BASIC FIREWALL PROJECT REPORT

Created by: Bhargav Raj Dutta

Date: 09/12/2024

Executive Summary

The Basic Firewall Project simulates a rule-based firewall using Python, showcasing how network traffic is controlled based on predefined rules. The project includes a console-based application, a graphical user interface, and a logging mechanism to record all actions. This project aims to enhance understanding of access control and rule enforcement in cybersecurity.

Objectives

1. Implement a basic firewall to simulate traffic filtering.
2. Use a JSON-based rules system for dynamic management.

3. Create an intuitive graphical interface for rule modification.

4. Log all traffic actions for auditing purposes.

5. Understand the importance of rule-based access control in network security.

Scope

This project focuses on:

- Simulating rule-based traffic filtering.
- Developing an interactive GUI for rule management.
- Generating random IP traffic for testing.
- Recording and analyzing traffic actions.

Features

Key Functionalities

- **Rule-Based Access Control:** Blocks or allows traffic based on predefined rules stored in a JSON file.

- **Traffic Simulation:** Generates random traffic for testing firewall behavior.
- **Logging:** Records all actions (block/allow) in a log file for audit purposes.
- **GUI Management:** Enables visualization and dynamic management of firewall rules.
- **Extensibility:** Modular design supports future enhancements like wildcard rules and real-time data integration.

Directory Structure

```
basic_firewall/
 ├── firewall.py         # Main script for console-based simulation
 ├── firewall_gui.py     # GUI version of the firewall
 ├── rules. json         # JSON file to store firewall rules
├── traffic.log          # Log file for traffic actions (auto created)
└── traffic_generator.py # Optional traffic generation module
```

File Descriptions

1. firewall.py: Handles rule enforcement and logs actions for simulated traffic.
2. firewall_gui.py: Provides a graphical interface for managing firewall rules.
3. rules. json: Stores IP rules in a structured format.
4. traffic.log: Captures details of all actions for audit purposes.
5. traffic_generator.py: Optionally generates random traffic for testing.

Firewall Rules

The rules.json file contains IP addresses and corresponding actions (block or allow).

Example:

```
{
   "192.168.1.1": "block",
   "192.168.1.4": "block",
   "192.168.1.9": "block"
   "192.168.1.13": "block",
   "192.168.1.16": "block",
   "192.168.1.19": "block"


}
```

Explanation

Key: Represents the IP address.

Value: Specifies the action (block or allow).

Detailed Implementation

Console-Based Simulation

Load Rules: Load firewall rules from rules.json using Python's json module.

Generate Traffic: Simulate random IP traffic using Python's random.randint().

Check Rules: Compare each IP against the rules.

If Matched: Apply the specified action (block or allow).

If Not Matched: Default to allow.

Log Actions: Write the results to traffic.log.

Graphical Interface

GUI Framework: Built using Python's tkinter library.

Interactive Rules Management:

- Add new rules dynamically.
- Remove or modify existing rules.

- Save updates back to rules.json.

Traffic Simulation: Simulate IP inputs directly in the GUI.

How to Set Up

Pre-Requisites

Python Version: Python 3.6 or higher.

Libraries: Ensure tkinter is installed (default in Python for most platforms).

Installation Steps

Download the project files to your system.

Open a terminal or command prompt.

Navigate to the project directory:

cd/Users/bhargavrajdutta/Desktop/
basic_firewall

(Optional) Create a virtual environment:

