Personal Firewall Using Python

Introduction

A firewall is a network security tool that monitors and controls the flow of network traffic based on predefined rules. The project aims to create a lightweight and customizable firewall that can BLOCK or ALLOW traffic according to user-defined rules. It helps users understand how basic firewalls function and provides hands-on experience in network security.

Objective

The main objective of this project is to develop a simple personal firewall that:

- Monitors incoming and outgoing packets.
- Filters packets based on IP, port, and protocol rules.
- Logs suspicious or blocked packets for review.

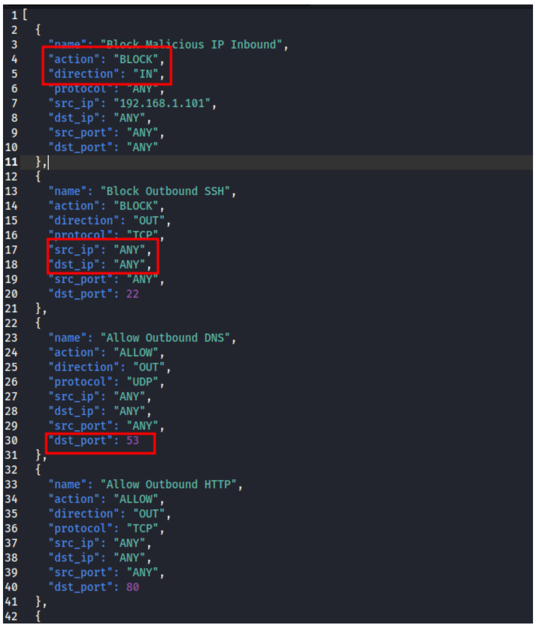
Steps Involved

1. Start: Run
   sudo python3 firewall_cli.py start.
2. Load Rules - The script reads the rules defined in config/rules.json.
3. Start Sniffing using Scapy (via core/engine.py) begins listening to network traffic on your machine.
4. Compare packet details against the loaded rules (from rules.json) one by one.
5. If a BLOCK rule matches, print "[LOGGED AS BLOCK]" message to the console and log a warning to logs/firewall.txt.
6. If an ALLOW rule matches *before* any block rule, print a "[LOGGED AS ALLOW]" message and log info to logs/firewall.txt.
7. Stop: press CTRL+C or run sudo python3 firewall_cli.py stop

Tools Used

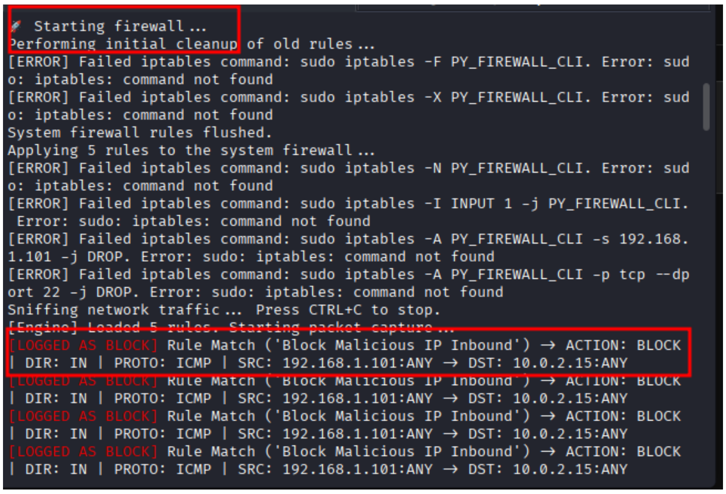
- Python3 – logical code for overall operations
- Scapy – network traffic sniffer
- Iptables – firewall configure to linux system

Snapshots



```json
[
  {
    "name": "Block Malicious IP Inbound",
    "action": "BLOCK",
    "direction": "IN",
    "protocol": "ANY",
    "src_ip": "192.168.1.101",
    "dst_ip": "ANY",
    "src_port": "ANY",
    "dst_port": "ANY"
  },
  {
    "name": "Block Outbound SSH",
    "action": "BLOCK",
    "direction": "OUT",
    "protocol": "TCP",
    "src_ip": "ANY",
    "dst_ip": "ANY",
    "src_port": "ANY",
    "dst_port": 22
  },
  {
    "name": "Allow Outbound DNS",
    "action": "ALLOW",
    "direction": "OUT",
    "protocol": "UDP",
    "src_ip": "ANY",
    "dst_ip": "ANY",
    "src_port": "ANY",
    "dst_port": 53
  },
  {
    "name": "Allow Outbound HTTP",
    "action": "ALLOW",
    "direction": "OUT",
    "protocol": "TCP",
    "src_ip": "ANY",
    "dst_ip": "ANY",
    "src_port": "ANY",
    "dst_port": 80
  },
  {
```

Fig: firewall rules in json file



```
Starting firewall ...
Performing initial cleanup of old rules ...
[ERROR] Failed iptables command: sudo iptables -F PY_FIREWALL_CLI. Error: sud
o: iptables: command not found
[ERROR] Failed iptables command: sudo iptables -X PY_FIREWALL_CLI. Error: sud
o: iptables: command not found
System firewall rules flushed.
Applying 5 rules to the system firewall ...
[ERROR] Failed iptables command: sudo iptables -N PY_FIREWALL_CLI. Error: sud
o: iptables: command not found
[ERROR] Failed iptables command: sudo iptables -I INPUT 1 -j PY_FIREWALL_CLI.
 Error: sudo: iptables: command not found
[ERROR] Failed iptables command: sudo iptables -A PY_FIREWALL_CLI -s 192.168.
1.101 -j DROP. Error: sudo: iptables: command not found
[ERROR] Failed iptables command: sudo iptables -A PY_FIREWALL_CLI -p tcp --dp
ort 22 -j DROP. Error: sudo: iptables: command not found
Sniffing network traffic ... Press CTRL+C to stop.
[Engine] Loaded 5 rules. Starting packet capture ...
[LOGGED AS BLOCK] Rule Match ('Block Malicious IP Inbound') → ACTION: BLOCK
| DIR: IN | PROTO: ICMP | SRC: 192.168.1.101:ANY → DST: 10.0.2.15:ANY
[LOGGED AS BLOCK] Rule Match ('Block Malicious IP Inbound') → ACTION: BLOCK
| DIR: IN | PROTO: ICMP | SRC: 192.168.1.101:ANY → DST: 10.0.2.15:ANY
[LOGGED AS BLOCK] Rule Match ('Block Malicious IP Inbound') → ACTION: BLOCK
| DIR: IN | PROTO: ICMP | SRC: 192.168.1.101:ANY → DST: 10.0.2.15:ANY
[LOGGED AS BLOCK] Rule Match ('Block Malicious IP Inbound') → ACTION: BLOCK
| DIR: IN | PROTO: ICMP | SRC: 192.168.1.101:ANY → DST: 10.0.2.15:ANY
```

Fig: firewall blocking malicious IP

Conclusion

This project successfully outlines the structure and core components for a functional, command-line personal firewall using Python. It effectively combines Scapy for network packet sniffing and analysis with iptables for actual system-level rule enforcement, managed through a clear JSON configuration. While a proof-of-concept, it provides a solid foundation for understanding packet filtering, rule management, and interfacing Python with system security tools.