**Personal Firewall Using Python(Scapy + GUI Simulation + Packet Filtering**)

**Introduction:**

A **Personal Firewall** monitors incoming/outgoing network traffic and applies custom rules to allow or block packets. It acts as a protective layer for users by inspecting packets in real time and logging any suspicious activities.

### Abstract:

This project uses **Python and Scapy** to sniff packets, filter them based on IP, port, and protocol rules, and optionally interact with Linux iptables for real-time blocking. A basic Tkinter GUI or CLI provides live rule management and logs abnormal packets.

## Scope of the Project:

* Develop a **user-defined rule-based personal firewall**.
* Real-time **packet sniffing and traffic monitoring** using Scapy.
* Detect **malicious IP addresses, blocked ports, and specific protocols (e.g., ICMP, Telnet)**.
* Log blocked packets into **local log files** for audit purposes.
* Provide a **GUI or console-based simulation** to visualize firewall activities.
* This project is designed for **educational and research purposes** to simulate firewall behavior at the system level.

**Tools Used:**

| **Tool** | Purpose |
| --- | --- |
| **Python** | Core logic |
| **scapy** | Packet sniffing and filtering |
| **iptables** | (Optional) System-level rule enforcement |
| **Tkinter** | GUI for live monitoring (optional) |
| **logging** | Logging blocked/suspicious packets |

**Major Script Functions & Explanation:**

| **Function Name** | **Purpose** |
| --- | --- |
| **Filter\_Packet(packet)** | Checks whether the incoming packet matches any of the firewall block rules. |
| **Log\_Block(packet)** | Logs blocked packet details into a .txt file and maintains a list for live display. |
| **Packet\_Callback(packet)** | Callback for Scapy's sniff function to process each incoming packet. |
| **Start\_Sniffing()** | Initiates Scapy's packet sniffing in a separate background thread. |
| **Display\_Status()** | Refreshes the console/GUI to show live blocked packets using clear\_output() in Jupyter. |
| **Apply\_Iptables()** | (Optional) Adds IP block and port filtering rules at the OS-level using Linux iptables. |

## Attack Scenarios Covered:

| **Attack Type** | **Detection Method** |
| --- | --- |
| **IP-based Blocking** | Any packet originating from blacklisted IPs is detected and logged as blocked. |
| **Port-based Blocking** | Traffic targeting sensitive ports (e.g., Telnet 23, SMB 445) is flagged & blocked. |
| **Protocol-based Blocking** | ICMP traffic (used in Ping floods) can be monitored and logged as malicious. |
| **Simulated XSS/SQLi** | (In extended modules) could integrate payload detection by inspecting packet payloads. |

## Expected Output:

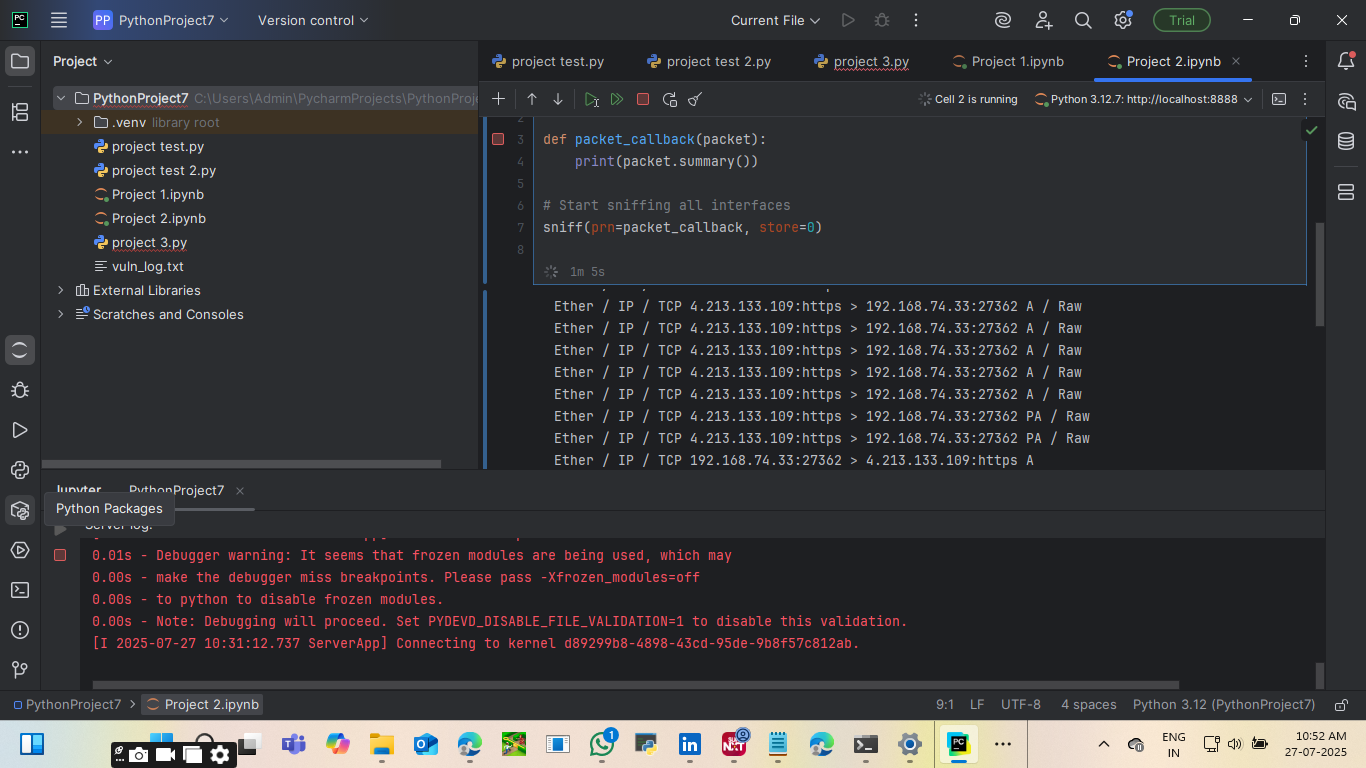
* Console/Jupyter will display:

Firewall is Running... Logs written to firewall\_log.txt

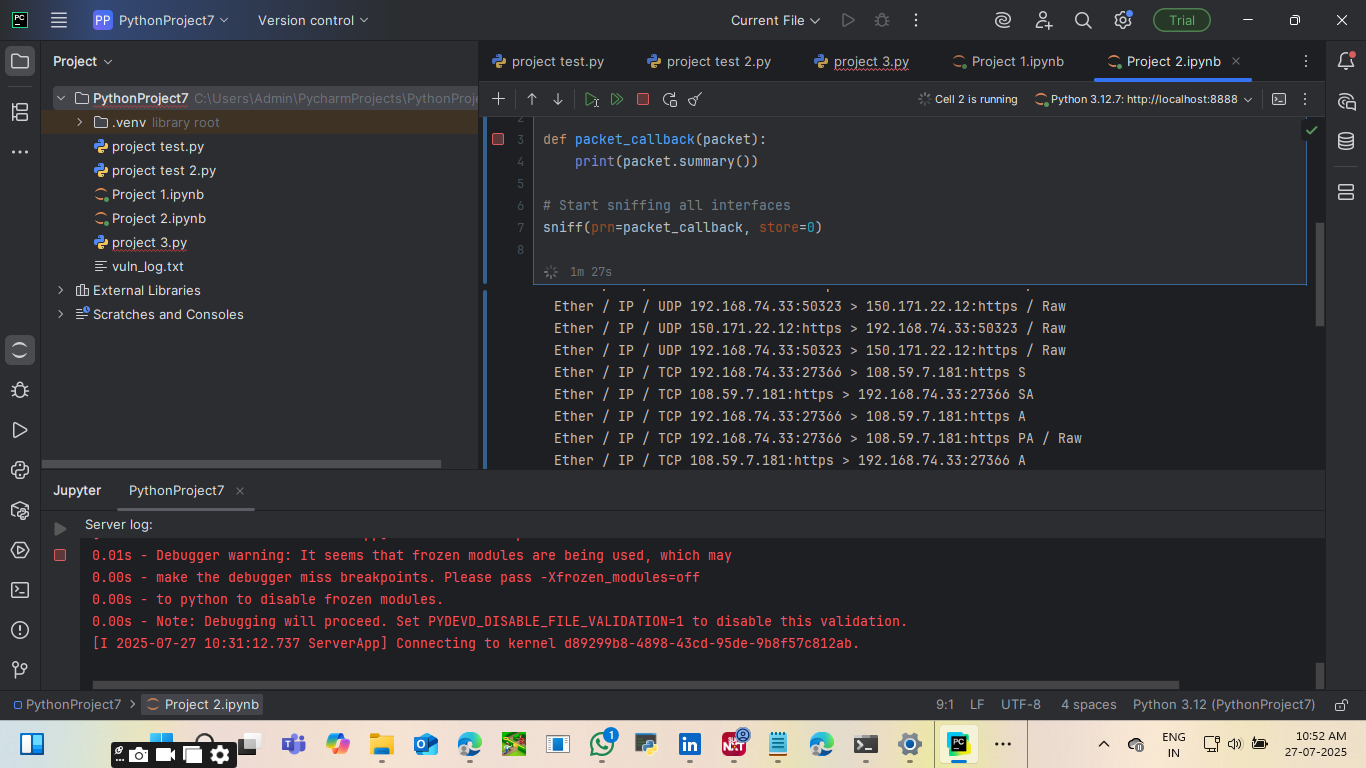
Blocked Packets (Live View):