python3 -m venv env source env/bin/activate

Running the Project

Console Simulation

Run the command:

```
python3 firewall.py
```

GUI Application

Launch the graphical interface:

```
python3 firewall_gui.py
```

Traffic Generator

Optionally, generate traffic logs:

```
python3 traffic_generator.py
```

Analyze Logs

View the log file for traffic actions:

```
cat traffic.log
```

Logging

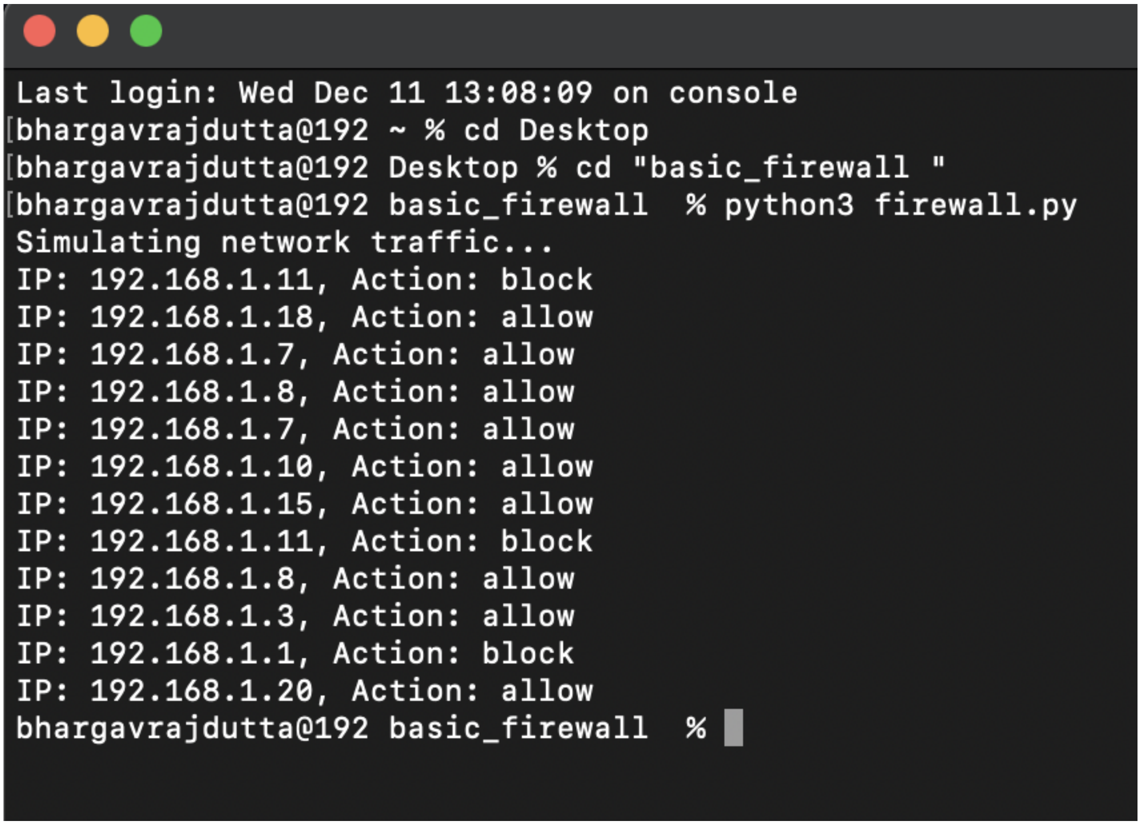
The traffic.log file captures details of all traffic actions.
Example entries:

```
2024-12-07 00:42:25 - IP: 192.168.1.19, Action: block
2024-12-07 00:42:25 - IP: 192.168.1.8, Action: allow
```

