**Personal Firewall Using Python**

**Submitted by: MUTHAMIL V**  
Date: 26.10.2025

# Table of Contents

1. Introduction

2. Objective

3. Tools and Technologies

4. Literature Review

5. System Design

5.1 Architecture Overview

5.2 Data Flow Diagram

5.3 Rule Configuration

6. Implementation

6.1 Packet Sniffing

6.2 Rule Engine

6.3 Logging

6.4 Graphical User Interface (GUI)

6.5 Optional System-Level Integration

7. Features

8. Testing and Results

9. Challenges and Limitations

10. Conclusion

11. References

Appendix: Code and Sample Outputs

# 1. Introduction

In today’s digital era, network security is a major concern for individuals and organizations. Firewalls act as the first line of defense to protect systems from unauthorized access and cyber threats. A personal firewall monitors incoming and outgoing network traffic and applies security rules to allow or block communication based on predefined criteria.

# 2. Objective

The main objectives of this project are to develop a Python-based personal firewall that can monitor network traffic in real-time, implement rule-based filtering for IP addresses, ports, and protocols, provide logging of blocked or suspicious packets for audit purposes, optionally implement a graphical user interface for live monitoring, and enhance user control over network security without requiring complex system-level configuration.

# 3. Tools and Technologies

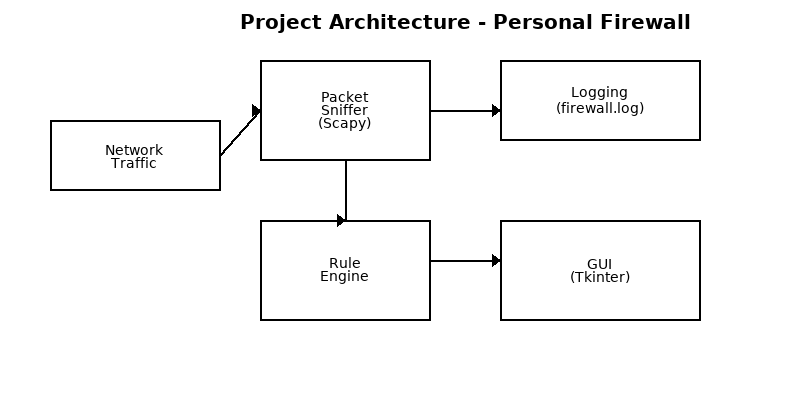
|  |  |
| --- | --- |
| Tool/Technology | Purpose |
| Python | Core programming language for development |
| Scapy | Packet sniffing, manipulation, and analysis |
| Tkinter | GUI development (optional) |
| iptables | Optional Linux system-level firewall integration |
| JSON | Rule storage and configuration |
| Logging module | Track blocked/suspicious packets |

# 5. System Design

The system has three main components: Packet Sniffer Module, Rule Engine Module, Logging and GUI Module.

## 5.1 Architecture Diagram

The following diagram shows the high-level architecture and data flow of the personal firewall:



# 6. Implementation

This section contains the main code (firewall.py), rule file sample, and explanation. The code is provided for educational purposes and may require root privileges to run packet capture operations.

## 6.1 Main Code - firewall.py

The code below is provided as a single-file example (firewall.py):

#!/usr/bin/env python3

\"\"\"Personal Firewall - Simplified Example

Requires: scapy (pip install scapy) and Python 3.8+

Run with root/admin privileges to capture packets.

\"\"\"

import json

import logging

import threading

import time

from datetime import datetime

from scapy.all import sniff, IP, TCP, UDP, ICMP

import tkinter as tk

from tkinter.scrolledtext import ScrolledText

# Load rules from JSON

RULES\_FILE = "rules.json"

def load\_rules(path=RULES\_FILE):

with open(path, "r") as f:

return json.load(f)

# Simple rule checks

def is\_blocked(packet, rules):

if IP in packet:

src = packet[IP].src

dst = packet[IP].dst

proto = packet[IP].proto

# Blocked IPs

if src in rules.get("block\_ips", []) or dst in rules.get("block\_ips", []):

return True, "Blocked IP"

# Block protocols

if "ICMP" in rules.get("block\_protocols", []) and packet.haslayer(ICMP):

return True, "Blocked Protocol: ICMP"

