

2023

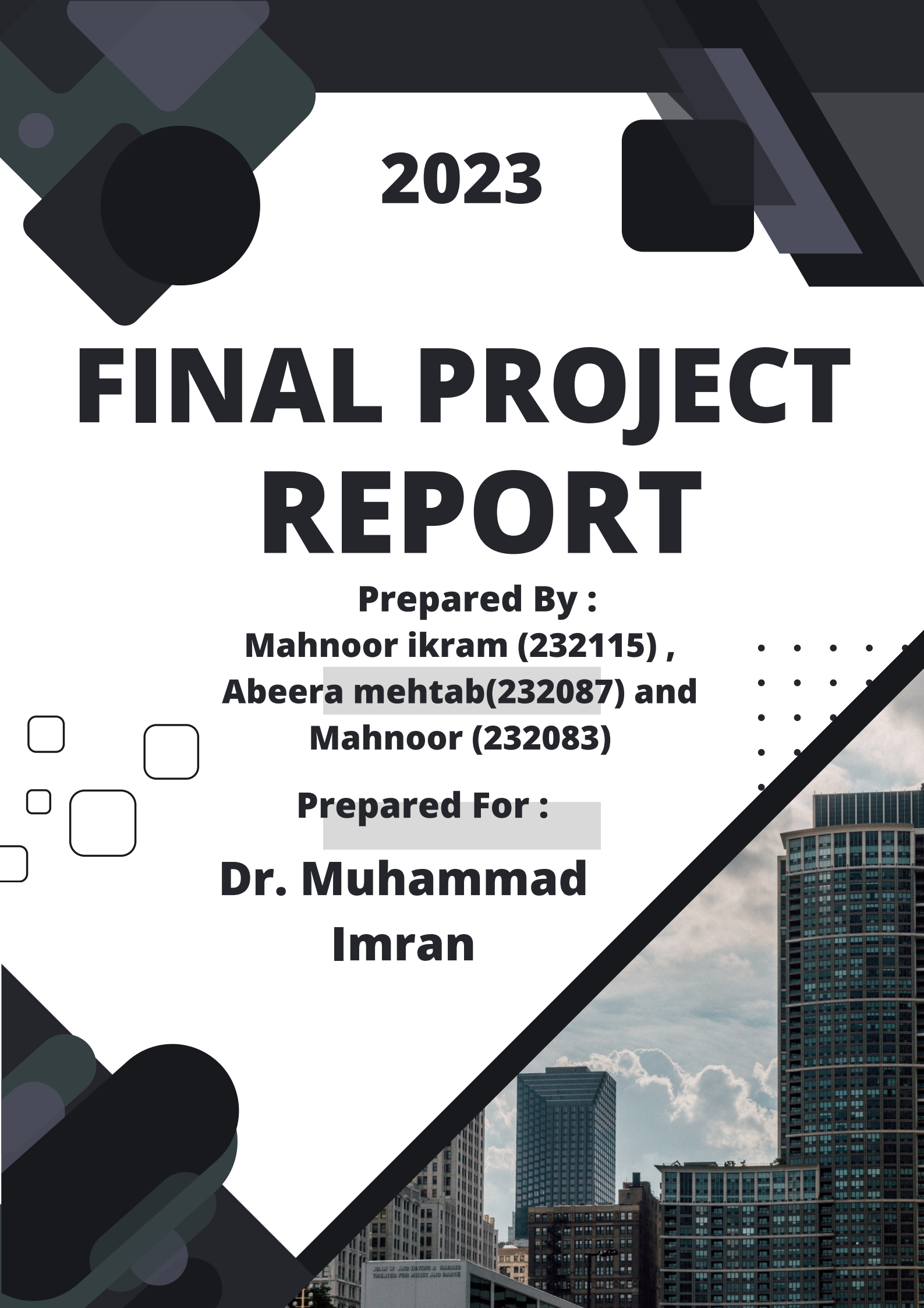
FINAL PROJECT REPORT

Prepared By :

