THE YENEPOYA INSTITUTE OF ARTS SCIENCE COMMERCE AND MANAGEMENT

BALMATTA, MANGLORE

YENEPOYA (DEEMED TO BE UNIVERSITY)

PROJECT REPORT ON

"WEB SERVER MONITORING TECHNIQUES"

SUBMITTED BY

MOHAMED SHAD

III BSC FORENSIC SCIENCE,DATA ANALYTICS,CYBER SECURITY WITH IBM

20BSFDC067

14847

UNDER THE GUIDANCE OF

MS. HANNATH

LECTURER

DEPARTMENT OF COMPUTER SCIENCE

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

"BACHELORS IN SCIENCE"

JUNE 2023

THE YENEPOYA INSTITUTE OF ARTS SCIENCE COMMERCE AND MANAGEMENT

BALMATTA, MANGLORE

YENEPOYA (DEEMED TO BE UNIVERSITY)

PROJECT REPORT ON

“WEB SERVER MONITORING TECHNIQUES”

SUBMITTED BY

MOHAMED SHAD

III BSC FORENSIC SCIENCE,DATA ANALYTICS,CYBER SECURITY WITH IBM

20BSFDC067

14847

UNDER THE GUIDANCE OF

MS. HANNATH

LECTURER

DEPARTMENT OF COMPUTER SCIENCE

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

“BACHELORS IN SCIENCE”

JUNE 2023

CERTIFICATE

This is to certify that the Project Report on “WEB SERVER MONITORING TECHNIQUES" submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor in Science during 2022-2023, THE YENEPOYA INSTITUTE OF ARTS, SCIENCE, COMMERCE AND MANAGEMENT, BALMATTA, (YIASCM), Mangalore by Mohamed Shad-20BSFDC067, that no part of this report has been submitted for the award of any degree, diploma, fellowship or similar titles or prizes and that the work has not been published in any journal or magazine.

APPROVED

INTERNAL GUIDE

MS. HANNATH

Department of Computer Science

The Yenepoya Institute of Arts, Science, Commerce and Management

Yenepoya (Deemed to be University)

PRINCIPAL

Dr. Arun Bhagawath

The Yenepoya Institute of Arts, Science, Commerce and Management

Yenepoya (Deemed to be University)

Place: Mangalore

Date: 23-06-2023

DECLARATION

I MOHAMED SHAD bearing Reg. No 20BSFDC067 here by declare that this project report entitled “WEB SERVER MONITORING TECHNIQUES” had been prepared by me towards the partial fulfilment of the requirement for the award of the Bachelor in Science at YENEPOYA INSTITUTE OF ARTS SCIENCE COMMERCE AND MANAGEMENT BALMATTA (YIASCM)

I also declare that this field study report is the result of my own effort and that it has not been submitted to any university for the award of any Degree, Diploma, Fellowship or similar prizes or titles and that the work has not been published on any journal or magazines.

APPROVED

INTERNAL GUIDE

MS. HANNATH

Lecturer

Bachelor of Computer Science

Yenepoya Institute of Arts Science Commerce and Management

Yenepoya (Deemed to be

PRINCIPAL

DR. Arun S BHAGAVATH

Yenepoya Institute of Arts Science Commerce and Management

Yenepoya (Deemed to be University)

Place: Mangalore

Date: 23-06-2023

ACKNOWLEDGEMENT

I, MOHAMED SHAD express my sincere thanks and gratitude to all those who have directly or indirectly helped me to complete my Project Report successfully

This project work is completed with immense amount of commitment, advice, encouragement and guidance of the people whom I could personally acknowledge

I would like to express my sincere gratitude to the Principal and Dean of Science Prof. (Dr.) ARUN BHAGAWATH to Vice Principal Dr. SHAREENA P & Vice Principal Dr. JEEVAN RAJ and to MR. NARAYANA SUKUMARAN (HOD)Department of Computer Science

I would like to express my sincere gratitude to MS. HANNATH (computer science) YENEPOYA COLLEGE, Mangalore, for her active support and guidance during my studies in this Institute. Without their kind supervision, preparing this report would be exceedingly difficult. 1 am also thankful to them for providing. I am also thankful to them for providing me all the relevant and available information to have a clear subject concept on the subjects. They provide me the guidance and counseling during my entire internship program. Their continuous and well-thought feedback enabled me to make this report a comprehensive one. I take this opportunity to extend thanks to all who has helped me and encouraged me all throughout in bringing the best of this project.

