Name: Aum Panchal
Reg No: 22BCE8203

ABSTRACT

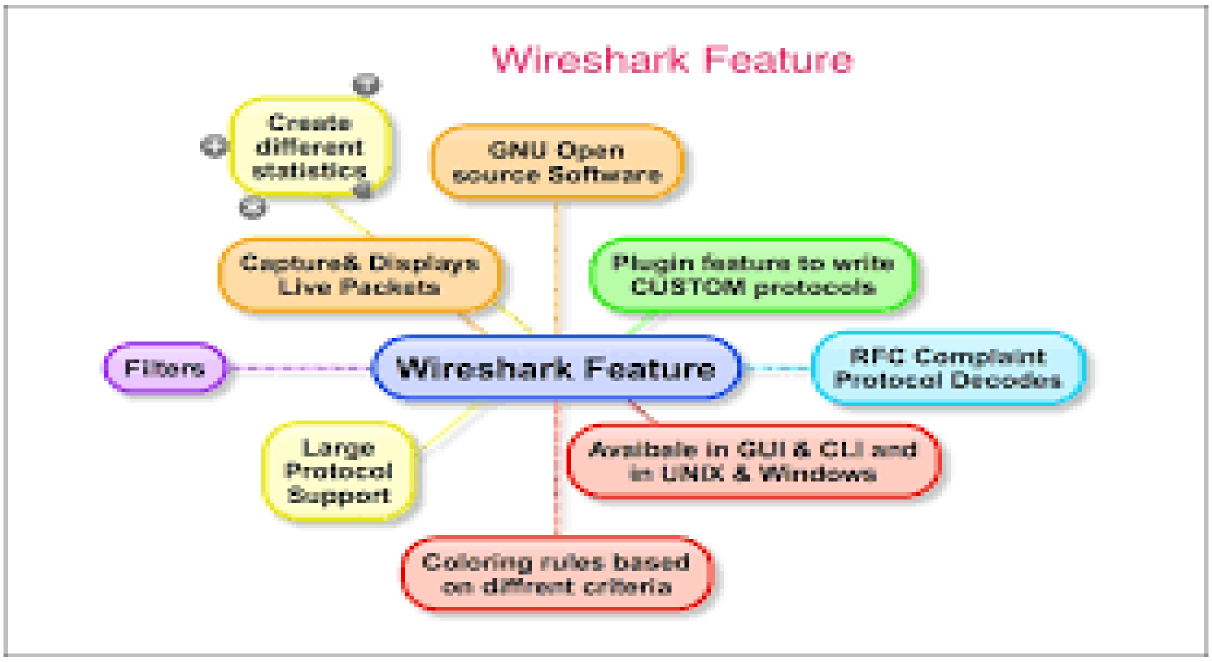
In the current era of advanced digital communication, organizations face an ever-increasing range of cyber threats that target their network infrastructure. Monitoring and analysing network traffic has become a critical aspect of maintaining security, optimizing performance, and ensuring regulatory compliance. The project titled **"Network Analysis Using Wireshark"** aims to provide a structured and detailed approach to inspecting network behaviour using Wireshark, a powerful open-source packet analyser. This tool enables real-time and offline analysis of network packets, offering deep insights into communication patterns, protocol behaviours, and potential security issues.

Key functionalities:

1. **Real-Time Packet Capture**
   - Capture live network traffic from selected network interfaces.
   - Monitor packets as they flow across the network in real time.
2. **Deep Packet Inspection (DPI)**
   - Analyse packet contents at all layers (Data Link, Network, Transport, and Application).
   - Dissect protocols like HTTP, DNS, TCP, UDP, ICMP, and more.
3. **Traffic Filtering and Protocol Analysis**
   - Apply powerful filters to isolate specific protocols, IP addresses, ports, or events.
   - Identify and analyse abnormal traffic patterns or unexpected protocol behaviour.
4. **Attack Detection & Signature Analysis**
   - Detect and investigate signs of network attacks such as:
     - ARP Spoofing
     - DNS Poisoning
     - SYN Flooding
     - ICMP-based scanning

INTRODUCTION

Wireshark is a powerful, open-source network protocol analyser that allows users to capture and interactively browse the traffic running on a computer network. Originally developed by Gerald Combs in 1998, Wireshark has become one of the most widely used tools for network troubleshooting, protocol development, and cybersecurity analysis.


Wireshark Feature
Create different statistics
GNU Open source Software
Capture& Displays Live Packets
Plugin feature to write CUSTOM protocols
Filters
Wireshark Feature
RFC Complaint Protocol Decodes
Large Protocol Support
Avaibale in GUI & CLI and in UNIX & Windows
Coloring rules based on diffrent criteria

Wireshark provides deep visibility into network traffic by capturing data packets in real time and displaying them in a human-readable format. It supports hundreds of protocols and media types, offering detailed insights into how systems and services communicate over a network.

