**INSTRUCTOR: Dr. Mohammad Imran**

Object Oriented Programming (CS-112)



semester project

**Semester Project Report**

**Team Members Name:**

* M Shaheer Ali Tarar (232078)
* M Imad Ul Islam (232082)
* Munawar Masood (232932)

**Task division among team members:**

* **Front End:**
* M. Imad Ul Islam
* Munawar Masood
* **Back End:**
* Shaheer Ali

**Design:**

**Class Hierarchies:**

Authenticate Form

IP\_Address

Rules

Packets

Firewall Info Form

Add Rules Form

File Dialog Box

Add Rules Panel

Packet Panel

File Dialog Box

Add Packet Panel

Result Panel

**Implementation:**

* **Program Flow:**

Following is the Program Flow

1. **Login Interface:**

* **Purpose:** Ensures secure access to firewall settings.
* **Action:** Users must authenticate by entering a valid password to proceed.

1. **Firewall Information Form:**

* **Buttons**:
* **Exit:** Exits the firewall interface.
* **Next:** Proceeds to the "**Add Rule**" form.

1. **Adding Rule Interface:**

* **Buttons:**
* **Add:** Adds new firewall rules.
* **Browse:** Imports rules from a file.
* **Clear:** Removes all added rules.
* **Action:** Click "**Next**" to proceed to the "**Adding Packets**" form.

1. **Adding Packets Interface:**

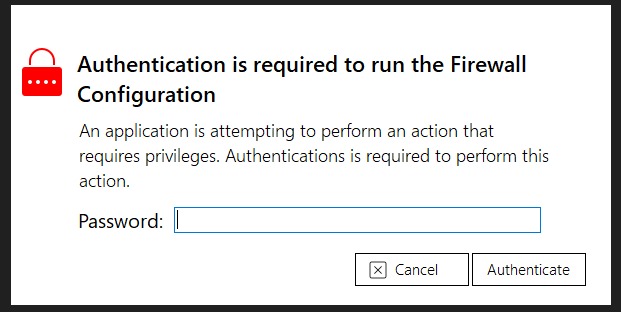
* **Buttons:**
* **Add:** Adds new firewall packets.
* **Browse:** Imports packets from a file.
* **Clear:** Removes all added packets.
* **Action:** Click "**Next**" to proceed to the "**Result Comparison**" interface.

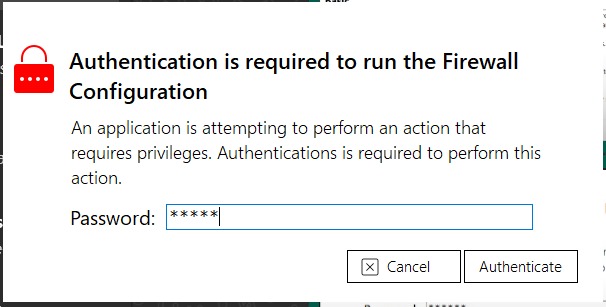
1. **Result Comparison Interface:**

* **Action:** The firewall will compare the packets with the rules defined by the user, reading each packet, searching the rules, and applying the appropriate actions based on those rules.
* **Table Columns:**
* **Result:** Indicates whether traffic is allowed or denied.
* **Action:** Describes the action taken (allow or deny).
* **Source:** Shows the IP address of the initiating device.
* **Destination:** Shows the IP address of the receiving device.
* **Src Port:** Displays the source port number.
* **Dst Port:** Displays the destination port number.
* **Protocol:** Indicates the protocol used (e.g., TCP, UDP).
* **Data:** Shows data in packet.
* **Time:** Shows the date and time the rule was applied.
* **Buttons:**
* **Show Rules:** Displays all the current rules.
* **Back:** Returns to the previous screen
* **User Manual:**