MOHAMED SHAD

III BSC IBM (BSFDC)

20BSFDC067

14847

Place: Mangalore

Date: 23-06-2023

TABLE OF CONTENTS

1. INTRODUCTION

The Network Analyzer Script serves as a valuable utility for network administrators, security professionals, and anyone interested in gaining insights into network traffic patterns. Whether the need to diagnose network issues, identify potential security threats, or optimize network performance, this script empowers with the ability to capture, analyze, and visualize network data with ease.

By utilizing this script, we can monitor network traffic in real-time, examine packet-level details, and gain a deeper understanding of network behavior. The script provides a user-friendly interface that allows us to choose from various monitoring options, such as capturing packets, analyzing protocols, and generating reports. It supports a wide range of network interfaces, enabling to monitor traffic on both wired and wireless networks.

1.1 Overview of the Project

The network analyzer script is a powerful tool designed to monitor and analyze network traffic in real-time. It provides network administrators, security professionals, and system analysts with valuable insights into network activity, helping them identify potential issues, troubleshoot problems, and optimize network performance.

With the increasing complexity and sophistication of modern networks, having a reliable network analysis tool is crucial for maintaining a secure and efficient network infrastructure. The network analyzer script aims to fulfill this need by offering a user-friendly interface and a comprehensive set of monitoring and analysis capabilities.

The script leverages various network protocols and techniques to capture and interpret network packets, enabling users to gain deep visibility into network traffic patterns, bandwidth usage, network latency, and other critical network metrics. It allows users to monitor specific network interfaces, filter packets based on different criteria, and generate detailed reports for further analysis.

Key features of the network analyzer script include:

1. Real-time Monitoring: The script provides real-time monitoring of network traffic, allowing users to observe network activity as it happens. This feature is particularly useful for detecting abnormal network behavior, identifying potential security threats, and troubleshooting network connectivity issues.

2. Packet Capture and Analysis: The script captures network packets and provides detailed analysis of packet headers, payload, and protocols. It supports a wide range of network protocols, including Ethernet, IP, TCP, UDP, HTTP, and more, enabling users to dissect and understand network traffic at a granular level.

3. Customizable Filters: Users can apply customizable filters to capture and analyze specific types of network traffic. This allows focusing on specific protocols, source or destination IP addresses, port numbers, or any combination thereof. Filtering helps narrow down the analysis scope and provides targeted insights into specific network segments or applications.

4. Statistics and Metrics: The script generates comprehensive statistics and metrics related to network traffic, including packet counts, bytes transferred, bandwidth utilization, packet loss, and round-trip time (RTT). These metrics provide quantitative data for performance evaluation, capacity planning, and troubleshooting network bottlenecks.

5. Reporting and Exporting: The script enables users to generate detailed reports based on captured network data. Reports can include summary statistics, graphical

representations, and other visualizations that help in presenting findings or sharing insights with colleagues and stakeholders. Additionally, the script allows exporting captured packets in standard formats for offline analysis or integration with other network analysis tools.

By providing a robust set of features and functionalities, the network analyzer script empowers network administrators and analysts to make informed decisions, optimize network performance, and enhance overall network security.

1.2 Objective of the Project

The objective of the network analyzer script is to provide a comprehensive and user-friendly tool for monitoring and analyzing network traffic. The project aims to achieve the following key objectives:

1. Network Traffic Monitoring: The primary objective of the network analyzer script is to monitor network traffic in real-time. By capturing and analyzing network packets, the script enables users to gain insights into network activity, identify potential issues, and track the performance of network connections and devices.

2. Network Troubleshooting: Another objective of the project is to assist network administrators and system analysts in troubleshooting network-related problems. The script offers advanced analysis capabilities that help identify network bottlenecks, packet loss, latency issues, and other performance anomalies. By pinpointing the root cause of network problems, users can take appropriate actions to resolve them promptly.

3. Security Analysis: The network analyzer script aims to contribute to network security by providing tools for analyzing and detecting potential security threats. It

allows users to examine network traffic for suspicious patterns, identify malicious activities, and detect network intrusion attempts. By enhancing network security monitoring, the script helps organizations protect their valuable data and resources from unauthorized access.

4. Performance Optimization: The project also focuses on optimizing network performance. The script provides insights into bandwidth utilization, application performance, and network latency, helping users identify areas for improvement. By analyzing network metrics and identifying performance bottlenecks, users can implement network optimizations to enhance overall network efficiency and user experience.

5. Reporting and Collaboration: The objective of the project includes facilitating effective reporting and collaboration among network administrators, security professionals, and system analysts. The script enables users to generate detailed reports based on captured network data, making it easier to communicate findings, share insights, and collaborate on network troubleshooting and optimization efforts.

The objective of the network analyzer script is to empower users with a powerful tool for monitoring, analyzing, and optimizing network traffic. By providing real-time visibility into network activity, advanced analysis capabilities, and reporting features, the project aims to enhance network performance, security, and overall operational efficiency.