The tool is available on multiple platforms, including Windows, Linux, and macOS, and it integrates with other command-line tools such as **TCPDump** and **TShark**. Common use cases include identifying misconfigurations, diagnosing network latency issues, detecting unauthorized access attempts, and learning about protocol behaviour.

Problem Statement:

One of the major challenges faced by network administrators today is the **inability to visualize and understand real-time traffic flows** at the packet level. Many traditional tools provide only superficial metrics or aggregated logs, which makes it difficult to identify the root causes of performance degradation or detect stealthy attacks such as ARP spoofing, DNS poisoning, and SYN floods. This project addresses this gap by utilizing Wireshark to perform comprehensive, packet-level analysis and make informed decisions based on actual traffic behaviour.

Tools and Applications Used:

- **Wireshark:** Core packet analyser used for live and offline packet analysis.
- **TCPDump:** Command-line packet capture utility.
- **Scapy (Python Library):** For crafting and injecting custom packets into the network.
- **VirtualBox:** For simulating a controlled test network using Linux and Windows virtual machines.
- **Kali Linux:** To simulate potential attacker behaviour and generate network anomalies.

These tools together help create, monitor, and analyse traffic in a controlled environment and provide complete visibility of network behaviour.

Scope of the Project:

The scope of the project **"Network Analysis Using Wireshark"** encompasses the comprehensive examination and evaluation of network traffic for the purposes of security auditing, performance monitoring, and protocol-level troubleshooting. The project is focused on both theoretical understanding and practical implementation of network analysis techniques using Wireshark.

❖ Key Areas Covered in the Project:

1. **Real-Time Traffic Monitoring:**
   Capture and observe live network packets to analyse communication between hosts.
2. **Protocol Analysis:**
   Study the behaviour of network protocols such as TCP, UDP, HTTP, DNS, ARP, and ICMP to understand their role in network communication.
3. **Offline PCAP File Analysis:**
   Analyse saved packet capture files for forensic investigation and post-event analysis.
4. **Traffic Visualization and Statistics:**
   Utilize Wireshark's built-in tools to generate visual representations such as protocol hierarchies, conversation lists, and endpoint summaries.

Limitations of the Project:

1. **Limited to Packet-Level Analysis**: Wireshark provides visibility at the packet level but does not offer high-level correlation or automated threat detection like modern Security Information and Event Management (SIEM) systems. Manual interpretation is required for threat identification.
2. **Requires Technical Expertise:** Effective use of Wireshark demands a good understanding of networking concepts and protocols. Beginners may find it challenging to interpret packet data and identify anomalies without guidance.
3. **Cannot Detect Encrypted Payloads:** Wireshark can capture encrypted traffic (e.g., HTTPS, SSL/TLS), but it **cannot decrypt encrypted payloads** without proper keys. This limits visibility into the content of secure communications.

4. **Static Environment Limitation:** The analysis in this project is performed in a **controlled, simulated environment** (e.g., VirtualBox VMs), which may not accurately reflect the complexity of dynamic, real-world enterprise networks.
5. **No Active Defence Mechanisms:** This project is focused solely on **passive traffic monitoring and analysis**. It does not include features for active defence or automated mitigation of threats.

Existing Solution:

In current network environments, organizations typically rely on a variety of tools and systems for traffic monitoring and security analysis, including:

1. **Firewalls and Intrusion Detection Systems (IDS):**
   These tools monitor incoming and outgoing traffic based on predefined rules and generate alerts for suspicious activity. However, they often operate at a high level and lack packet-level inspection.
2. **Log-based Monitoring Tools (e.g., Splunk, Nagios):**
   These tools collect system logs, flow data, and other metrics but provide limited visibility into real-time packet content or protocol-level behaviour.
3. **Network Monitoring Solutions (e.g., SolarWinds, PRTG):**
   These tools provide bandwidth and uptime statistics but are typically used for performance monitoring rather than security forensics.
4. **Packet Analysis Tools (e.g., TCPDump):**
   Command-line tools like TCPDump can capture packets but lack the graphical interface, filtering power, and protocol dissection that Wireshark offers.

Limitations of Existing Systems:

- Limited depth in packet inspection
- Minimal support for detailed forensic analysis
- Lack of educational usability and real-time visualization
- Require integration with multiple tools for full visibility

Proposed Plan:

To address the gaps in existing solutions, this project proposes the implementation of a **Wireshark-based network analysis framework** that focuses on **deep packet inspection**, **real-time traffic analysis**, and **educational clarity**. The proposed plan is divided into the following phases:

1. Environment Setup

- Use **VirtualBox** to create a simulated lab environment with Linux and Windows virtual machines.

2. Traffic Generation

- Generate both normal traffic (HTTP, DNS, FTP) and malicious traffic (ARP spoofing, SYN flood, etc.) to capture a variety of scenarios.
- Use **Scapy** or manual commands to inject crafted packets into the network.