1. **Login Page:**

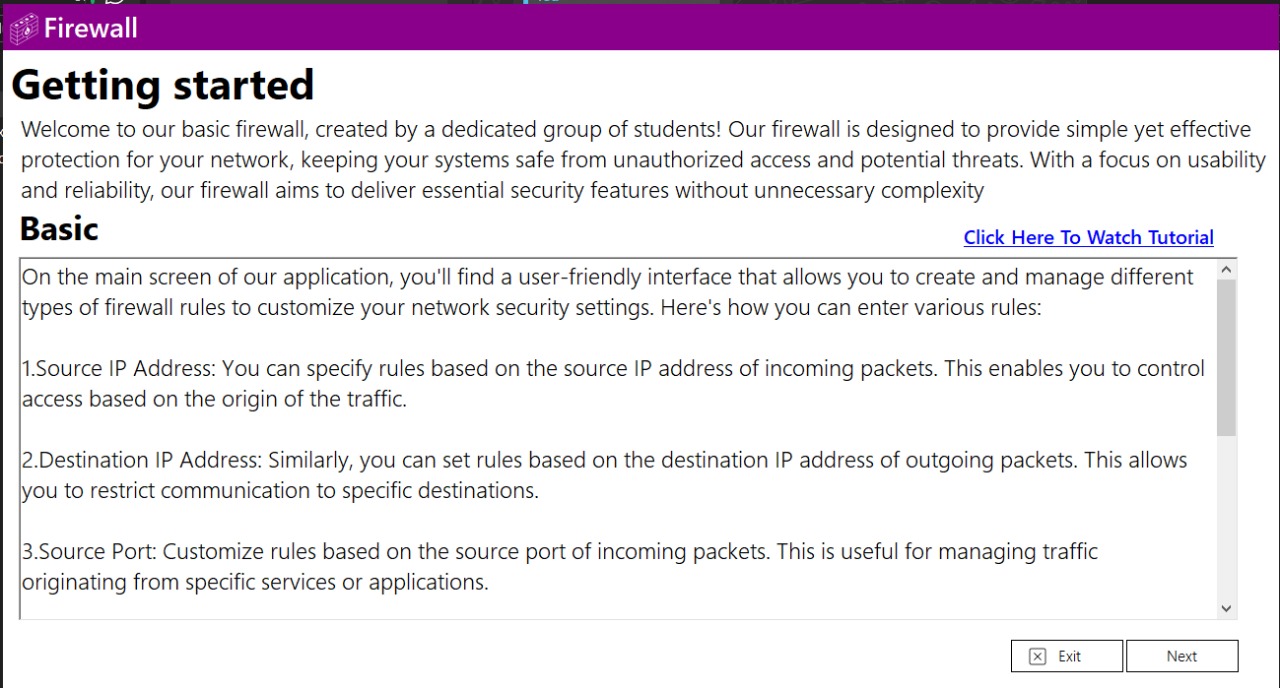
login page ensures secure access to firewall settings. Users must authenticate to perform any privileged actions.





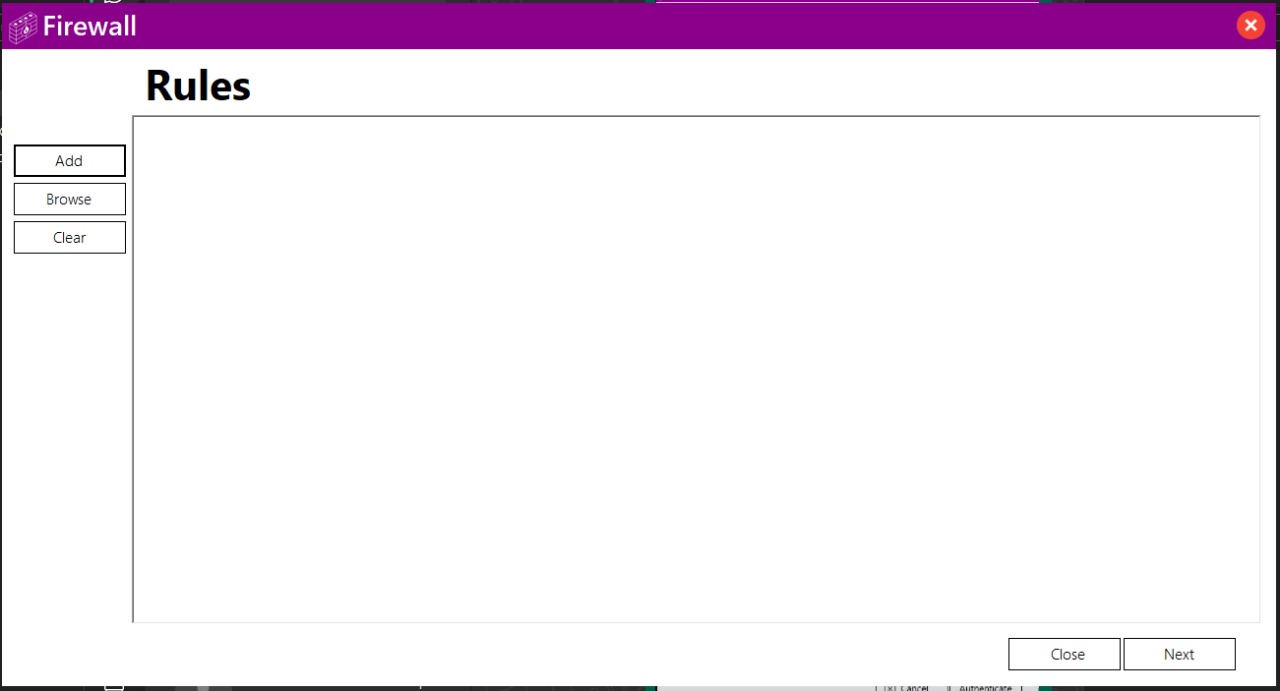
1. **Firewall Information:**

The “**Firewall Information**” page introduces a student-created basic firewall. It aims to provide effective network protection with a user-friendly interface. Users can customize rules based on source IP addresses, destination IP addresses, source ports, etc.