1.3 Project Category

The network analyzer script falls under the category of network management and analysis tools. It belongs to the field of network administration and is specifically designed to assist network administrators, system analysts, and security professionals in monitoring, troubleshooting, and optimizing network traffic.

1.4 Tools and Platforms to be Used.

The development and implementation of the network analyzer script will require the following tools and platforms:

1. Linux Operating System: The project will be developed and executed on a Linux-based operating system such as Ubuntu, Fedora, or CentOS. Linux provides a robust and flexible environment for scripting and automation tasks, making it an ideal choice for this project.

2. Visual Studio Code (VS Code): VS Code is a popular and feature-rich code editor that supports various programming languages and offers powerful editing and debugging capabilities. It provides an intuitive user interface, extensibility through plugins, and integrated version control, making it a suitable choice for scripting the network analyzer.

3. Bash Scripting: The network analyzer script will be implemented using the Bash scripting language. Bash is a widely used scripting language on Linux systems and offers a rich set of built-in commands and utilities for automating tasks and interacting with the operating system. It provides flexibility and ease of use, making it an ideal choice for this project.

4. Wireshark: Wireshark is a widely used network protocol analyzer that allows capturing and analyzing network traffic. It provides a graphical user interface (GUI) and powerful packet analysis capabilities, making it an essential tool for inspecting and dissecting network packets. The network analyzer script may leverage Wireshark for capturing and analyzing network data.

5. TCP Dump: TCP Dump is a command-line packet analyzer that captures and analyzes network traffic. It runs on various operating systems, including Linux, and provides detailed packet-level information. The network analyzer script may utilize TCP Dump for capturing packets programmatically and extracting relevant information.

6. Networking Tools: The project may involve the use of various networking tools available on Linux, such as ping, traceroute, netstat, ipconfig, and Ip. These tools provide valuable information about network connectivity, routing, and interface configurations. They can be used within the script to gather network-related data and perform diagnostics.

1.5 Overview of the Technologies Used

1.5.1 Hardware Requirements

The network analyzer script project does not have specific hardware requirements as it primarily focuses on software development and network analysis. However, to execute the script and perform network analysis effectively, a computer system with the following minimum specifications is recommended:

- Processor: Dual-core or higher
- RAM: 4 GB or higher
- Storage: Sufficient disk space to store captured network data and the script itself
- Network Interface Card (NIC): A functional NIC capable of capturing network traffic

It is important to note that the hardware requirements may vary depending on the scale and complexity of the network analysis tasks performed using the script. For large-scale network analysis or capturing high volumes of network traffic, more powerful hardware may be required to ensure smooth execution and optimal performance.

1.5.2 Software Requirements

The network analyzer script project relies on various software components for development, execution, and network analysis. The following software requirements should be fulfilled:

- Linux Operating System: The project requires a Linux-based operating system such as Ubuntu, Fedora, or CentOS. The specific distribution can be chosen based on personal preference or organizational requirements. It is essential to ensure the availability of necessary networking tools and utilities on the chosen Linux distribution.

- Bash Shell: The network analyzer script is developed using the Bash scripting language. A compatible version of the Bash shell should be available on the Linux system. Typically, most Linux distributions include Bash as the default shell.

- Text Editor or Integrated Development Environment (IDE): A text editor or IDE is required to write, modify, and maintain the network analyzer script. Popular choices include Visual Studio Code (VS Code), Sublime Text, or any other preferred editor with support for Bash scripting. In this project, we recommend using VS Code for its rich features and extensions.

- Wireshark: Wireshark is a critical component for capturing and analyzing network traffic. It should be installed on the Linux system to leverage its powerful packet analysis capabilities. The appropriate version of Wireshark can be installed from the official website or package manager of the Linux distribution.

- TCP Dump: TCPDump is a command-line packet analyzer used for capturing and analyzing network traffic. It should be installed on the Linux system to enable programmatically capturing packets and extracting relevant information using the network analyzer script. TCP Dump can be installed using the package manager available on the Linux distribution.

1.6 Structure of the Program

The network analyzer script is designed with a modular structure to enhance code organization, reusability, and maintainability. It consists of the following modules:

1. User Interface Module:
   - This module handles user interaction and input validation.
   - It prompts the user for input parameters such as the network interface, capture duration, and analysis options.
   - It provides informative output to the user, displaying captured network data and analysis results.

2. Capture Module:
   - This module is responsible for capturing network traffic using the TCPDump tool.
   - It utilizes the specified network interface to capture packets for the specified duration.

- It saves the captured data to a temporary file for further analysis.