Log Format

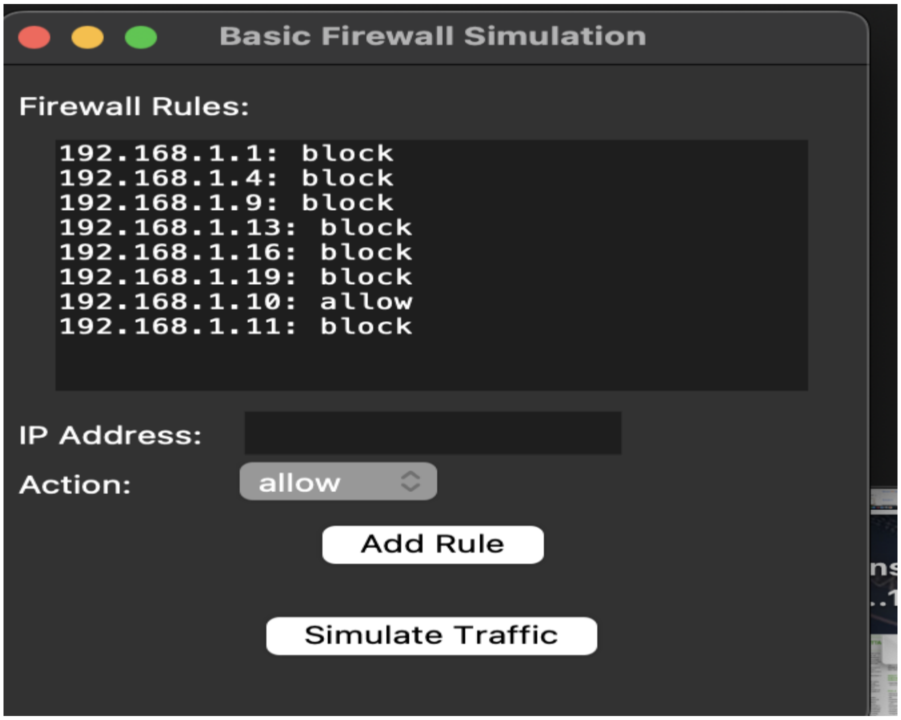
Timestamp: Records when the action occurred.
IP: Displays the simulated IP address.
Action: Indicates whether the traffic was blocked or allowed.

Console Output



```
 Last login: Wed Dec 11 13:08:09 on console
[bhargavrajdutta@192 ~ % cd Desktop
[bhargavrajdutta@192 Desktop % cd "basic_firewall "
[bhargavrajdutta@192 basic_firewall  % python3 firewall.py
 Simulating network traffic...
 IP: 192.168.1.11, Action: block
 IP: 192.168.1.18, Action: allow
 IP: 192.168.1.7, Action: allow
 IP: 192.168.1.8, Action: allow
 IP: 192.168.1.7, Action: allow
 IP: 192.168.1.10, Action: allow
 IP: 192.168.1.15, Action: allow
 IP: 192.168.1.11, Action: block
 IP: 192.168.1.8, Action: allow
 IP: 192.168.1.3, Action: allow
 IP: 192.168.1.1, Action: block
 IP: 192.168.1.20, Action: allow
 bhargavrajdutta@192 basic_firewall  %
```