**Mahnoor ikram (232115) ,
Abeera mehtab(232087) and
Mahnoor (232083)**

Prepared For :

**Dr. Muhammad
Imran**





Introduction

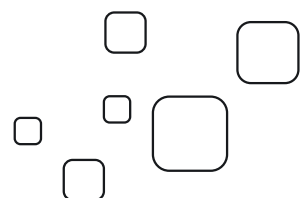
The project involves the development of a firewall program designed to regulate and control network traffic based on specified rules. A firewall acts as a security barrier, allowing or denying data packets entry into a secure network according to predefined criteria. The project aims to implement efficient computational tasks, sharpen programming skills, and deepen understanding of security concepts.

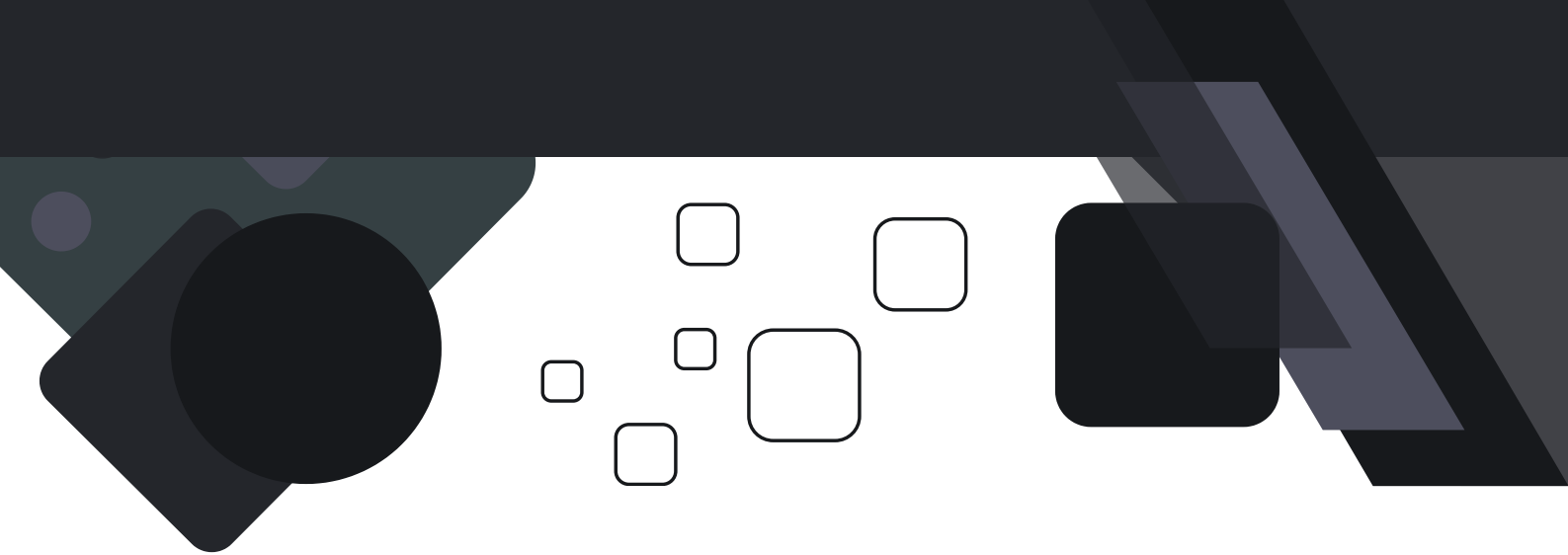
Purpose:

The primary purpose of this assignment is fourfold:

Out-of-the-Box Thinking: Encouraging creative problem-solving and innovative thinking by designing an efficient firewall program.

Skill Enhancement: Providing an opportunity to apply and enhance programming skills by implementing course concepts in a real-world context.





Security Concept Understanding: Facilitating a better understanding of security concepts through the practical implementation of a firewall, a fundamental security component.