3. Analysis Module:

- This module performs various network analysis tasks on the captured data.
- It utilizes Wireshark's command-line tools and custom Bash scripts to extract relevant information from the captured packets.
- It analyzes packet headers, payload, and network statistics to provide insights into the network activity.

4. Reporting Module:

- This module generates detailed reports based on the analysis results.
- It formats and presents the analysis findings in a user-friendly manner.
- It generates reports containing information such as top protocols, source-destination pairs, and network performance metrics.

5. Utility Module:

- This module contains utility functions used across multiple modules.
- It includes functions for file manipulation, error handling, and data formatting.
- It provides reusable code snippets that enhance the overall functionality of the network analyzer script.

The modular structure of the program allows for easy maintenance and scalability. Each module focuses on a specific aspect of the network analysis process, making it easier to understand, modify, and extend the functionality of the script.

1.7 Statement of the Problem

The network analyzer script project aims to address the challenge of efficiently capturing and analyzing network traffic for diagnostic and troubleshooting purposes. Traditional network analysis tools often require manual interaction and lack the flexibility to automate repetitive tasks. This project aims to automate the network analysis process by developing a script that captures packets, extracts relevant information, and generates informative reports.

The primary problem this project seeks to solve is the time-consuming and error-prone nature of manual network analysis. By automating the process, network administrators, security professionals, and network enthusiasts can streamline their workflow, save time, and gain valuable insights into network behavior.

Additionally, the project aims to provide a user-friendly interface that simplifies the interaction with the script. By prompting users for input parameters and presenting analysis results in an organized manner, the script enhances usability and accessibility for users with varying levels of technical expertise.

The network analyzer script project addresses the need for a comprehensive and efficient tool that simplifies network analysis tasks, enables faster troubleshooting, and improves overall network performance.

2. SOFTWARE REQUIREMENTS SPECIFICATION

2.1 INTRODUCTION

2.1.1 Purpose

The purpose of this document is to outline the software requirements for the development of the network analyzer script. It specifies the functionalities, constraints, and performance criteria of the software.

2.1.2 Scope of the Project

The network analyzer script is intended to provide a tool for capturing and analyzing network traffic. It aims to automate the network analysis process and generate informative reports for diagnostic and troubleshooting purposes. The scope of the project includes the development of the script, its user interface, and integration with the necessary tools and platforms.

2.1.3 Intended Audience and Reading Suggestions

The intended audience for this document includes software developers, network administrators, system administrators, and individuals involved in network analysis tasks. It is recommended that readers have a basic understanding of network protocols, command-line interfaces, and scripting languages.

Reading Suggestions:

- Familiarize with network analysis techniques and tools.

- Understand the basics of network protocols and packet analysis.

- Gain knowledge of command-line interfaces and scripting languages.

2.1.4 Definitions, Acronyms, and Abbreviations

- TCP: Transmission Control Protocol

- UDP: User Datagram Protocol

- IP: Internet Protocol

- GUI: Graphical User Interface

- CLI: Command-Line Interface

- API: Application Programming Interface

2.2 OVERALL DESCRIPTION

2.2.1 Product Perspective

The network analyzer script is a standalone software application that interacts with the underlying operating system and network interfaces. It utilizes existing tools and libraries, such as TCPDump and Wireshark, for capturing and analyzing network packets. The script is designed to integrate seamlessly with the user's environment and provide a user-friendly interface for capturing, analyzing, and reporting network traffic.

2.2.2 Product Features

The key features of the network analyzer script include:

- Capturing network packets from a specified interface.
- Analyzing packet headers, payload, and network statistics.
- Generating informative reports based on the analysis results.
- Supporting various network protocols, including TCP, UDP, and IP.
- Providing a user-friendly interface for easy interaction.

2.2.3 User Characteristics

The users of the network analyzer script are expected to have basic knowledge of networking concepts and protocols. They may range from network administrators and security professionals to network enthusiasts and individuals involved in troubleshooting network issues. Users should be comfortable working with command-line interfaces and have a basic understanding of scripting languages.

2.3 OPERATING ENVIRONMENT

2.3.1 Design and Implementation Constraints

Design and implementation constraints for the network analyzer script include:

- Compatibility with Linux-based operating systems.
- Utilization of open-source tools and libraries.

- Support for common network interfaces and protocols.

- Efficient utilization of system resources (CPU, memory, disk space).

2.3.2 General Constraints

General constraints for the network analyzer script include:

- Adherence to applicable security guidelines and regulations.

- Compliance with open-source licenses and usage terms.

- Compatibility with the selected development platforms and programming languages.

- Consideration of potential limitations in network bandwidth and packet capture capabilities.