GUI Demonstration



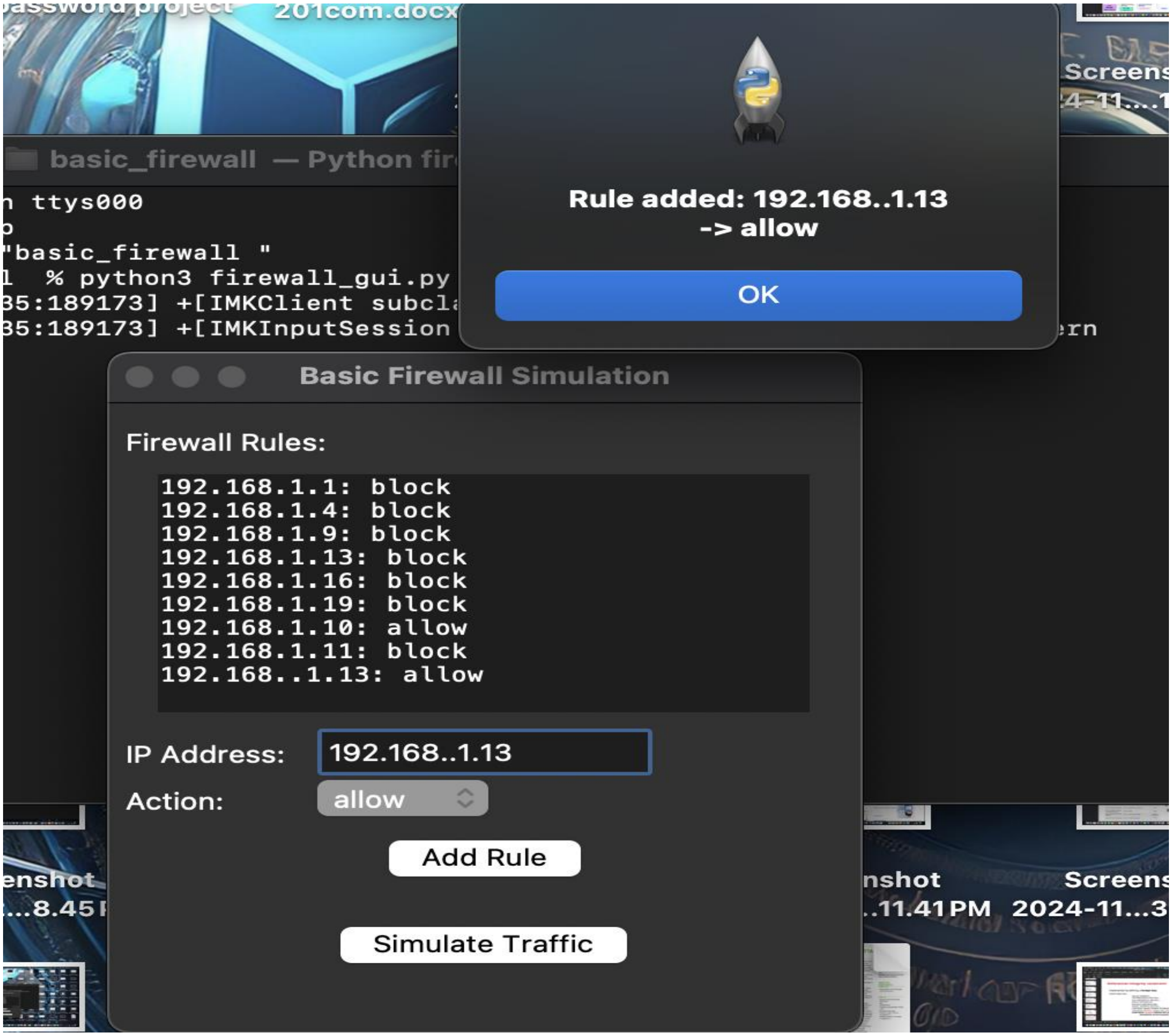
Firewall Rules:

```
192.168.1.1: block
192.168.1.4: block
192.168.1.9: block
192.168.1.13: block
192.168.1.16: block
192.168.1.19: block
192.168.1.10: allow
192.168.1.11: block
```

IP Address:

Action: allow

The GUI allows users to:
View current rules.
Add new rules for IP filtering.
Save changes back to the rules file.



```
n ttys000
p
"basic_firewall "
l  % python3 firewall_gui.py
35:189173] +[IMKClient subcla
35:189173] +[IMKInputSession                                          ern
```

Rule added: 192.168..1.13 -> allow

Basic Firewall Simulation

Firewall Rules:

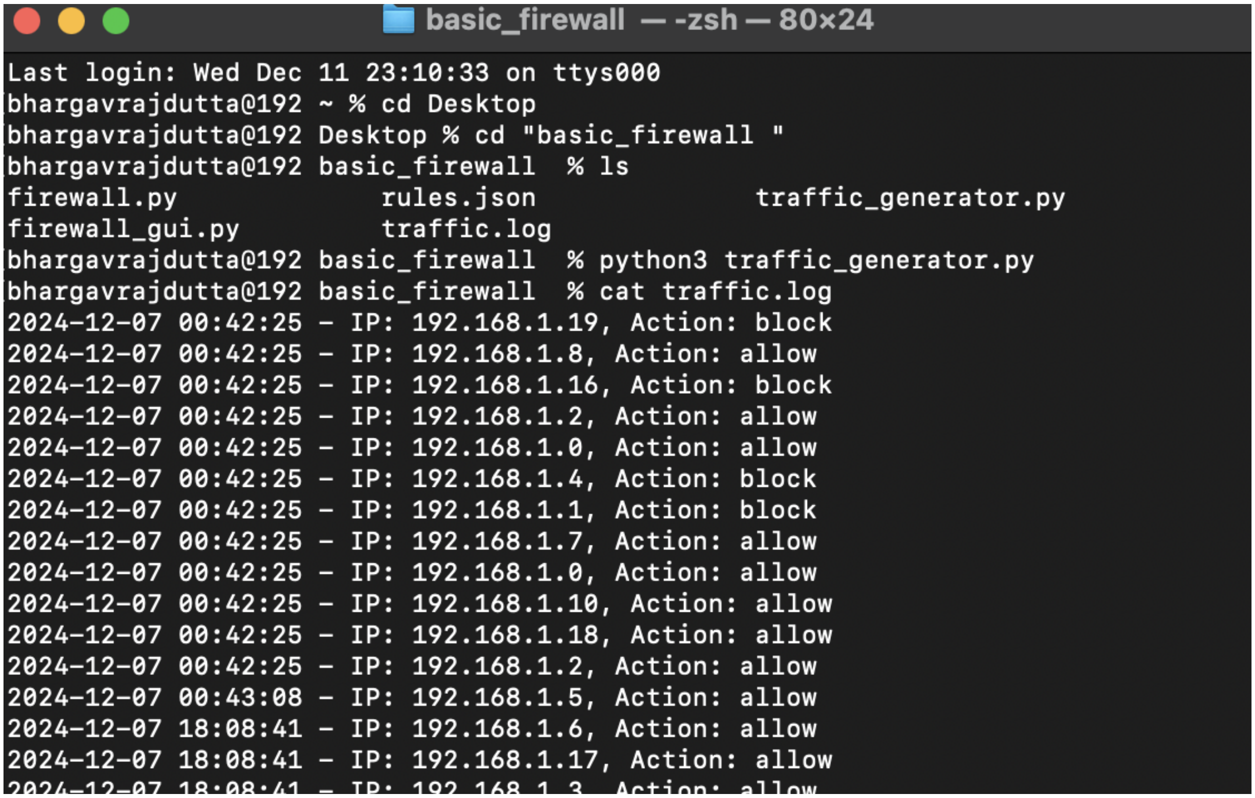
```
192.168.1.1: block
192.168.1.4: block
192.168.1.9: block
192.168.1.13: block
192.168.1.16: block
192.168.1.19: block
192.168.1.10: allow
192.168.1.11: block
192.168..1.13: allow
```