1. **Adding Rules:**

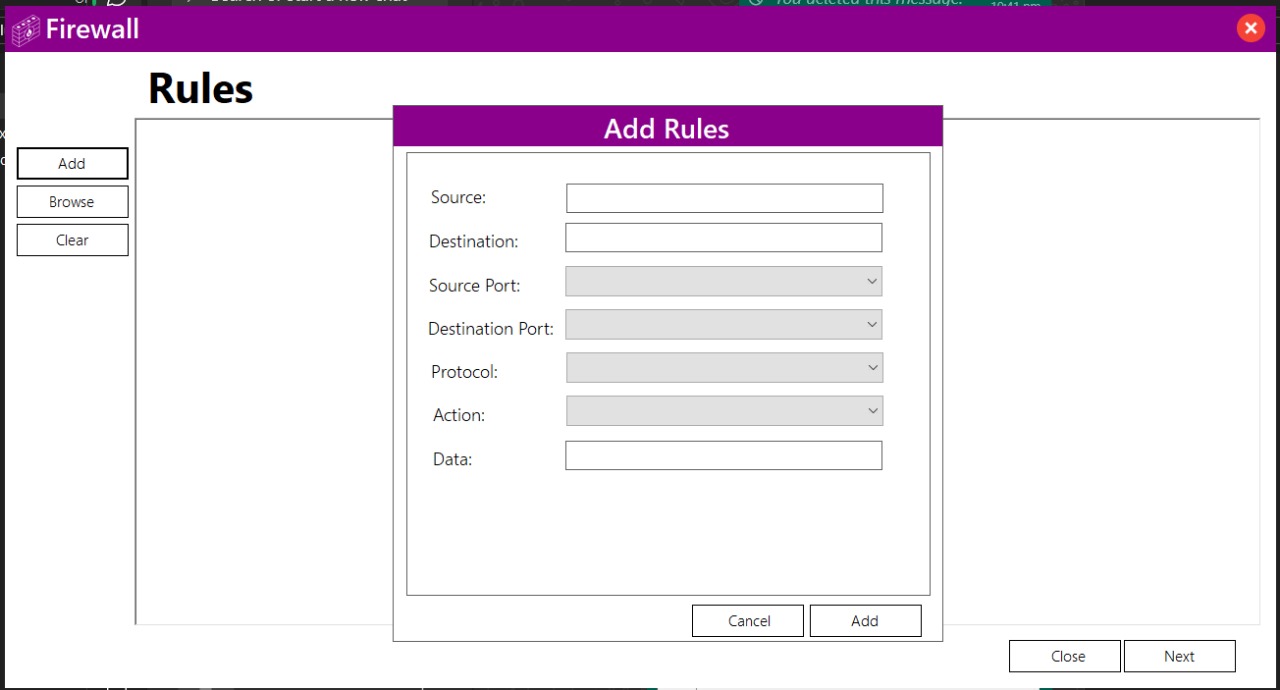
The "**Adding Rule**" interface features three buttons: "**Add**," "**Browse**," and "**Clear**." The "**Add**" button allows you to add new firewall rules, the "**Browse**" button lets you import rules from a file, and the "**Clear**" button removes all added rules.



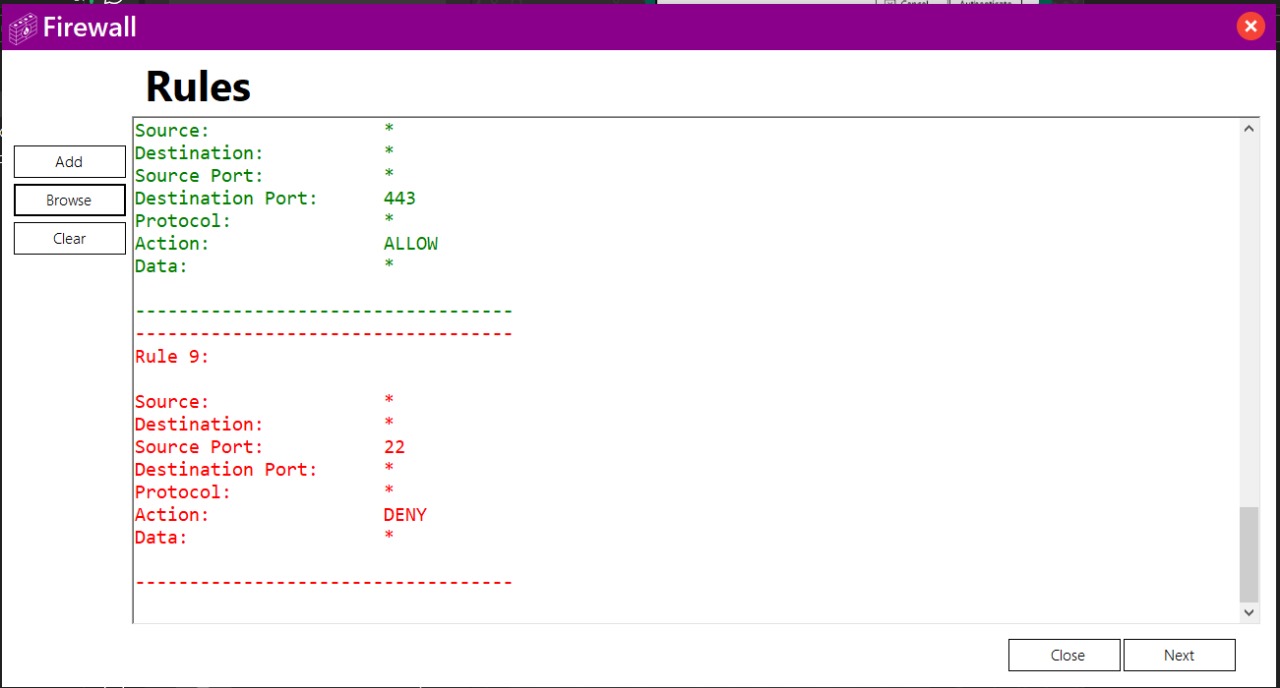
Upon clicking the "**Add**" button, a new panel appears for adding rules to the firewall. This panel includes the following fields:

* **Source:** Textbox to enter the source IP address.
* **Destination:** Textbox to enter the destination IP address.
* **Source Port:** Textbox to enter the source port.
* **Destination Port:** Textbox to enter the destination port.
* **Protocol:** Textbox to specify the protocol.
* **Action:** Textbox to define the action.
* **Data:** Textbox to input additional data.

Clicking the "**Add**" button will save the entered rule.

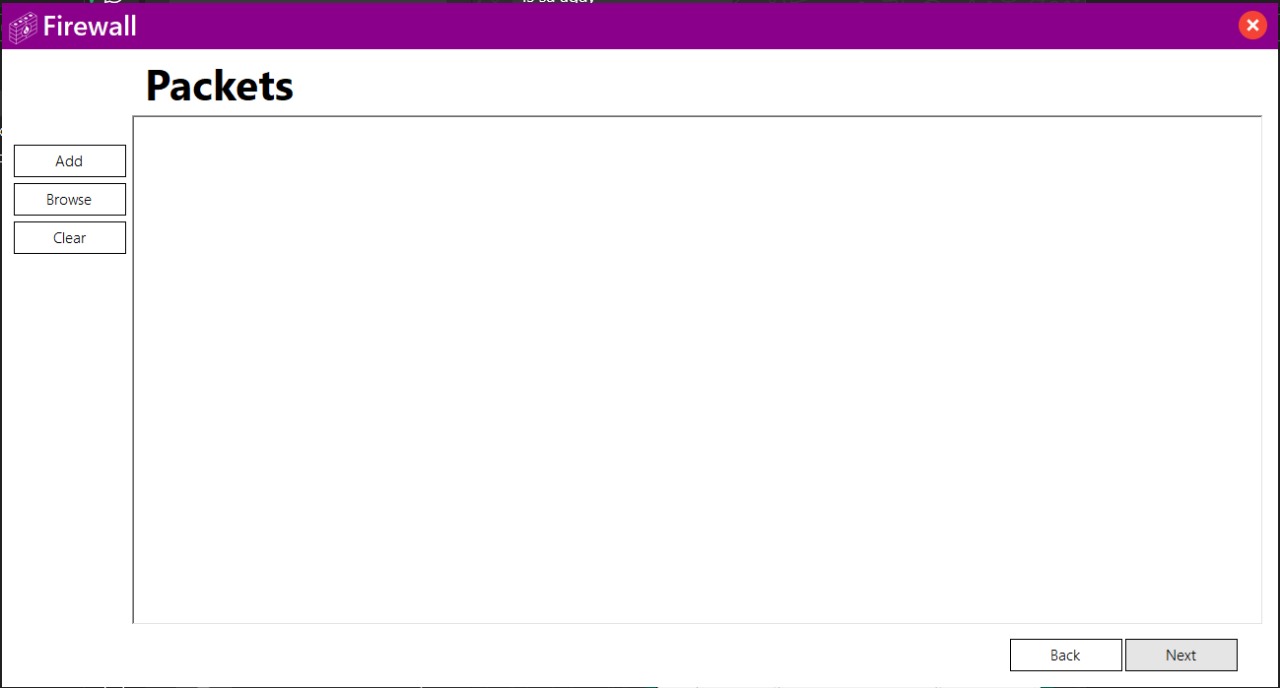


Rules were added



1. **Adding Packets:**