2.3.3 Assumptions and Dependencies

Assumptions and dependencies for the network analyzer script include:

- Availability of the necessary hardware and software resources.

- Proper configuration of network interfaces and permissions.

- Access to external tools and libraries (TCPDump, Wireshark, etc.).

- Proper functioning and compatibility of the underlying operating system.

2.4 SPECIFIC REQUIREMENTS

2.4.1 External Interface Requirements

2.4.1.1 User Interface

The user interface of the network analyzer script should:

- Provide interactive prompts for input parameters (interface, duration, etc.).

- Display captured network data and analysis results in a readable format.

- Support intuitive navigation and error handling for user interactions.

2.4.1.2 Hardware, Software, and Communication Interface

The network analyzer script requires the following interfaces:

- Hardware: Standard network interface cards (NICs) compatible with the operating system.

- Software: Linux-based operating system with required dependencies (TCPDump, Wireshark, etc.).

- Communication: Interaction with the network interfaces for packet capture and analysis.

2.4.2 Functional Requirements

The functional requirements of the network analyzer script include:

- Capturing network packets from the specified interface.

- Analyzing packet headers, payload, and network statistics.

- Extracting relevant information from the captured packets.

- Generating reports based on the analysis results.

- Providing options for filtering and sorting the captured data.

- Supporting command-line parameters for non-interactive usage.

2.4.3 Performance Requirements

The performance requirements of the network analyzer script include:

- Efficient packet capture without significant impact on network performance.

- Fast analysis of captured packets to provide real-time or near-real-time results.

- Optimized memory usage to handle large amounts of captured data.

- Minimized processing time for generating reports and displaying analysis results.

2.4.4 Design Constraints

Design constraints for the network analyzer script include:

- Utilization of open-source tools and libraries for packet capture and analysis.

- Adherence to the modular and extensible design for easy maintenance and scalability.

- Compliance with coding standards and best practices for readability and maintainability.

2.4.5 Requirements

The requirements for the network analyzer script include:

- Support for capturing and analyzing multiple network protocols (TCP, UDP, IP, etc.).

- Ability to filter captured packets based on specified criteria (source, destination, port, etc.).

- Generation of informative reports containing analysis findings and network statistics.

- Compatibility with Linux-based operating systems and common hardware configurations.

- Seamless integration with the user's environment and existing network analysis tools.

3. SYSTEM ANALYSIS AND DESIGN

3.1 INTRODUCTION

The system analysis and design phase involve analyzing the requirements of the network analyzer script and designing the system architecture and user interface to meet those requirements. This phase focuses on understanding the system's functional and non-functional aspects and developing an efficient design to implement the desired functionalities.

3.2 SYSTEM DESIGN IMPLEMENTATION

During the system design implementation, the architecture and components of the network analyzer script are defined. This phase involves breaking down the system into smaller modules and defining their functionalities and interactions.

3.2.1 Use Case

A use case represents a specific interaction between the user and the network analyzer script. It captures the actions performed by the user and the corresponding system response. The use cases for the network analyzer script may include:

- Capture Packet: This use case involves the user selecting the network interface and starting the packet capture process. The system captures network packets from the specified interface and stores them for further analysis.

- Analyze Packets: In this use case, the user selects the captured packets and initiates the analysis process. The system analyzes the packet headers, payload, and network statistics to derive meaningful insights.

- Generate Report: This use case allows the user to generate a report based on the analysis results. The system compiles the relevant information and presents it in a formatted report, including network statistics, identified anomalies, and any other relevant findings.

3.3 USER INTERFACE DESIGN

The user interface design focuses on creating an intuitive and user-friendly interface for the network analyzer script. It aims to provide easy navigation, clear information presentation, and efficient interaction with the system.

The user interface of the network analyzer script should include the following elements:

- Menu or Command Options: The user should be presented with a menu or command options to initiate various actions, such as capturing packets, analyzing them, generating reports, and configuring settings.

- Input Prompts: When required, the system should display input prompts to gather necessary information from the user, such as the network interface to capture packets from, the duration of capture, or any specific filters to apply.

- Data Presentation: The captured packet data and analysis results should be presented in a readable and structured format. This may include tables, graphs, or other visual representations to convey information effectively.

- Error Handling: The user interface should include proper error handling mechanisms to notify the user of any errors or issues encountered during the execution of commands or processes.

- Interactive Feedback: The system should provide feedback to the user during long-running operations, such as packet capture or analysis, to indicate progress and ensure that the user is aware of the system's activities.

The user interface design should prioritize simplicity, ease of use, and clarity to enhance the user experience and facilitate efficient interaction with the network analyzer script.