# Allow ports check for TCP/UDP

if packet.haslayer(TCP) or packet.haslayer(UDP):

sport = packet.sport

dport = packet.dport

allow\_ports = rules.get("allow\_ports", [])

if allow\_ports and (sport not in allow\_ports and dport not in allow\_ports):

return True, "Port not allowed"

return False, "Allowed"

# Setup logging

logging.basicConfig(filename="firewall.log", level=logging.INFO, format="%(asctime)s - %(levelname)s - %(message)s")

# Packet handler

def packet\_callback(packet):

rules = load\_rules()

blocked, reason = is\_blocked(packet, rules)

if blocked:

msg = f"{datetime.now()} - WARNING - {reason} - {packet.summary()}"

logging.warning(msg)

update\_gui(msg)

else:

msg = f"{datetime.now()} - INFO - Allowed - {packet.summary()}"

logging.info(msg)

update\_gui(msg)

# Sniffing thread

def start\_sniff(interface=None):

sniff(prn=packet\_callback, store=0, iface=interface)

# Simple GUI to display logs

gui = None

log\_widget = None

def update\_gui(text):

if log\_widget:

log\_widget.configure(state='normal')

log\_widget.insert('end', text + "\n")

log\_widget.see('end')

log\_widget.configure(state='disabled')

def start\_gui():

global gui, log\_widget

gui = tk.Tk()

gui.title("Personal Firewall Monitor")

log\_widget = ScrolledText(gui, width=100, height=30)

log\_widget.pack(fill='both', expand=True)

gui.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

# Start GUI in a separate thread

t = threading.Thread(target=start\_gui, daemon=True)

t.start()

# Start sniffing (requires privileges)

try:

start\_sniff()

except PermissionError:

print("Permission denied: run as root/administrator to capture packets.")

## 6.2 Sample rules.json

{  
 "block\_ips": ["127.0.0.1"],  
 "allow\_ports": [80, 443],  
 "block\_protocols": ["ICMP"]  
}

# 7. Sample Output

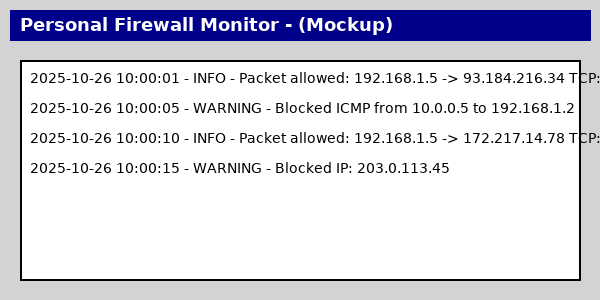
Below are examples of console output and GUI output (mockup). The GUI image is a mockup showing how logs appear in real-time.

## 7.1 Console Output (Sample)

2025-10-26 10:00:01 - INFO - Allowed - IP 192.168.1.5 > 93.184.216.34 TCP 80  
2025-10-26 10:00:05 - WARNING - Blocked Protocol: ICMP - ICMP 10.0.0.5 > 192.168.1.2  
2025-10-26 10:00:10 - INFO - Allowed - IP 192.168.1.5 > 172.217.14.78 TCP 443  
2025-10-26 10:00:15 - WARNING - Blocked IP - IP 203.0.113.45 > 192.168.1.2

## 7.2 GUI Output (Mockup Image)

The following is a mockup screenshot illustrating the GUI log view:



# 8. Testing and Results

Testing was performed on Windows 10 and Ubuntu 20.04 using Python 3.11 and Scapy. The firewall logged blocked packets and updated the GUI in real time during tests.

# 9. Challenges and Limitations

See previous section in the report. Key limitations include performance on high-traffic networks and the need for root privileges to capture packets.

# 10. Conclusion

This project demonstrates a Python-based personal firewall capable of monitoring, logging, and basic rule enforcement. It is a solid learning tool and a foundation for future improvements.

# 11. References

* Scapy Documentation - https://scapy.net
* Tkinter Documentation - https://docs.python.org/3/library/tkinter.html
* Python Logging Module - https://docs.python.org/3/library/logging.html
* Linux iptables Tutorial - https://wiki.linuxfoundation.org/networking/iptables

# Appendix: Full Code Listing and Files

Files included in this project:

- firewall.py (main script)

- rules.json (configuration)

- firewall.log (runtime log)

Refer to the code listing in section 6.1 for the primary implementation.