Teamwork Skills: Fostering teamwork skills by requiring collaboration among team members for the successful completion of the project.

The development of the firewall program aligns with the course objectives, ensuring practical application of theoretical concepts and preparing students for real-world scenarios in the field of computer science and network security.

Team Information

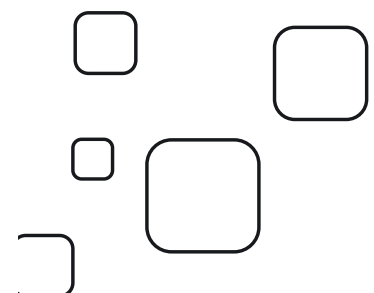
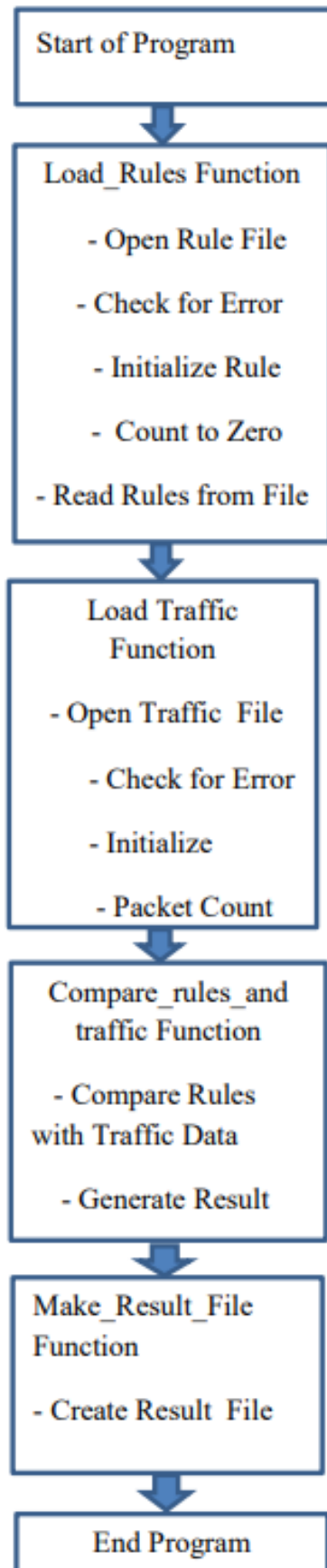
Team Members and task contribution:

Mahnoor 232087: Research and Understanding and Building Logic, Function Creation and Implementation .

Mahnoor Ikram 232115: Research, Testing and Debugging, Report Writing Flowchart creation.

Abeera 232083: Function Creation and Implementation Final code writing.

Flowchart:





Design Choices and Logic Explanation

1. Load_Rules Function:

- Objective: Open the rule file, check for errors, and initialize rule-related variables.
- Logic:
 - The function begins by opening the rule file to access the specified rules.
 - Error checks ensure the validity of the file and its content.
 - Rule-related variables are initialized, including the rule count.
 - Rules are read from the file to be processed later in the program.

2. Load_Traffic Function:

- Objective: Open the traffic file, check for errors, and initialize packet count.
- Logic:
 - Opens the traffic file to retrieve traffic data.
 - Error checks ensure the file's integrity and content.
 - Packet-related variables are initialized, including the packet count.
 - Reads traffic data from the file for subsequent processing.

3. comp_rules_and_traffic Function:

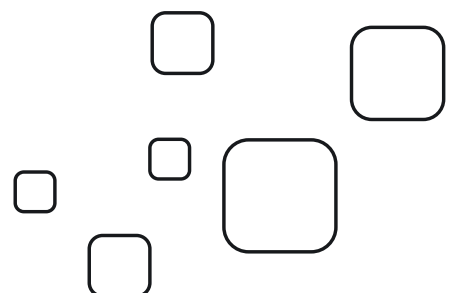
- Objective: Compare rules with traffic data and generate results.
- Logic:
 - Compares each rule with the traffic data, determining if the rule is applicable.
 - Generates results based on the applied rule for each data packet.