4. TESTING

4.1 INTRODUCTION

The testing phase aims to verify the functionality, reliability, and performance of the network analyzer script. This phase involves the systematic execution of test cases to identify any defects or inconsistencies in the script's behavior. The testing process ensures that the script meets the specified requirements and performs as intended.

4.2 TESTING OBJECTIVE

The primary objectives of testing the network analyzer script are as follows:

- Verify Functionality: The test cases will validate that the script correctly captures and analyzes network packets, generates accurate reports, and performs other intended functions as specified.

- Ensure Reliability: The script should consistently produce reliable results under varying network conditions and inputs. The testing process will assess the script's stability, error handling, and robustness.

- Evaluate Performance: Performance testing will measure the script's efficiency in terms of packet capture speed, analysis speed, and resource utilization. This evaluation will help identify potential bottlenecks or areas for optimization.

- Validate Compatibility: The script should be compatible with different operating systems, network interfaces, and packet formats. Testing will ensure that the script functions correctly across a range of supported environments.

- Identify and Resolve Defects: The testing process will uncover any defects, bugs, or unexpected behavior in the script. These issues will be logged and addressed to improve the overall quality of the script.

4.3 TEST CASES

Test cases are designed to validate specific aspects of the network analyzer script. Each test case consists of a set of inputs, the expected outputs or behaviors, and the steps to execute the test. The test cases for the network analyzer script may include:

- Test Case 1: Packet Capture Functionality
  - Input: Select network interface and start packet capture.
  - Expected Output: Packets are captured and stored in the designated storage location.
  - Steps: Open the script, select the desired network interface, initiate packet capture, and verify that packets are being captured and saved.

- Test Case 2: Packet Analysis Accuracy
  - Input: Select captured packets for analysis.
  - Expected Output: Accurate analysis results, including packet statistics, protocols, and any identified anomalies.

- Steps: Load the captured packets into the script, initiate the analysis process, and verify that the analysis results match the expected outcomes.

- Test Case 3: Report Generation
  - Input: Analysis results and report parameters.
  - Expected Output: A comprehensive report containing relevant network statistics, analysis findings, and visual representations.
  - Steps: Provide the analysis results as input, specify the desired report parameters (e.g., format, content), generate the report, and review its content for accuracy.

It is important to create a comprehensive set of test cases that cover different aspects and functionalities of the network analyzer script. Each test case should be documented with clear instructions, expected results, and steps to execute the test.

5. SYSTEM SECURITY

5.1 Introduction

System security is a critical aspect of the network analyzer script to ensure the protection of sensitive data and prevent unauthorized access. The following security measures will be implemented:

- Access Control: Access control mechanisms will be implemented to restrict user privileges and define different levels of access based on user roles and responsibilities.

5.2 SOFTWARE SECURITY

Software security measures will be implemented to mitigate potential vulnerabilities and protect the network analyzer script from malicious attacks. These measures include:

- Input Validation: The script will validate and sanitize all user inputs to prevent common security vulnerabilities such as SQL injection, cross-site scripting (XSS), and command injection.

- Error Handling: Proper error handling mechanisms will be implemented to prevent information leakage and ensure that error messages do not disclose sensitive information about the system.

- Secure Coding Practices: The script will be developed following secure coding practices, such as avoiding hardcoded credentials, using parameterized queries, and applying secure coding guidelines specific to the programming language and framework used.

- Regular Updates and Patch Management: The script will undergo regular updates and patch management to address any identified security vulnerabilities and ensure that it remains up to date with the latest security patches.

6. CONCLUSION

In conclusion, the development of the network analyzer script has been guided by a comprehensive software requirements specification, meticulous testing, and adherence to security best practices. The script offers a powerful and reliable solution for capturing and analyzing network packets, generating insightful reports, and enhancing network troubleshooting and monitoring capabilities.

7. FUTURE ENHANCEMENTS

While the current version of the network analyzer script meets the specified requirements, there are several avenues for future enhancements and improvements. Some potential areas for further development include:

- Integration with Additional Protocols: Enhancing the script to support additional network protocols, expanding its versatility and usefulness in various network environments.

- Real-Time Monitoring: Adding real-time monitoring capabilities to the script, allowing users to visualize and analyze network traffic as it occurs.

- Advanced Reporting: Implementing more advanced reporting features, such as customizable dashboards, trend analysis, and anomaly detection to provide deeper insights into network performance.

- Enhanced Security Features: Continuously improving the script's security measures to stay ahead of evolving threats and address emerging vulnerabilities.

