NETWORK PACKET SNIFFER

**Software Requirements Specification**

**Version <1.0.0>**

TEAM:

Parthiv Kumar Nikku [CB.EN.U4CYS22046]

Lalitha. K [CB.EN.U4CYS22037]

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <29/02/24> | <1.0.0> | SRS 1.0.0 | Group-16 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 5

2. Overall Description 5

3. Specific Requirements 5

3.1 Functional Requirements 5 3.1.1 Capture Packets 5

3.1.2 Packet Filtering 5

3.1.3 Packet Analysis 6

3.1.4 Exporting Data 6

3.2 Non-Functional Requirements 6

3.1.1 Performance 6

3.1.2 User Interface 6

3.1.3 Compatibility 6

3.3 Action Plan 6

3.4 Key Features 7

3.5 Benefits 7

4. System Architecture 8

4.1 Components 8

4.2 Deployment 8

5. Constraints 8

**Software Requirements Specification**

# **1.** **Introduction**

The Software Requirements Specification (SRS) provides a comprehensive overview of the Network Packet Sniffer project, outlining its purpose, scope, definitions, requirements, key features, abstract and system specifications. The aim of this document is to define the requirements for developing a **Network packet Sniffer.**

Network packet sniffer or simply packet sniffer is a packet analyser software that monitors all network traffic. The proposed project is implemented in Java programming language, and using this software admin of the system can capture network packets and analyse data received/sent from/to the network.

## **1.1** **Purpose**

The purpose and aim of this document is to provide a comprehensive overview of the requirements for developing a Network Packet Sniffer application using Java Programming Language.

## **1.2** **Scope**

The network packet sniffer will be capable of capturing and analysing network packets on a given network interface. The primary data captured by this software is the packets source and destination addresses. It will provide functionalities for packet capturing, filtering, and analysis.

It includes support for different network interfaces, comprehensive filtering options, detailed packet analysis, and compatibility across multiple operating systems.

## **1.3** **Definitions, Acronyms, and Abbreviations**

|  |  |
| --- | --- |
| JPCap | Java Packet Capture, a Java library for capturing and analyzing network packets. |
| PCAP | Packet Capture, a standard file format used for storing network packet data. |
| JAR | (Java Archive) file is a compressed file format used to package Java class files, associated metadata, and resources into a single archive for distribution or deployment. |
| CSV | Comma-Separated Values, a simple file format used to store tabular data. |
|  |  |

## **1.4** **References**

The references are:

<https://www.codewithc.com/network-packet-sniffer-java-project/#system-specificationsrequirements>

JPCap Library Documentation (GitHub Repository): <https://github.com/jpcap/jpcap/blob/master/README>

Oracle Java Documentation: <https://docs.oracle.com/en/java/>

**1.5** **Overview**

The network packet sniffer project aims to develop a robust Java application capable of capturing, filtering, and analysing network packets from various interfaces. This application targets network administrators, security analysts, and developers who require detailed insights into network traffic for monitoring, troubleshooting, and security analysis purposes.

**2. Overall Description**

The network packet sniffer project aims to develop a sophisticated Java application capable of monitoring, capturing, filtering, and analyzing network packets across various network interfaces. This application serves as a valuable tool for network administrators, security analysts, and developers who require detailed insights into network traffic for monitoring, troubleshooting, and security analysis purposes.

In conclusion, the network packet sniffer application aims to provide a robust and user-friendly solution for capturing, filtering, and analyzing network packets. By meeting the outlined requirements and specifications, the application will empower users with valuable insights into network traffic for efficient network management, security analysis, and troubleshooting purposes.

**3. Specific Requirements**

**3.1 Functional Requirements**

**3.1.1 Capture Packets**

The application shall capture packets from specified network interfaces using the JPCap library. It will support capturing packets from Ethernet, Wi-Fi, and other interfaces, with options for specifying capture filters.

**3.1.2 Packet Filtering**

Users should be able to apply filters based on various criteria such as source IP address, destination IP address, protocol type, port number, etc.

The application shall support logical operators for combining multiple filter conditions and provide pre-defined filter templates.

**3.1.3 Packet Analysis**

Captured packets will be parsed to provide detailed analysis and display relevant information such as source and destination addresses, protocol type, packet size, etc.

It should provide statistics on packet types, traffic volume, and other relevant metrics.

**3.1.4 Exporting Data**

Users should be able to export captured packet data in standard formats such as PCAP (Packet Capture) file format for compatibility with other tools.

Additionally, filtered packet data can be exported in text or CSV format for further analysis or reporting.

**3.2 Non-Functional Requirements**

**3.2.1 Performance**

The application shall be optimized for minimal CPU and memory usage to handle high-volume network traffic efficiently without performance degradation.

**3.2.2 User Interface**

The user interface should be intuitive and easy to navigate.

It should provide real-time updates on captured packets and analysis results.

**3.2.3 Compatibility**

The application should be compatible with various operating systems including Windows, macOS, and Linux.

It should support multiple network interfaces and adapt to changes in network configurations.

**3.3 Action Plan**

The action plan for the project involves:

1. Our project implementation starts with defining the project objectives and gathering the functional requirements
2. We will conceptualize the system architecture and define data models, interfaces, and interactions between system components.
3. This includes selecting and understanding the appropriate technologies and libraries, such as Java, JPCap, and JavaFX, to support the implementation of network packet sniffer application.
4. Then we go to the development phase where we will implement the core functionalities of the application based on the defined requirements and design.
5. It includes developing features for packet capturing, filtering, analysis, and user interface.
6. We will integrate with the JPCap library to leverage packet capturing and parsing capabilities.
7. Throughout the development phase, we will conduct regular code reviews and testing, documenting each step.

**3.4 Key Features**

* Real-time packet capturing
* 24×7 network monitoring
* Advanced protocol analyzing
* Comprehensive packet decoding
* Automatic expert diagnosing
* Complex network analyzing
* Conduct packet level analysis
* Solve network problems

**3.5 Benefits**

* Efficient Monitoring: Real-time analysis of network traffic for better understanding of network performance.
* Effective Troubleshooting: Quick identification and resolution of network issues for improved reliability.
* Enhanced Security Analysis: Detection and investigation of potential security threats for better network security.
* Customizable Filtering: Tailored filtering options to focus on relevant network traffic and extract valuable insights.
* Comprehensive Reporting: Export capabilities for generating detailed reports for analysis and compliance purposes.
* Cross-Platform Compatibility: Deployment on multiple operating systems for broader accessibility.
* User-Friendly Interface: Intuitive interface for easy navigation and utilization of application features.
* Scalability and Flexibility: Modular architecture allows for easy adaptation to changing network requirements and integration with other systems.

**4. System Architecture**

**4.1 Components**

The application will be built using Java and JavaFX (if required) for the graphical user interface (GUI).

It will integrate with the JPCap library for low-level packet capture and parsing functionalities. The system architecture will consist of modules for packet capturing, filtering, analysis, and the GUI.

**4.2 Deployment**

The application will be deployed as a standalone Java executable or packaged as a Java Archive (JAR) file for easy distribution.

**5. Constraints**

The application may require administrative privileges to access network interfaces for packet capturing.

It may be subject to limitations imposed by the underlying operating system or network hardware.