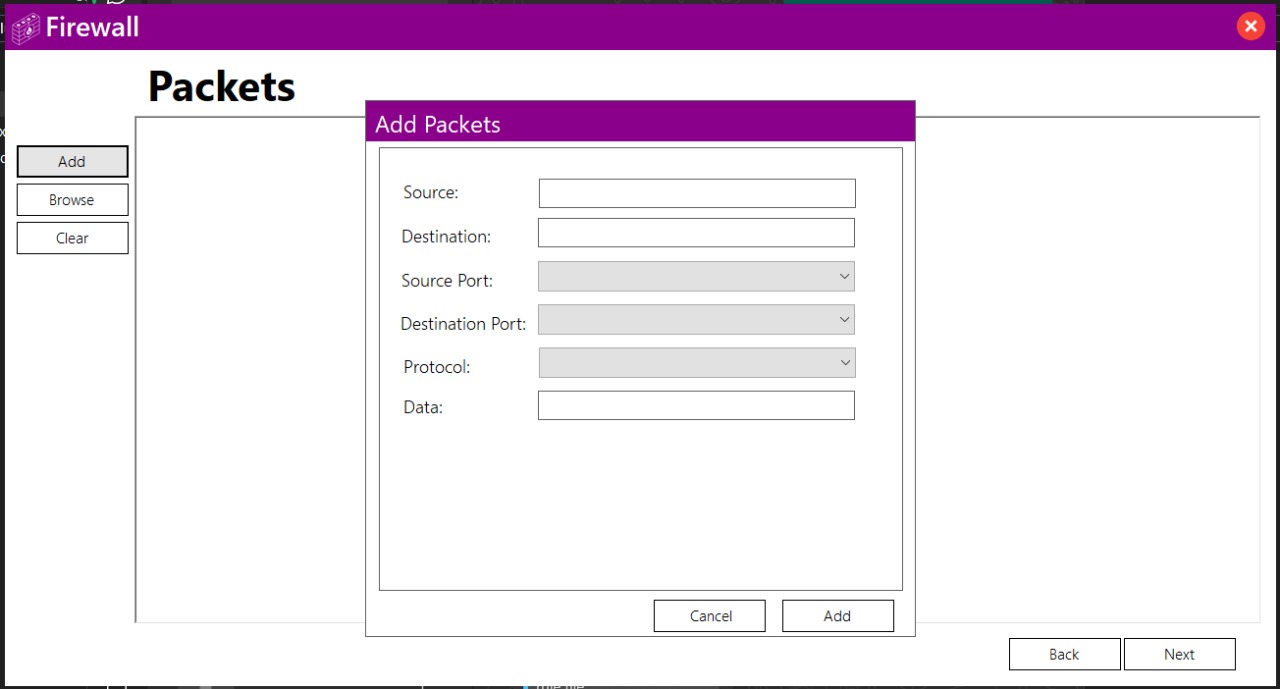
The "**Adding Packets**" interface features three buttons: "**Add**," "**Browse**," and "**Clear**." The "**Add**" button allows you to add new firewall Packets, the "**Browse**" button lets you import Packets from a file, and the "**Clear**" button removes all added Packets.



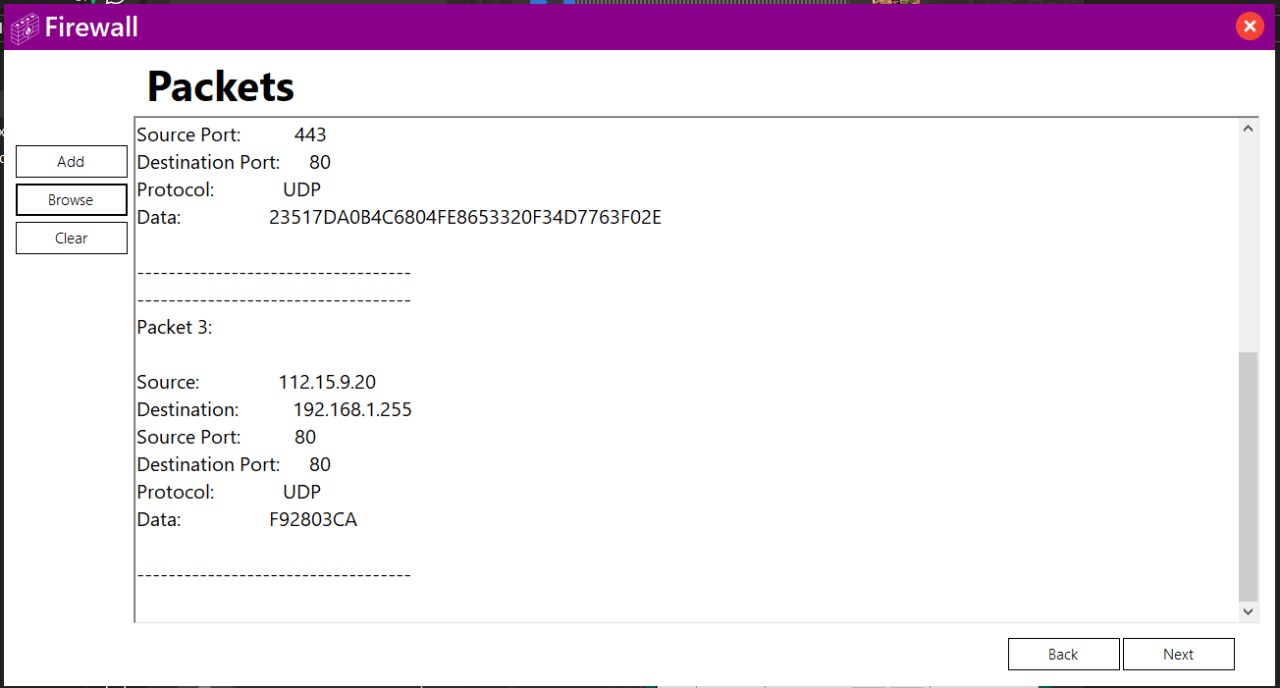
Upon clicking the "**Add**" button, a new panel appears for adding packets to the firewall. This panel includes the following fields:

* **Source:** Textbox to enter the source IP address.
* **Destination:** Textbox to enter the destination IP address.
* **Source Port:** Textbox to enter the source port.
* **Destination Port:** Textbox to enter the destination port.
* **Protocol:** Textbox to specify the protocol.
* **Data:** Textbox to input additional data.

Clicking the "**Add**" button will save the entered packets.



The added packets will be displayed on the screen.

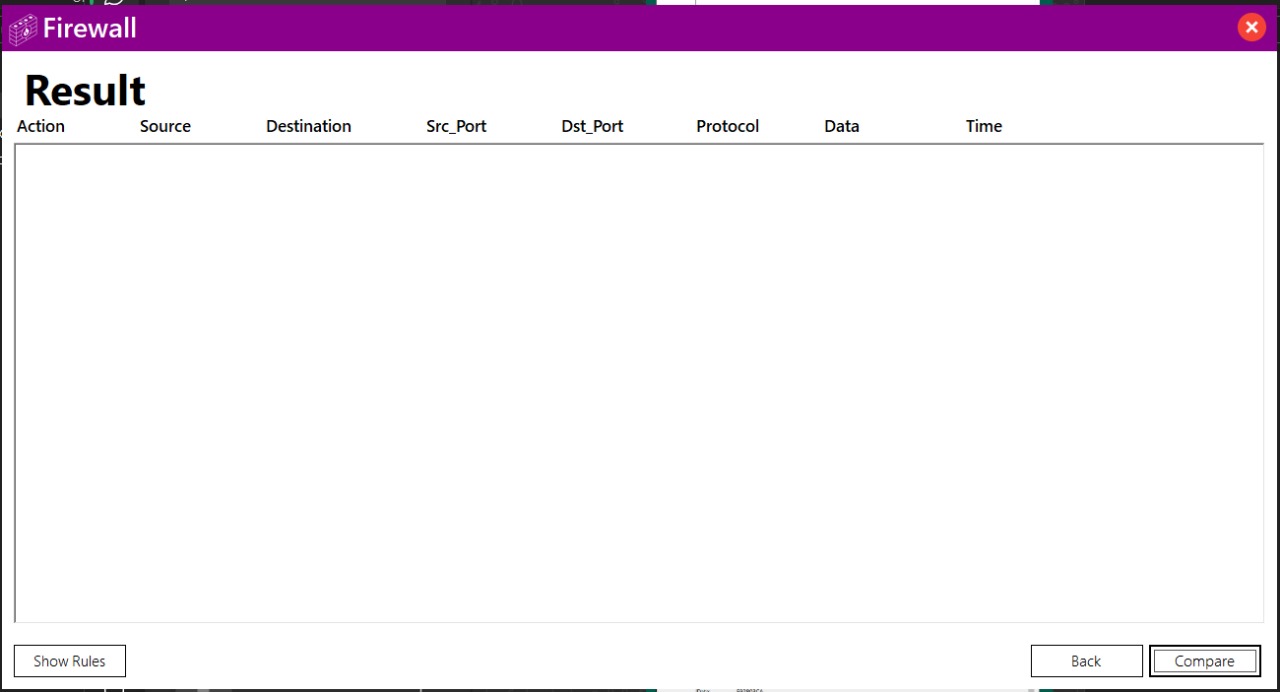


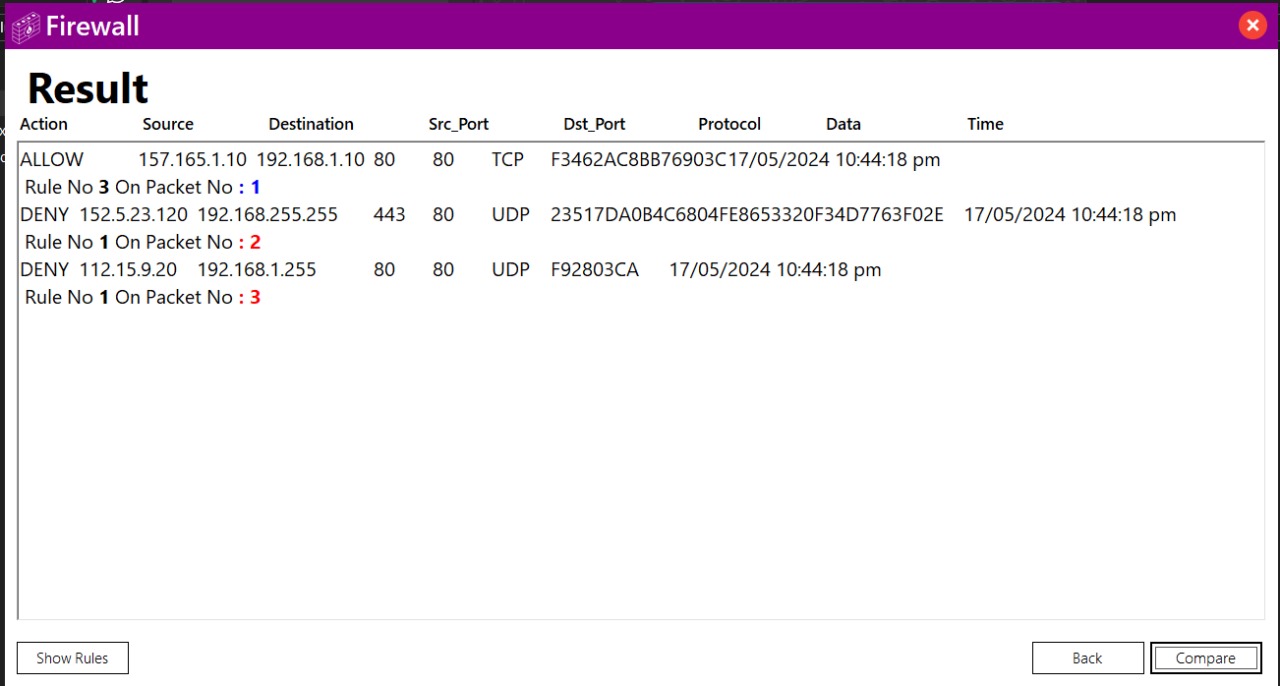
1. **Results:**

Result comparing Interface shows a table with the following columns:

* **Result:** This column shows whether a rule allowed or denied traffic.
* **Action:** This column describes the action taken such as allowing or denying traffic.
* **Source:** This column shows the IP address of the device that initiated the communication.
* **Destination:** This column shows the IP address of the device that received the communication.
* **Src Port:** This column shows the source port number. Ports are used to differentiate between different applications on a device.
* **Dst Port:** This column shows the destination port number.
* **Protocol:** This column shows the protocol used, such as TCP or UDP. TCP is a connection-oriented protocol, while UDP is connectionless.
* **Data:** This column likely refers to the data of packet.
* **Time:** This column shows the date and time the packet was added.

There is also a button labeled "**Show Rules**" and another button labeled "**Back**" in the bottom right corner.





**Testing:**

**->> Detail of testing process:**

**1.** Conduct trials with various passwords to test the login functionality.

**2**. Create various rule configurations for source and destination IP addresses, protocols, ports, and data types. Test combinations of allow and deny actions. Validate these rules by applying them to incoming network packets.

**3.** Evaluate the "Browse" button's response to incorrect file selections, such as adding a packet file instead of a rule file or vice versa, and assess the system's error handling in these scenarios.

**4.** Evaluate different rules with various packet types to ensure thorough coverage and effectiveness.

**Testing Process Details:**

**1. Types of Tests:**

* **Unit Tests:** Test individual components such as login functionality, rule creation, and packet handling independently.