IP Address: 192.168..1.13

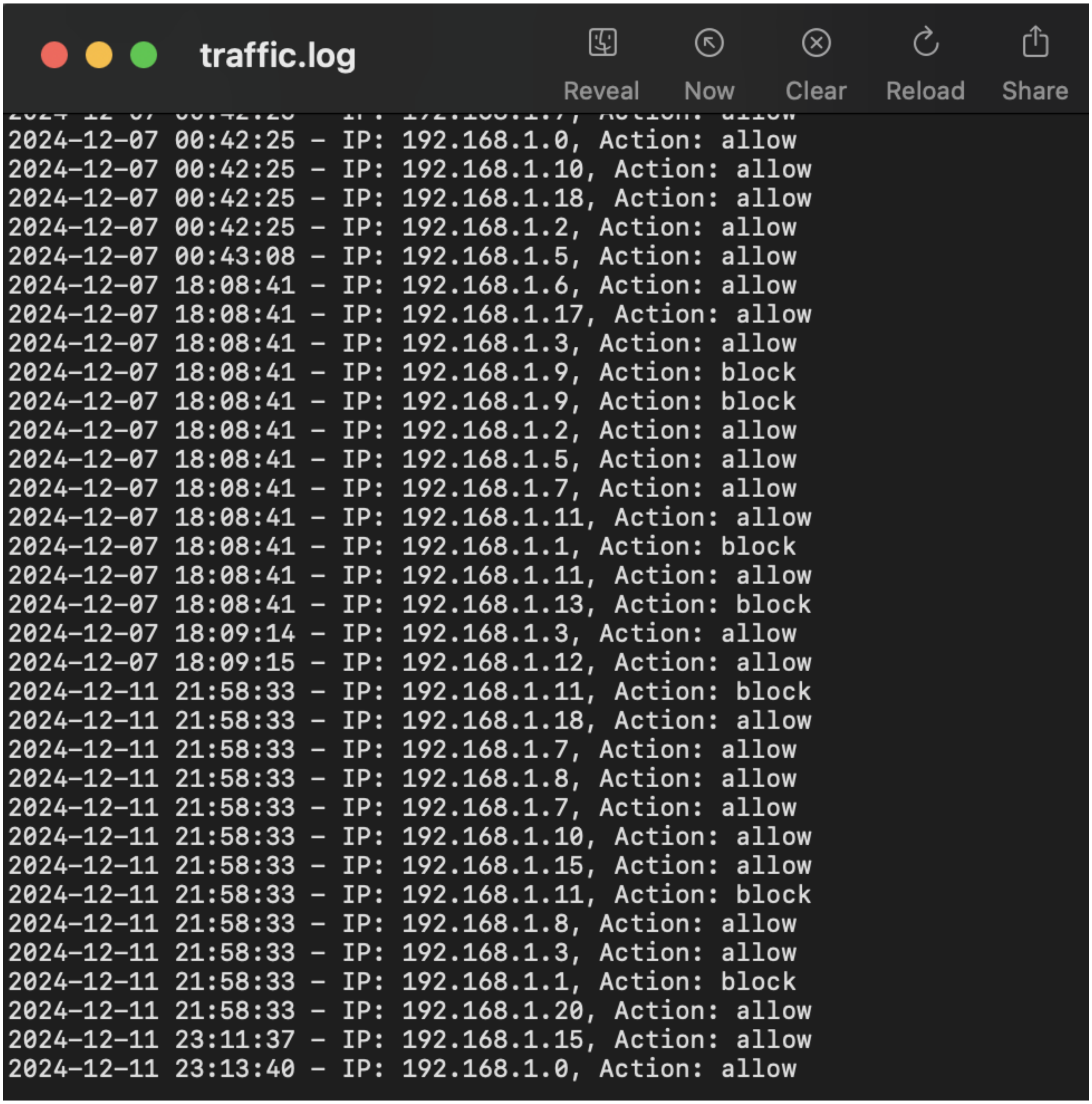
Action: allow

Log Analysis

Logs provide a detailed audit trail for all actions:



```
Last login: Wed Dec 11 23:10:33 on ttys000
bhargavrajdutta@192 ~ % cd Desktop
bhargavrajdutta@192 Desktop % cd "basic_firewall "
bhargavrajdutta@192 basic_firewall  % ls
firewall.py             rules.json              traffic_generator.py
firewall_gui.py         traffic.log
bhargavrajdutta@192 basic_firewall  % python3 traffic_generator.py
bhargavrajdutta@192 basic_firewall  % cat traffic.log
2024-12-07 00:42:25 - IP: 192.168.1.19, Action: block
2024-12-07 00:42:25 - IP: 192.168.1.8, Action: allow
2024-12-07 00:42:25 - IP: 192.168.1.16, Action: block
2024-12-07 00:42:25 - IP: 192.168.1.2, Action: allow
2024-12-07 00:42:25 - IP: 192.168.1.0, Action: allow
2024-12-07 00:42:25 - IP: 192.168.1.4, Action: block
2024-12-07 00:42:25 - IP: 192.168.1.1, Action: block
2024-12-07 00:42:25 - IP: 192.168.1.7, Action: allow
2024-12-07 00:42:25 - IP: 192.168.1.0, Action: allow
2024-12-07 00:42:25 - IP: 192.168.1.10, Action: allow
2024-12-07 00:42:25 - IP: 192.168.1.18, Action: allow
2024-12-07 00:42:25 - IP: 192.168.1.2, Action: allow
2024-12-07 00:43:08 - IP: 192.168.1.5, Action: allow
2024-12-07 18:08:41 - IP: 192.168.1.6, Action: allow
2024-12-07 18:08:41 - IP: 192.168.1.17, Action: allow
2024-12-07 18:08:41 - IP: 192.168.1.3, Action: allow
```



```
2024-12-07 00:42:25 - IP: 192.168.1.7, Action: allow
2024-12-07 00:42:25 - IP: 192.168.1.0, Action: allow
2024-12-07 00:42:25 - IP: 192.168.1.10, Action: allow
2024-12-07 00:42:25 - IP: 192.168.1.18, Action: allow
2024-12-07 00:42:25 - IP: 192.168.1.2, Action: allow
2024-12-07 00:43:08 - IP: 192.168.1.5, Action: allow
2024-12-07 18:08:41 - IP: 192.168.1.6, Action: allow
2024-12-07 18:08:41 - IP: 192.168.1.17, Action: allow
2024-12-07 18:08:41 - IP: 192.168.1.3, Action: allow
2024-12-07 18:08:41 - IP: 192.168.1.9, Action: block
2024-12-07 18:08:41 - IP: 192.168.1.9, Action: block