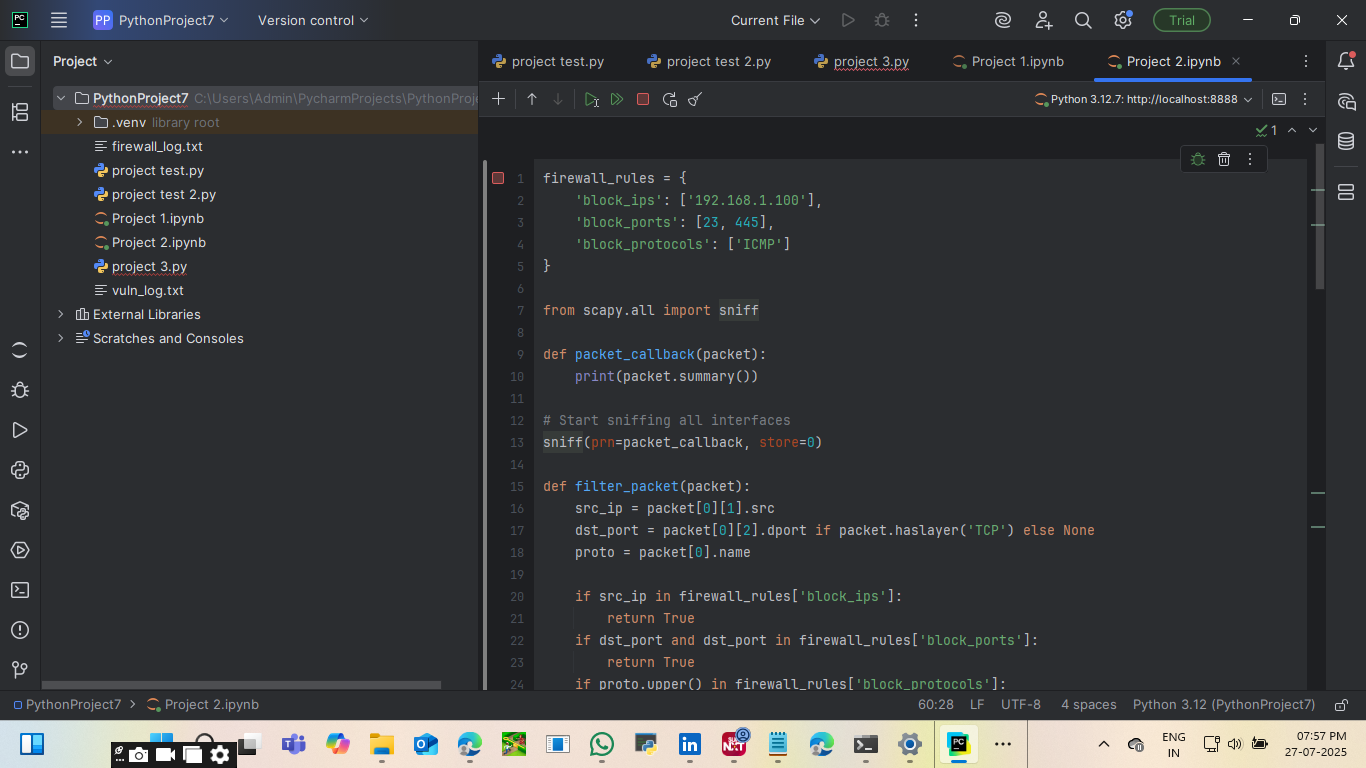
[BLOCKED] IP / TCP 192.168.1.100:445 → 192.168.1.101:445