* **Integration Tests:** Evaluate the seamless integration of login, rule configuration, and packet processing modules.

**2. Test Scenarios and Expected Outcomes:**

* **Login Functionality Test:**
* **Scenario:** Attempt login with correct password.
* **Expected Outcome:** Successful authentication and access to firewall settings.
* **Scenario:** Attempt login with incorrect password.
* **Expected Outcome:** Authentication failure, preventing access to firewall settings.
* **Rule Configuration Test:**
* **Scenario:** Create rules based on source IP addresses and specify allow action.
* **Expected Outcome:** Rules successfully applied to allow traffic from specified source IP addresses.
* **Scenario:** Define rules for destination IP addresses with deny action.
* **Expected Outcome:** Rules effectively block traffic to specified destination IP addresses.
* **Scenario:** Configure rules for specific protocols and ports, allowing certain types of traffic.
* **Expected Outcome:** Only traffic matching specified protocols and ports is permitted, while others are blocked.
* **Scenario:** Set rules based on data types to allow or deny specific types of data packets.
* **Expected Outcome:** Data packets are filtered according to specified rules, allowing or denying based on defined criteria.
* **Error Handling Test:**
* **Scenario:** Accidentally select a packet file instead of a rule file when defining rules.
* **Expected Outcome:** System should provide a clear error message indicating the incorrect file type and prompt the user to select the appropriate file.
* **Scenario:** Mistakenly upload a rule file instead of packet files when adding packets.
* **Expected Outcome:** Similar to above, the system should detect the incorrect file type and guide the user to upload the correct files.
* **Packet-Rule Interaction Test:**
* **Scenario:** Apply different rules to various types of incoming packets.
* **Expected Outcome:** Rules are correctly enforced, allowing or denying traffic based on specified criteria.
* **Scenario:** Test rule effectiveness by sending packets with different characteristics.
* **Expected Outcome:** Each packet is processed according to the corresponding rule, demonstrating the accuracy and reliability of the firewall's filtering capabilities.