8 . APPENDIX

```bash
#!/bin/bash


clear
# Function to check if the script is running with root privileges
check_sudo() {
    if [[ $EUID -ne 0 ]]; then
        echo -e "\033[1;33m [!] Please run this script with sudo or as
root.\033[0m"
        exit 1
    fi
}




# Function to print colorful output
show_banner() {
echo ""

echo "███╗░░██╗███████╗████████╗░██╗░░░░░░░██╗░█████╗░██████╗░██╗░░██╗"
echo "████╗░██║██╔════╝╚══██╔══╝░██║░░██╗░░██║██╔══██╗██╔══██╗██║░██╔╝"
echo "██╔██╗██║█████╗░░░░░██║░░░░╚██╗████╗██╔╝██║░░██║██████╔╝█████═╝░"
echo "██║╚████║██╔══╝░░░░░██║░░░░░████╔═████║░██║░░██║██╔══██╗██╔═██╗░"
echo "██║░╚███║███████╗░░░██║░░░░░╚██╔╝░╚██╔╝░╚█████╔╝██║░░██║██║░╚██╗"
echo "╚═╝░░╚══╝╚══════╝░░░╚═╝░░░░░░╚═╝░░░╚═╝░░░╚════╝░╚═╝░░╚═╝╚═╝░░╚═╝"
echo ""
echo
"░█████╗░███╗░░██╗░█████╗░██╗░░░░░██╗░░░██╗███████╗███████╗██████╗░"
echo
"██╔══██╗████╗░██║██╔══██╗██║░░░░░╚██╗░██╔╝╚════██║██╔════╝██╔══██╗"
echo
"███████║██╔██╗██║███████║██║░░░░░░╚████╔╝░░░███╔═╝█████╗░░██████╔╝"
echo
"██╔══██║██║╚████║██╔══██║██║░░░░░░░╚██╔╝░░██╔══╝░░██╔══╝░░██╔══██╗"
echo
"██║░░██║██║░╚███║██║░░██║███████╗░░░██║░░░███████╗███████╗██║░░██║"
echo
"╚═╝░░╚═╝╚═╝░░╚══╝╚═╝░░╚═╝╚══════╝░░░╚═╝░░░╚══════╝╚══════╝╚═╝░░╚═╝"
echo ""
echo "\n\n"
}
```

```
print_colorful() {
    local color=$1
    local message=$2
    echo -e "\e[${color}m${message}\e[0m"
}
# Function to print a colorful banner
print_colorful_banner() {
    local color=$1
    local banner
    banner=$(show_banner)
    echo -e "\e[${color}m${banner}\e[0m"
}

show_menu() {
    echo -e "\033[1;32m╔════════════════════════════════════════╗"
    echo -e "║           \033[1;34mNetwork
Analyzer\033[1;32m            ║"
    echo -e "╠════════════════════════════════════════╣"
    echo -e "║          \033[1;33mSelect an
option:\033[1;32m            ║"
    echo -e "╠════════════════════════════════════════╣"
    echo -e "║   1. \033[1;33mMonitor TCP SYN
requests\033[1;32m         ║"
    echo -e "║   2. \033[1;33mMonitor TCP reset
connections\033[1;32m    ║"
    echo -e "║   3. \033[1;33mMonitor TCP established
conn.\033[1;32m    ║"
    echo -e "║   4. \033[1;33mMonitor TCP half-open
conn.\033[1;32m      ║"
    echo -e "║   5. \033[1;33mMonitor app. requests on
Web\033[1;32m      ║"
    echo -e "║   6. \033[1;33mMonitor HTTP GET
requests\033[1;32m        ║"
    echo -e "║   7. \033[1;33mMonitor server
bandwidth\033[1;32m         ║"
    echo -e "║   8. \033[1;33mCheck port status of the
server\033[1;32m  ║"
    echo -e "║   9.
\033[1;33mExit\033[1;32m                               ║"
    echo -e "╚════════════════════════════════════════╝"
}



# Function to monitor TCP SYN requests
monitor_tcp_syn_requests() {
    print_colorful 34 "Incoming TCP SYN requests:"
#    show_spinner &  # Start the spinner in the background
    sudo tcpdump -n -i "$interface" 'tcp[tcpflags] == tcp-syn'
```

```
}

# Function to monitor TCP reset connections
monitor_tcp_reset_connections() {
    print_colorful 34 "TCP reset connections:"
    sudo tcpdump -n -i "$interface" 'tcp[tcpflags] == tcp-rst'
}

# Function to monitor TCP established connections
monitor_tcp_established_connections() {
    print_colorful 34 "TCP established connections:"
    netstat -antp | awk '$6 == "ESTABLISHED" {print}'
}

# Function to monitor TCP half-open connections
monitor_tcp_half_open_connections() {
    print_colorful 34 "TCP half-open connections:"
    netstat -antp | awk '$6 == "SYN_SENT" {print}'
}

# Function to monitor specific application requests on the web server
monitor_application_requests() {
    print_colorful 32 "Monitoring specific application requests:"
    sudo tcpdump -n -i "$interface" host "$machine_ip"
}

# Function to monitor HTTP GET requests on the web server
monitor_http_get_requests() {
    print_colorful 32 "Monitoring HTTP GET requests:"
    sudo tcpdump -n -i "$interface" 'tcp port 80 and (tcp[((tcp[12:1] &