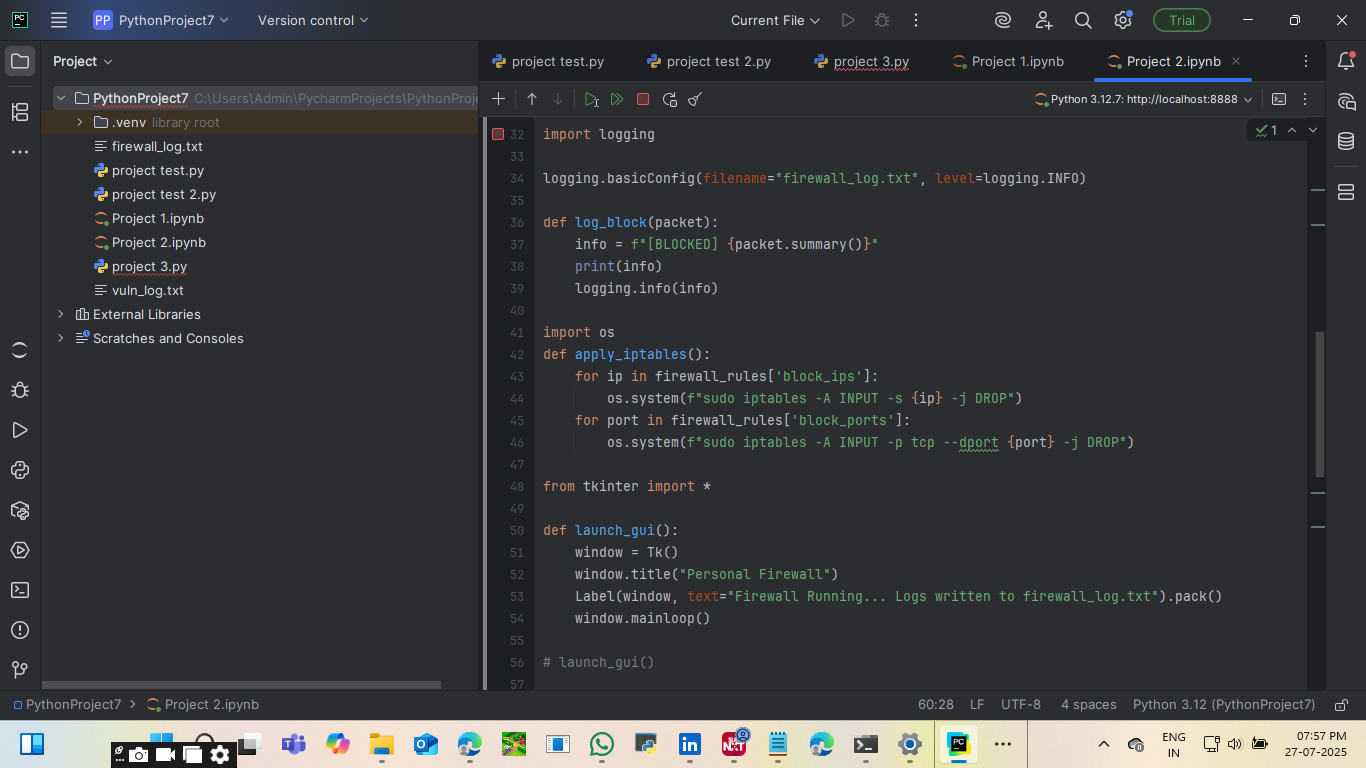
[BLOCKED] IP / ICMP Echo Request 192.168.1.100 → 192.168.1.101



**Sniffed Packets :**







 All blocked packets will be saved in **firewall\_log.txt**.

 The system will continuously monitor packets until terminated.

 Visual feedback (GUI/console refresh) will simulate real-time firewall actions.

## ****Conclusion:****

This Python-based Personal Firewall demonstrates the core principles of **network traffic monitoring and basic intrusion prevention** using custom rules. Although it lacks OS-level enforcement like enterprise-grade firewalls, it effectively showcases how **IP/Port/Protocol filtering, logging, and live monitoring** work at a conceptual level. This project serves as a learning platform for students and cybersecurity enthusiasts to understand packet inspection and firewall logic.

## **Future Scope:**

* Extend detection to **application-layer attacks** (XSS, SQL Injection) by inspecting HTTP payloads.
* Integrate with **Linux iptables dynamically (auto-block detected IPs)**.
* Develop a **full-fledged GUI dashboard** with real-time charts and logs.
* Implement **alerting mechanisms (Email/SMS notifications)** for critical attacks.
* Support **Outbound Traffic Filtering (Egress Filtering)**.
* Add **stateful inspection** and maintain connection states.
* Evolve into a **HIDS (Host-based Intrusion Detection System)** with machine learning anomaly detection.