2024-12-07 18:08:41 - IP: 192.168.1.2, Action: allow
2024-12-07 18:08:41 - IP: 192.168.1.5, Action: allow
2024-12-07 18:08:41 - IP: 192.168.1.7, Action: allow
2024-12-07 18:08:41 - IP: 192.168.1.11, Action: allow
2024-12-07 18:08:41 - IP: 192.168.1.1, Action: block
2024-12-07 18:08:41 - IP: 192.168.1.11, Action: allow
2024-12-07 18:08:41 - IP: 192.168.1.13, Action: block
2024-12-07 18:09:14 - IP: 192.168.1.3, Action: allow
2024-12-07 18:09:15 - IP: 192.168.1.12, Action: allow
2024-12-11 21:58:33 - IP: 192.168.1.11, Action: block
2024-12-11 21:58:33 - IP: 192.168.1.18, Action: allow
2024-12-11 21:58:33 - IP: 192.168.1.7, Action: allow
2024-12-11 21:58:33 - IP: 192.168.1.8, Action: allow
2024-12-11 21:58:33 - IP: 192.168.1.7, Action: allow
2024-12-11 21:58:33 - IP: 192.168.1.10, Action: allow
2024-12-11 21:58:33 - IP: 192.168.1.15, Action: allow
2024-12-11 21:58:33 - IP: 192.168.1.11, Action: block
2024-12-11 21:58:33 - IP: 192.168.1.8, Action: allow
2024-12-11 21:58:33 - IP: 192.168.1.3, Action: allow
2024-12-11 21:58:33 - IP: 192.168.1.1, Action: block
2024-12-11 21:58:33 - IP: 192.168.1.20, Action: allow
2024-12-11 23:11:37 - IP: 192.168.1.15, Action: allow
2024-12-11 23:13:40 - IP: 192.168.1.0, Action: allow
```

Observations

The project successfully enforces predefined rules.

Logs provide transparency and traceability for all actions.

The GUI allows intuitive rule management.

Challenges

Dynamic Rule Updates: Ensuring immediate effect of rule changes.

Scalability: Handling larger datasets or more complex rules.

GUI Performance: Optimizing for better responsiveness.

Future Enhancements

Pattern Matching: Support for wildcard rules (e.g., 192.168.*).

Real-Time Traffic: Integrate with live network data.

Threat Intelligence: Use external sources to block malicious IPs.

Conclusion

The Basic Firewall Project demonstrates a rule-based traffic filtering system, combining both theoretical and practical knowledge of cybersecurity. Its modular design and GUI make it a versatile learning tool.