**->> Test Results Report:**

**1. Login Functionality Test:**

* **Result:** Successful authentication with correct password. Authentication failure observed with incorrect password.
* **Comments:** Login functionality operates as expected, accurately restricting access with incorrect credentials.

**2. Rule Configuration Test:**

* **Result:** Rules based on source and destination IP addresses, protocols, ports, and data types were successfully created and applied.
* **Comments:** Rule configuration functionalities function as intended, effectively filtering traffic based on specified criteria.

**3. Error Handling Test:**

* **Result:** The system correctly detected and handled errors when incorrect file types were selected.
* **Comments:** Clear error messages were displayed, guiding users to select the appropriate file types.

**4. Packet-Rule Interaction Test:**

* **Result:** Rules were successfully enforced, allowing or denying traffic based on defined criteria.
* **Comments:** The firewall effectively processed incoming packets according to the rules, demonstrating its filtering capabilities.

**Bug Encounters:**

1. **Error in Rule Application:**

* **Issue:** In some cases, rules were not properly applied to incoming packets.
* **Resolution:** After debugging, the issue was traced to an error in the rule processing algorithm, which was fixed to ensure consistent rule enforcement.

1. **Unexpected Behavior with Rule Imports:**

* **Issue:** Upon importing rule files, unexpected behavior was observed, leading to incorrect rule configurations.
* **Resolution:** Further investigation revealed inconsistencies in file parsing. The issue was resolved by refining the algorithm to accurately interpret rule configurations.

1. **Inconsistency in Error Messaging:**

* **Issue:** Error messages displayed upon selecting incorrect file types were inconsistent across different interfaces.
* **Resolution:** To improve user experience, error messages were standardized across all interfaces, offering clear guidance on file selection errors.

**Challenges Faced:**

**->> Identify challenges encountered during the project:**

1. **Data Transfer between Window Forms:**

* **Description:** Difficulty was encountered in transferring data between different window forms within the application.
* **Impact:** This challenge hindered the seamless flow of information between various stages of the firewall configuration process.

1. **Text Formatting in Result File:**

* **Description:** Formatting issues arose when generating result files, impacting the readability and presentation of the data.
* **Impact:** Poor text formatting made the result files difficult for users to interpret accurately.

**->> Strategies for overcoming challenges:**

**1. Data Transfer between Window Forms:**

* **Resolution:** Resolved through debugging and implementing effective data transfer methods like using global variables or passing parameters between forms.

1. **Text Formatting in Result File:**

* **Resolution**: Resolved by implementing text formatting techniques and user-friendly interfaces to ensure data readability and customization of formatting preferences.

**Lessons Learned:**

**->> Reflecting on lessons learned from the project:**

* **Insights into C++ OOP Principles:**
* The project deepened my grasp of C++ OOP principles, including classes, inheritance, encapsulation, and polymorphism. Implementing these concepts improved code organization and reusability.
* **Understanding of Firewall Functionality and Implementation:**
* The project offered valuable insights into firewall functionality, including rule-based filtering and packet inspection. This understanding empowered me to develop effective strategies for configuring and managing firewall rules to secure network traffic.
* **GUI Development in C++:**
* I also gained experience in GUI development in C++, creating user-friendly interfaces that seamlessly interact with the firewall application. This expanded my skills in integrating intuitive graphical interfaces with underlying functionality.

**Suggestions for Future Improvements:**

* Propose enhancements or additional features for the firewall application.
* Explore real-world firewall deployment scenarios to better understand industry-standard practices and requirements.
* Gain insights into networking protocols and configurations commonly used in firewall deployments, ensuring alignment with industry standards and best practices.

**Analysis of Learning:**

The firewall project provided valuable insights into C++ OOP principles, particularly through the implementation of a graphical user interface (GUI). By designing intuitive interfaces and integrating them with underlying functionality, I learned how to apply OOP concepts like classes and inheritance to create user-friendly applications. This practical experience not only enhanced my understanding of OOP but also demonstrated its relevance in developing interactive and efficient software solutions for real-world scenarios.