Project Report

On

FIREWALL CONFIGURATION TOOL

Submitted in partial fulfilment of the requirements for the award of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

(Artificial Intelligence & Machine Learning)

by

Ms. CH.BHAVYA (22WH1A6624)

Ms. L.LASYA (22WH1A6634)

Ms. A.AKSHAYA (22WH1A6647)

Ms. M.NANDITHA (22WH1A6649)

Under the esteemed guidance of

Ms. P Anusha

Assistant Professor, CSE(AI&ML)

Department of Computer Science & Engineering

(Artificial Intelligence & Machine Learning)

BVRITHYDERABAD COLLEGEOF ENGINEERINGFOR WOMEN

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with A Grade

Bachupally, Hyderabad – 500090

2024-25

ABSTRACT

The **Firewall Configuration Tool** is a command-line application designed to manage firewall rules effectively. It provides users with the ability to add, delete, list, and configure firewall rules for managing network traffic. This tool supports up to 100 rules, each specifying an IP address, port number, and action (either "ALLOW" or "BLOCK").

Key features of the application include:

- **Add Rule**: Enables the user to create a new firewall rule with specified parameters.
- **Delete Rule**: Allows the user to remove a rule by its index from the existing rule set.
- **List Rules**: Displays all current firewall rules for review.
- **Configure Firewall**: Simulates applying the defined rules to a firewall system.

The tool operates in an interactive loop, ensuring seamless user interaction with intuitive options for managing firewall configurations. While the current implementation focuses on functionality in a simulated environment, it can be extended to interact with real system commands or APIs for practical deployment.

PROBLEM STATEMENT

With the increasing complexity of networked systems, managing and securing network traffic has become a critical task. Organizations and individuals require tools to efficiently configure firewalls to control access and prevent unauthorized activities. However, many existing firewall configuration tools are either overly complex, requiring extensive expertise, or lack essential functionalities to manage rules dynamically.

The primary challenge is to create a lightweight, user-friendly application that allows users to add, delete, view, and apply firewall rules with minimal effort. The tool must enable users to define rules based on IP addresses, ports, and actions (e.g., "ALLOW" or "BLOCK") while ensuring simplicity and clarity in interaction. Additionally, scalability is essential to handle a reasonable number of rules, and the tool should simulate the process of configuring firewalls to provide users with feedback on applied rules. Addressing these challenges ensures improved security and accessibility in firewall management for both technical and non-technical users.

FUNCTIONAL REQUIREMENTS

1. **Add Rule**
    - The system must allow users to input an IP address, port number, and action ("ALLOW" or "BLOCK") to create a new firewall rule.
    - It must validate inputs for correct formats (e.g., IP address syntax).
2. **Delete Rule**
    - The system must allow users to delete a specific rule by providing its index in the rule list.
    - It must handle invalid indices with appropriate error messages.
3. **List Rules**
    - The system must display all current firewall rules, including their index, IP address, port number, and action.
    - It must indicate if no rules are available.
4. **Configure Firewall**
    - The system must simulate the application of defined rules to a firewall and display feedback for each applied rule.
5. **Interactive Menu**
    - The tool must provide a menu-driven interface to access functionality, with options to exit the application.
6. **Rule Capacity**
    - The system must support up to 100 rules, notifying the user when the limit is reached.

NON-FUNCTIONAL REQUIREMENTS

1. **Usability**
    - The tool must provide clear prompts and error messages to guide users effectively.
2. **Performance**
    - The system should perform all operations (add, delete, list, configure) within a few milliseconds, even at maximum capacity.
3. **Scalability**
    - The tool should allow future expansion of rule capacity without significant redesign.
4. **Portability**
    - The application must run on standard C-supported platforms with no dependency on external libraries.
5. **Reliability**
    - The tool should handle invalid inputs gracefully without crashing.
6. **Maintainability**
    - The code must be modular and well-documented to allow for easy updates or integration with system-level firewall configurations.
7. **Security**
    - The system should not expose sensitive data and must validate inputs to prevent injection attacks or invalid configurations.

SOURCE CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_RULES 100
typedef struct {
    char ip_address[16];
    int port;
    char action[10]; // "ALLOW" or "BLOCK"
} FirewallRule;
void add_rule(FirewallRule rules[], int *rule_count);
void delete_rule(FirewallRule rules[], int *rule_count);
void list_rules(FirewallRule rules[], int rule_count);
void configure_firewall(FirewallRule rules[], int rule_count);
int main() {
    FirewallRule rules[MAX_RULES];
    int rule_count = 0;
    int choice;
    while (1) {
        printf("\nFirewall Configuration Tool\n");
        printf("1. Add Rule\n");
        printf("2. Delete Rule\n");
        printf("3. List Rules\n");
        printf("4. Configure Firewall\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
```

```
            add_rule(rules, &rule_count);
            break;
        case 2:
            delete_rule(rules, &rule_count);
            break;
        case 3:
            list_rules(rules, rule_count);
            break;
        case 4:
            configure_firewall(rules, rule_count);
            break;
        case 5:
            printf("Exiting...\n");
            exit(0);
        default:
            printf("Invalid choice! Please try again.\n");
      }
   }

   return 0;
}
void add_rule(FirewallRule rules[], int *rule_count) {
   if (*rule_count >= MAX_RULES) {
      printf("Rule limit reached! Cannot add more rules.\n");
      return;
   }

   printf("Enter IP address: ");
   scanf("%s", rules[*rule_count].ip_address);
   printf("Enter port: ");
```

```
    scanf("%d", &rules[*rule_count].port);
    printf("Enter action (ALLOW or BLOCK): ");
    scanf("%s", rules[*rule_count].action);