0xf0) >> 2):4] = 0x47455420)'
}

# Function to monitor server bandwidth
monitor_server_bandwidth() {
    print_colorful 32 "Server bandwidth monitoring:"
    sudo ifstat -t -i "$interface"
}

# Function to perform custom port scanning
custom_port_scan() {
    print_colorful 32 "Performing custom port scanning..."
    read -p "Enter the port(s) to scan (e.g., 80,443): " ports
    read -p "Use -Pn flag? (y/n): " use_pn
    if [[ $use_pn == "y" ]]; then
        sudo nmap -p $ports -Pn "$machine_ip"
    else
        sudo nmap -p $ports  "$machine_ip"
    fi
}

# Function to handle user input from the port scanning sub-menu
show_port_scan_menu() {
    echo -e ""
    echo -e "\033[1;32m╔════════════════════════════════════════╗"
    echo -e "║           \033[1;34mPORT SCANNING
MENU\033[1;32m           ║"
    echo -e "╠════════════════════════════════════════╣"
    echo -e "║       \033[1;33mSelect an
option:\033[1;32m                ║"
    echo -e "╠════════════════════════════════════════╣"
    echo -e "║    1. \033[1;33mCustom Port
Scan\033[1;32m                 ║"
    echo -e "║    2. \033[1;33mFull Port
Scan\033[1;32m                   ║"
    echo -e "║    3.
\033[1;33mBack\033[1;32m                             ║"
    echo -e "╚════════════════════════════════════════╝"
    echo -e ""
}

# Function to perform full port scanning
full_port_scan() {
    print_colorful 32 "Full Port Scannig Started"
    read -p "Use -Pn flag? (y/n): " use_pn
    if [[ $use_pn == "y" ]]; then
        sudo nmap -p- "$machine_ip" -Pn
    else
        sudo nmap -p- "$machine_ip"
    fi
}



# Print ASCII banner
print_colorful_banner 31

# Example usage:
check_sudo

# List available network interfaces
interfaces=$(ip -o link show | awk -F': ' '{print $2}')

echo -e ""
echo -e "\033[1;32m╔══════════════════════════════════════════╗"
echo -e "║       \033[1;34mAvailable network
interfaces\033[1;32m     ║"
echo -e "╠══════════════════════════════════════════╣"
echo -e "║          \033[1;33mSelect an
Interface\033[1;32m           ║"
echo -e "╠══════════════════════════════════════════╣"

# Display the numbered list of interfaces
counter=1
for interface in $interfaces; do
    echo -e "\033[1;32m║   $counter. \033[1;33m$interface\033[0m"
    ((counter++))
done

echo -e "\033[1;32m╚══════════════════════════════════════════╝"
echo -e ""

# Prompt for user input
while true; do
    echo -n "Enter the number of the interface you want to select: "
    read option

    if ((option >= 1 && option < counter)); then
        interface=$(echo "$interfaces" | awk -v option="$option"
'NR==option')
        echo -e "\nYou selected interface: $interface"
        break
    else
        echo -e "\nInvalid option. Please select a valid interface
number."
    fi
done



# Get machine's IP address
machine_ip=$(ip -o -4 addr show $interface | awk '{print $4}' | awk -
F'/' '{print $1}')

# Prompt for selecting an option
while true; do
    show_menu
    read -p "Enter your choice: " choice

    case $choice in
        1) monitor_tcp_syn_requests ;;
        2) monitor_tcp_reset_connections ;;
        3) monitor_tcp_established_connections ;;
        4) monitor_tcp_half_open_connections ;;
        5) monitor_application_requests ;;
        6) monitor_http_get_requests ;;
        7) monitor_server_bandwidth ;;
        8) while true; do
                show_port_scan_menu
                read -p "Enter your choice: " port_scan_choice

                case $port_scan_choice in
                    1) custom_port_scan ;;
                    2) full_port_scan ;;
                    3) break ;;
                    *) echo "Invalid choice. Please try again." ;;
                esac
            done
            ;;
        9) exit ;;
        *) echo "Invalid choice. Please select a valid option." ;;
    esac
done
```