4. make_result_file Function:

- Objective: Create a result file and write results to it.
- Logic:
 - Creates a result file to store the outcome of rule application on traffic data.
 - Writes result details, including source and destination IP addresses, protocol, decision, and applied rule number.

5. End of Program:

- Objective: Mark the conclusion of the program.
- Logic:
 - Represents the end of the program flow.





Header File Explanation:

Purpose:

Defines structures (Rule and Packet) for storing rule and packet data.

Declares function prototypes used for loading rules, loading traffic, comparing rules with traffic, and creating a result file.

source File Explanation:

Contents:

Includes the header.h file.

Defines the main function where the program execution starts.

Creates arrays for rules and packets.

Calls functions to load rules and traffic, compare rules with traffic, generate results, and create a result file.

Purpose:

Implements the main logic of the firewall program.

Calls functions declared in the header file to perform specific tasks, such as loading rules and traffic, processing data, and creating output.



Functions Explanation:

Load_Rules:

Reads rule data from a file and populates the rules array.
Checks for file errors and ensures proper data retrieval.

Load_Traffic:

Reads traffic data from a file and populates the packets array.
Separates packets from the input string, extracts packet data, and handles potential errors.

comp_rules_and_traffic:

Compares rules with traffic data to determine the outcome for each packet.
Handles IP address ranges and applies the last rule if no rules match.

make_result_file:

Creates a result file and writes the results for each packet to the file.
Handles file errors during the process.

Logic:

Modular Approach:

Functions are designed to perform specific tasks, promoting code organization and readability.
Modular functions enhance maintainability and facilitate testing and debugging.

Structures:

Structures (Rule and Packet) provide a convenient way to organize and store related data.

Error Handling:

Functions include error checks to ensure file operations and data processing occur without issues.
Messages are displayed to notify users of any errors encountered during the program's execution.

Main Function:

Calls functions in a sequence to perform the required tasks.
Utilizes the result of comp_rules_and_traffic to generate and store the final results in a result file.



Problems Faced and Solutions

Problem: 1st problem Mai yeh editing karo: Difficulty in parsing packet data from traffic file due to following reasons

A) Dividing packets due to all packets coming in one continuous string

B) Removing Packet content Tags such as SRC, DST and PRO in storing them in Struct Arrays

Solution: Implemented a systematic approach using substrings and delimiters to extract packet information.

Problem: Handling IP address ranges in rule comparisons.

Solution: Modified the comparison logic to correctly handle IP address ranges using start and end IP addresses.

Limitations

The project currently supports a maximum of 100 rules and 100 packets. It may not scale well for larger datasets.

The program assumes a specific format for the rule and traffic files. Any deviation from this format may result in errors.

The system currently focuses on rule matching based on source IP, destination IP, and protocol, but more complex rule conditions may not be supported.



Conclusion

Summary

In conclusion, this project aimed to develop a firewall program capable of filtering network traffic based on specified rules. The report discussed the objectives, requirements, and implementation details of the project. Key components included functions to load rules and traffic, compare rules with incoming traffic, and generate a result file.

Achievements

Functionality Implementation:

Successful implementation of functions to read rule and traffic files, compare rules with traffic, and generate a result file.

Logical structuring of the program to handle rule evaluation for each packet, considering source and destination IP addresses, protocols, and decisions.

Header-Source Organization:

Effective organization of code into header and source files, promoting modularity and code reuse.

Clear struct definitions for rules and packets, enhancing code readability.

Error Handling:

Basic error handling mechanisms incorporated for file operations, providing informative messages in case of file-related issues.

Report and Documentation:

Creation of a detailed report outlining project objectives, requirements, design choices, and explanations of key functions.

Inclusion of a comprehensive flowchart illustrating the program's logic.

Team Collaboration:

Successful collaboration among team members in dividing tasks and contributing to different aspects of the project.

Effective communication and teamwork demonstrated throughout the project.