    (*rule_count)++;
    printf("Rule added successfully.\n");
}
void delete_rule(FirewallRule rules[], int *rule_count) {
    if (*rule_count == 0) {
        printf("No rules to delete.\n");
        return;
    }

    int rule_index;
    printf("Enter the index of the rule to delete (0 to %d): ", *rule_count - 1);
    scanf("%d", &rule_index);

    if (rule_index < 0 || rule_index >= *rule_count) {
        printf("Invalid index! Please try again.\n");
        return;
    }

    for (int i = rule_index; i < *rule_count - 1; i++) {
        rules[i] = rules[i + 1];
    }

    (*rule_count)--;
    printf("Rule deleted successfully.\n");
}
void list_rules(FirewallRule rules[], int rule_count) {
```

```
    if (rule_count == 0) {
        printf("No rules to display.\n");
        return;
    }

    printf("\nCurrent Firewall Rules:\n");
    for (int i = 0; i < rule_count; i++) {
        printf("%d. IP: %s, Port: %d, Action: %s\n",
               i, rules[i].ip_address, rules[i].port, rules[i].action);
    }
}
void configure_firewall(FirewallRule rules[], int rule_count) {
    if (rule_count == 0) {
        printf("No rules to configure.\n");
        return;
    }

    printf("\nConfiguring firewall with the following rules:\n");
    for (int i = 0; i < rule_count; i++) {
        printf("Applying rule: IP: %s, Port: %d, Action: %s\n",
               rules[i].ip_address, rules[i].port, rules[i].action);
     system commands or APIs
    }

    printf("Firewall configured successfully.\n");
```

Output

Output

```
       Firewall Configuration Tool
====================================
1. Add Rule
2. Delete Rule
3. List Rules
4. Search Rule
5. Update Rule
6. Configure Firewall
7. Exit
Enter your choice: 1
Enter IP address (e.g., 192.168.1.1): 192.168.1.1:1080
Enter port (e.g., 80): 90
Enter action (ALLOW or BLOCK): ALLOW
Rule added successfully!

====================================
       Firewall Configuration Tool
====================================
1. Add Rule
2. Delete Rule
3. List Rules
4. Search Rule
5. Update Rule
6. Configure Firewall
7. Exit
Enter your choice: 1
Enter IP address (e.g., 192.168.1.1): 192.345.67:102


Enter port (e.g., 80): 23
Enter action (ALLOW or BLOCK): ALLOW
Rule added successfully!
```

Output

```
====================================
       Firewall Configuration Tool
====================================
1. Add Rule
2. Delete Rule
3. List Rules
4. Search Rule
5. Update Rule
6. Configure Firewall
7. Exit
Enter your choice: 1
Enter IP address (e.g., 192.168.1.1): 109.203.1.4:123
Enter port (e.g., 80): 34
Enter action (ALLOW or BLOCK): ALLOW
Rule added successfully!

====================================
       Firewall Configuration Tool
====================================
1. Add Rule
2. Delete Rule
3. List Rules
4. Search Rule
5. Update Rule
6. Configure Firewall
7. Exit
Enter your choice: 2
Enter the index of the rule to delete (0 to 2): 0
Rule deleted successfully!

====================================
```

Output

```
==================================
       Firewall Configuration Tool
==================================
1. Add Rule
2. Delete Rule
3. List Rules
4. Search Rule
5. Update Rule
6. Configure Firewall
7. Exit
Enter your choice: 3

Current Firewall Rules:
0. IP: 2.345.67:102, Port: 23, Action: ALLOW
1. IP: 109.203.1.4:123, Port: 34, Action: ALLOW

==================================
       Firewall Configuration Tool
==================================
1. Add Rule
2. Delete Rule
3. List Rules
4. Search Rule
5. Update Rule
6. Configure Firewall
7. Exit
Enter your choice: 4
Enter IP address to search: 109.203.1.4:123
Enter port to search: 23
No matching rule found.
```

Output

```
==================================
       Firewall Configuration Tool
==================================
1. Add Rule
2. Delete Rule
3. List Rules
4. Search Rule
5. Update Rule
6. Configure Firewall
7. Exit
Enter your choice: 4
Enter IP address to search: 109.203.1.4:123
Enter port to search: 34
Rule Found: IP: 109.203.1.4:123, Port: 34, Action: ALLOW

==================================
       Firewall Configuration Tool
==================================
1. Add Rule
2. Delete Rule
3. List Rules
4. Search Rule
5. Update Rule
6. Configure Firewall
7. Exit
Enter your choice: 5
Enter the index of the rule to update (0 to 1): 1
Enter new IP address (e.g., 192.168.1.2): 123.45.67:8
Enter new port (e.g., 443): 144
Enter new action (ALLOW or BLOCK): ALLOW
Rule updated successfully!
```

Output

```
==================================
       Firewall Configuration Tool
==================================
1. Add Rule
2. Delete Rule
3. List Rules
4. Search Rule
5. Update Rule
6. Configure Firewall
7. Exit
Enter your choice: 6

Applying Firewall Configuration:
Applying Rule: IP: 2.345.67:102, Port: 23, Action: ALLOW
Applying Rule: IP: 123.45.67:8, Port: 144, Action: ALLOW
Firewall configured successfully!

==================================
       Firewall Configuration Tool
==================================
1. Add Rule
2. Delete Rule
3. List Rules
4. Search Rule
5. Update Rule
6. Configure Firewall
7. Exit
Enter your choice: 7
Exiting Firewall Configuration Tool. Goodbye!
```