A

SYNOPSIS

of

MINOR PROJECT

on

Analyzing Network Communications with Packet Sniffing

Submitted by

JAWED KHAN

21EGICS048

Project Guide
Mr.Ajay Kumar Sharma

Head of Department
Dr. Mayank Patel

Geetanjali Institute of Technical Studies, Dabok , Udaipur (Raj.)
Department of Computer Science and Engineering
October,2023

Problem Statement:

In today's interconnected world, monitoring and analyzing network traffic is crucial for various purposes such as network security, performance optimization, and debugging. The challenge lies in developing a robust and efficient network packet sniffer that can capture and analyze packets flowing through a network interface.

Brief Description:

The project aims to create a network packet sniffer in C++ capable of intercepting and analyzing data packets transmitted over a network. This involves capturing raw packets, parsing their headers to extract essential information such as source and destination IP addresses, protocol types (TCP, UDP), and payload content. The program will provide functionalities to display packet details in real-time and log them for further analysis.

Objective and Scope:

The primary objective of the project is to design and implement a packet sniffer that:

- Captures network packets in real-time using raw sockets.
- Parses and interprets packet headers (Ethernet, IP, TCP/UDP) to extract relevant information.
- Displays packet information including source/destination IPs, protocols, and payload details.
- Implements basic filtering capabilities based on IP addresses or protocol types.

- Logs captured packets to a file for offline analysis.

The scope includes focusing on the implementation of low-level network programming techniques using C++, handling various packet types and protocols, and ensuring robust error handling to maintain reliability.

Methodology:

1. Packet Capture: Utilize raw sockets to capture packets directly from the network interface, bypassing the operating system's network stack.

2. Packet Parsing: Decode the headers of captured packets (Ethernet, IP, TCP/UDP) to extract pertinent information like source and destination IP addresses, protocol type, port numbers, and payload content.

3. Information Display: Present parsed packet details in a readable format either on the console or through a graphical user interface (GUI), allowing real-time monitoring of network traffic.

4. Filtering Mechanism: Implement filters to selectively capture packets based on specified criteria such as IP addresses or protocol types, enhancing the tool's flexibility for targeted analysis.

5. Logging: Provide an option to log captured packets to a file, facilitating comprehensive offline analysis and long-term traffic monitoring.

Hardware and Software Requirements:

- Hardware: A standard PC or laptop with an Ethernet interface.
- Software:
  - Operating System supporting raw socket programming (e.g., Linux, macOS).
  - C++ development environment (e.g., GCC/G++ compiler).
  - Optionally, libraries like libpcap for simplified packet capture operations.

Technologies:
- Programming Language: C++
- Libraries: Standard C++ libraries for socket programming, optionally libpcap for packet capture.
- Tools: Development tools (e.g., IDEs like Visual Studio Code, debugging tools), Wireshark for comparison and validation of captured data.

Testing Techniques:
- Unit Testing: Test individual functions and methods to ensure correctness and reliability in packet parsing, filtering, and logging.

- Integration Testing: Validate the interaction between various components of the packet sniffer (e.g., capture module, parsing module, user interface).

- System Testing: Evaluate the overall functionality and performance of the packet sniffer under various network conditions and traffic scenarios.

- Performance Testing: Measure and optimize packet capture and processing times to ensure efficient handling of high-volume network traffic.

Project Contribution:

This project contributes to the understanding and practical application of:

- Network Protocols: Understanding Ethernet, IP, TCP/UDP packet structures and their roles in data transmission.

- Network Programming: Developing proficiency in low-level network programming using raw sockets in C++.

- Security and Monitoring: Gaining insights into network security concepts, traffic monitoring techniques, and debugging practices.

By completing this project, one will enhance their skills in C++ programming, network protocol analysis, and software development for network utilities, preparing them for challenges in network administration, cybersecurity, and software engineering roles.