Personal Firewall Using Python

Submitted by:

Miraclin Akshaya
Organization: Elevate Labs
Date of Submission: October 27, 2025

1. Objective

The objective of this project is to develop a lightweight **personal firewall** that can monitor, analyze, and control incoming and outgoing network traffic using user-defined rules.
The project aims to give users greater visibility into how data packets move through their system and enable them to filter packets based on criteria such as IP addresses, ports, and protocols.

This project demonstrates fundamental concepts of **packet filtering**, **network monitoring**, and **rule-based decision-making**, which form the foundation of cybersecurity and intrusion prevention systems.

2. Tools and Technologies Used

- **Programming Language:** Python 3
- **Libraries:**
  - Scapy — used for sniffing, analyzing, and filtering packets at the user level
  - JSON — for defining firewall rule sets (blocked IPs, ports, and protocols)
- **Operating System:** Kali Linux (Debian-based)
- **Optional Tools:** iptables (for system-level enforcement, not implemented here)

This setup offers portability, ease of debugging, and clarity in understanding packet-level control without requiring root-level kernel operations.

3. System Design and Architecture

The personal firewall operates in **user space**, capturing packets via the Scapy library and applying rule checks against the predefined configuration file rules.json.
Each packet is analyzed in real time and categorized as **ALLOWED** or **BLOCKED** based on the defined filtering logic.

3.1 Core Components

- **Packet Sniffer:** Captures live packets using scapy.sniff()
- **Rule Engine:** Evaluates each packet against conditions defined in rules.json
- **Logger:** Prints packet activity to the console, showing decisions and timestamps

- **Rule Configuration File (rules.json):**

```
{
  "blocked_ips": ["8.8.8.8"],
  "blocked_ports": [80],
  "block_protocols": ["ICMP"],
  "enforce_with_iptables": false
}
```

This allows flexible customization without editing the Python code.

4. Implementation

The firewall was developed as a **command-line interface (CLI)** program using Python and Scapy.
It works in real time, sniffing packets and filtering them dynamically.

Steps Involved

1. **Environment Setup**
   - Installed python3 and scapy in the Kali Linux environment.
   - Created a project directory and initialized rules.json with blocking criteria.
2. **Packet Capture**
   - Used scapy.sniff() to intercept incoming and outgoing packets.
   - Extracted source and destination IPs, protocol, and ports for analysis.
3. **Rule Evaluation**
   - Compared each packet's attributes against the JSON ruleset.
   - If any parameter matched a blocked rule, the packet was labeled **BLOCKED**; otherwise, **ALLOWED**.
4. **Real-Time Logging**
   - Printed packet data with timestamps, status, and reasons for blocking.
5. **Testing**
   - Used ICMP (ping) and HTTP requests to verify that traffic to blocked IPs and ports was filtered correctly.

5. Output

Below is a snippet from the firewall output showing packets being analyzed and filtered:

```
[2025-10-27 05:04:02] 10.0.2.15 -> 8.8.8.8 (ICMP) => BLOCKED [ip-block (10.0.2.15 or 8.8.8.8)]

[2025-10-27 05:05:07] 10.0.2.15 -> 142.250.195.46 (TCP) => BLOCKED [port-block (dport=80)]

[2025-10-27 05:05:07] 192.168.1.1 -> 10.0.2.15 (UDP) => ALLOWED [allowed]
```

This output confirms that the system successfully monitors packets, applies filtering rules, and logs the results in real time.

6. Results and Discussion

The firewall successfully identifies and filters packets matching user-defined rules.
It provides clear visibility into live network activity and demonstrates the power of user-space packet handling.

Compared to kernel-level tools like iptables, this Python-based implementation offers:

- **Transparency:** Users can view, modify, and test rules easily.
- **Portability:** Works across most Linux-based environments.
- **Educational Value:** Demonstrates how network inspection and filtering work internally.

7. Learning Reflection

This project provided a deeper understanding of **network security fundamentals** and **traffic filtering** mechanisms.
By implementing packet sniffing and rule enforcement manually, I learned how real-world firewalls make decisions based on multiple parameters and how to structure configurable systems using JSON.

The debugging process also strengthened my ability to read and interpret raw network data, an essential skill for cybersecurity and ethical hacking.

8. Conclusion

The Personal Firewall project demonstrates how Python can be used to design a functional and educational network monitoring tool.
It highlights how Scapy empowers developers to explore packet-level security controls and lays a foundation for future enhancements such as GUI visualization, system-level enforcement, and automated logging to external databases.