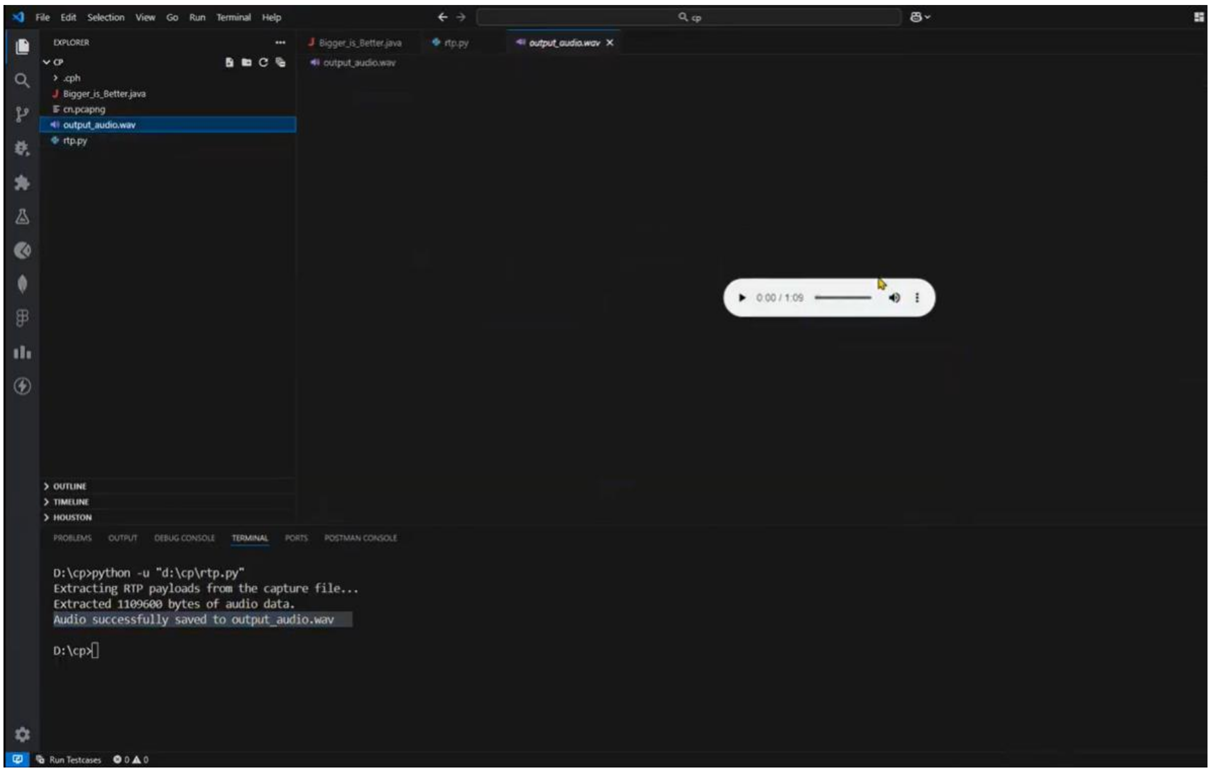
For instance, using Wireshark, users can reconstruct VoIP calls, diagnose DNS resolution delays, or identify rogue devices on a network.

Output:



```
D:\cp>python -u "d:\cp\rtp.py"
Extracting RTP payloads from the capture file...
Extracted 1109600 bytes of audio data.
Audio successfully saved to output_audio.wav

D:\cp>
```



```
D:\cp>python -u "d:\cp\rtp.py"
Extracting RTP payloads from the capture file...
Extracted 1109600 bytes of audio data.
Audio successfully saved to output_audio.wav

D:\cp>
```

---

5. Conclusion

Wireshark is an indispensable tool for anyone involved in networking or cybersecurity. Its ability to capture, inspect, and analyze network traffic provides users with unparalleled visibility into the data flowing through their networks. From troubleshooting and performance monitoring to security investigations and educational purposes, Wireshark's versatility makes it an essential tool in the digital age. By mastering Wireshark, users can enhance their understanding of network protocols, improve security measures, and ensure smooth operation of networked systems.