3. Packet Capture with Wireshark

- Capture real-time traffic using Wireshark on a monitoring VM.
- Save traffic in .pcap format for offline analysis and documentation.

4. Packet Filtering and Protocol Analysis

- Use display filters to isolate traffic by protocol, IP, or port.
- Analyse protocols such as TCP, UDP, ICMP, ARP, DNS, and HTTP in detail.

5. Anomaly and Attack Detection

- Study the patterns and characteristics of malicious packets.
- Identify signatures or behaviours indicating attacks like DNS poisoning or ARP spoofing.

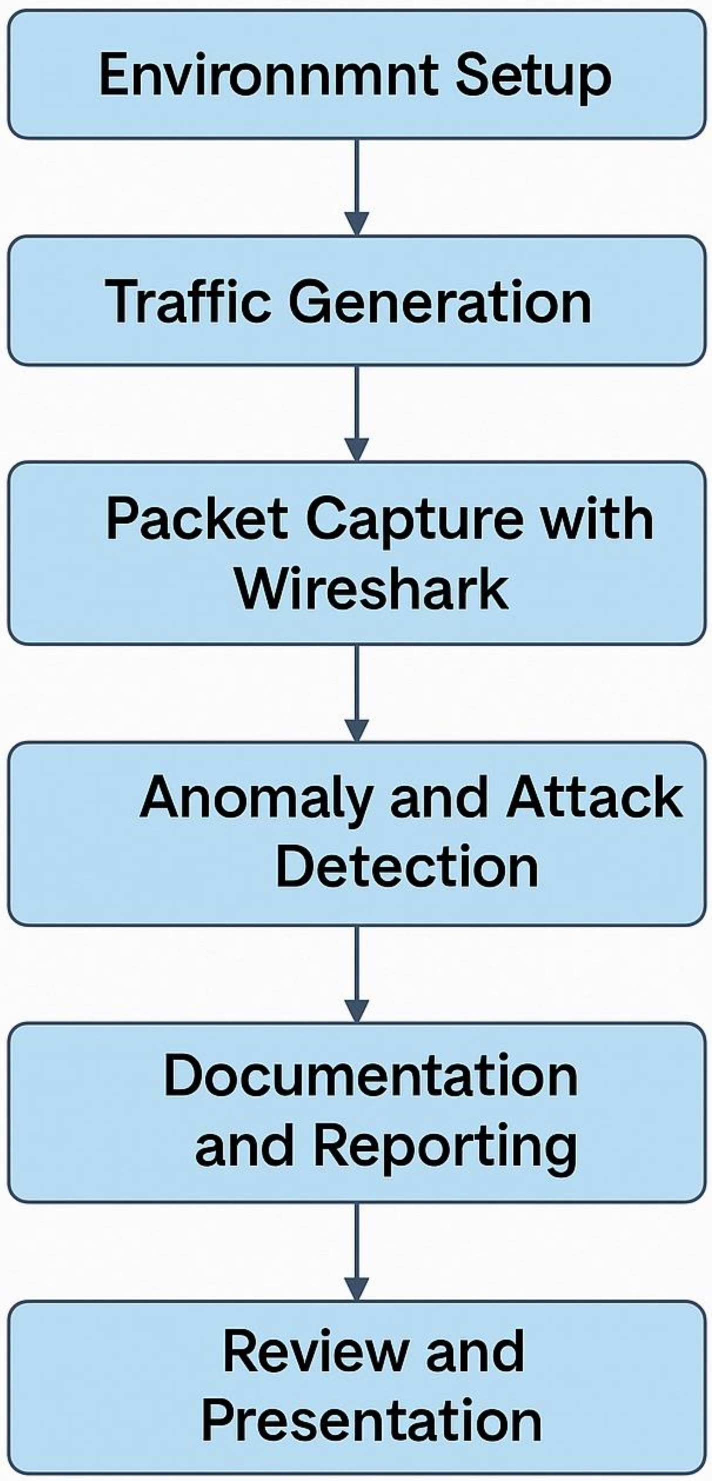
6. Documentation and Reporting

- Document findings with screenshots, flow diagrams, and protocol hierarchies.
- Generate reports summarizing anomalies, protocol usage, and suspected threats.

7. Review and Presentation

- Prepare a structured presentation of the workflow, findings, and lessons learned.
- Reflect on limitations and suggest future enhancements (e.g., real-time alerting, integration with SIEM).

Proposed Plan


Environnmnt Setup
Traffic Generation
Packet Capture with Wireshark
Anomaly and Attack Detection
Documentation and Reporting
Review and Presentation

Conclusion:

The project is expected to deliver a practical and educational framework for network traffic analysis using Wireshark. By the end of the project, users will be capable of performing deep packet inspection, identifying anomalies, recognizing signatures of common attacks, and documenting results effectively.