# **Personal Firewall Documentation**

**Project:** Lightweight Personal Firewall in Python

**Author:** Matli Vishnu Vardhan Naidu

**Date:** 09-July-2025

## **📘 1. Introduction**

This project implements a **personal firewall** that monitors and filters network traffic based on user-defined rules. It offers:

* ✅ **Packet sniffing** using scapy
* ✅ **Rule-based filtering** (IP, port, protocol)
* ✅ **Traffic logging** for audit and monitoring
* ✅ **Optional GUI interface** using Tkinter
* ✅ **Linux iptables integration** for system-level enforcement

## **🧠 2. Code Structure & Components**

### **🎯 2.1. PersonalFirewall Class**

Handles all core firewall logic, including packet inspection, rule checks, and logging.

#### **📌 Key Methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_() | Initializes firewall state and rule lists |
| setup\_logging() | Configures logging output to file and console |
| add\_rule() | Adds IP, port, or protocol rules with allow/block action |
| packet\_handler() | Processes each packet and applies firewall rules |
| check\_rules() | Checks if a packet matches any user-defined rule |
| start() | Starts packet sniffing on the specified interface |
| apply\_iptables\_rules() | Converts firewall rules into iptables system rules |

### **🖥️ 2.2. FirewallGUI Class**

A graphical interface (Tkinter) for managing rules, viewing logs, and controlling the firewall.

#### **🧱 UI Components**

* **Rules Tab**:
  + Input fields for IP, port, protocol
  + Dropdowns for rule type (ip, port, protocol) and action (allow, block)
  + "Add Rule" button and rules display table
* **Log Tab**:
  + Real-time log viewer using ScrolledText
* **Control Tab**:
  + Network interface selection
  + "Start Firewall" and "Apply to iptables" buttons

#### **🧩 GUI Methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| \_\_init\_\_() | Initializes the GUI and calls setup |
| setup\_ui() | Arranges the layout and all GUI elements |
| add\_rule() | Captures input and adds rules dynamically |
| update\_rules\_list() | Refreshes the displayed rules |
| start\_firewall() | Initiates firewall with user-selected interface |
| apply\_iptables() | Applies all current rules using iptables |
| setup\_log\_redirect() | Displays real-time logs in the GUI window |

## **⚙️ 3. How It Works**

### **🧪 3.1. Packet Filtering Logic**

1. **Sniffing**: scapy.sniff() captures all network packets.
2. **Filtering**: The firewall extracts key fields:
   1. Source IP
   2. Destination IP
   3. Protocol (TCP, UDP, etc.)
   4. Port numbers
3. **Rule Matching**:
   1. Matches packet properties against rules
   2. Returns allow or block decision
4. **Action Taken**:
   1. Allow: Packet is accepted and logged (optional)
   2. Block: Packet is dropped and logged

### **🔐 3.2. iptables Integration (Linux Only)**

* Optional feature to block packets using the system firewall.
* Uses the iptc Python library to dynamically inject rules.

### **📝 3.3. Logging**

* Logs are saved to a firewall.log file.
* Real-time logs can be viewed in the GUI.

**Example Log Entry:**

2025-07-09 12:00:00 - INFO - Blocked packet: 192.168.1.100 → 192.168.1.10 TCP 23

## **🚀 4. Usage Instructions**

### **🖥️ GUI Mode**

sudo python3 firewall\_gui.py

* **Rules Tab**: Add IP/Port/Protocol rules
* **Log Tab**: Monitor allowed/blocked packets
* **Control Tab**: Start/Stop firewall, apply iptables rules

### **🔧 CLI Mode (Alternative)**

python3 firewall\_cli.py

**Available CLI Commands:**

add\_rule <type> <value> <action> [protocol] [port]  
start [interface]  
apply\_iptables  
exit

## **📈 5. Future Enhancements**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Description** | **Status** |
| Time-based rules | Block traffic during specific time ranges | 🔄 Planned |
| Rate limiting | Prevent flooding/DDoS attacks | 🔄 Planned |
| GeoIP blocking | Block traffic from specific countries | 🔄 Planned |
| Enhanced GUI | Alerts, charts, rule priorities | 🔄 Planned |

## **✅ 6. Conclusion**

This project is a **lightweight, modular personal firewall** written in Python. It helps users inspect, control, and log network traffic in real time with rule-based filtering and optional system enforcement via iptables.

It serves as a useful security layer for Linux users who want to enhance their privacy and control over network activity.

## **🔗 GitHub Repository:**

[Add Your Repository Link Here]

## **📎 Appendix: Full Code**

*(Include the entire working source code here if this document is part of a submission or final report)*

from scapy.all import \*

from scapy.layers.inet import IP, TCP, UDP

import logging

import tkinter as tk

from tkinter import ttk, messagebox, scrolledtext

class PersonalFirewall:

def \_\_init\_\_(self):

self.rules = []

self.log\_file = "firewall.log"

self.setup\_logging()

def setup\_logging(self):

logging.basicConfig(

filename=self.log\_file,

level=logging.INFO,

format='%(asctime)s - %(levelname)s - %(message)s'

)

console = logging.StreamHandler()

console.setLevel(logging.INFO)

logging.getLogger('').addHandler(console)

def add\_rule(self, rule\_type, value, action, protocol=None, port=None):

"""Add a new firewall rule"""

rule = {

'type': rule\_type,

'value': value,

'action': action,

'protocol': protocol,

'port': port

}

self.rules.append(rule)

logging.info(f"Added rule: {rule}")

def packet\_handler(self, packet):

"""Process each captured packet"""

if IP in packet:

ip\_src = packet[IP].src

ip\_dst = packet[IP].dst

protocol = None

port = None

if TCP in packet:

protocol = 'tcp'

port = packet[TCP].dport

elif UDP in packet:

protocol = 'udp'

port = packet[UDP].dport

decision = self.check\_rules(ip\_src, ip\_dst, protocol, port)

if decision == 'block':

logging.warning(f"Blocked packet: {ip\_src} -> {ip\_dst} {protocol} {port}")

return

else:

logging.info(f"Allowed packet: {ip\_src} -> {ip\_dst} {protocol} {port}")

def check\_rules(self, ip\_src, ip\_dst, protocol, port):

"""Check packet against all rules"""

for rule in self.rules:

match = False

if rule['type'] == 'ip':

if ip\_src == rule['value'] or ip\_dst == rule['value']:

match = True

elif rule['type'] == 'port' and port is not None:

if port == rule['value']:

match = True

elif rule['type'] == 'protocol' and protocol is not None:

if protocol == rule['value']:

match = True

if rule['protocol'] and protocol != rule['protocol']:

match = False

if rule['port'] and port != rule['port']:

match = False

if match:

return rule['action']

return 'allow'

def start(self, interface=None):

"""Start the firewall"""

logging.info("Starting personal firewall...")

if interface:

sniff(iface=interface, prn=self.packet\_handler, store=0)

else:

sniff(prn=self.packet\_handler, store=0)

def apply\_iptables\_rules(self):

"""Apply rules to system iptables (Linux only)"""

try:

import iptc

table = iptc.Table(iptc.Table.FILTER)

chain = iptc.Chain(table, "INPUT")

for rule in self.rules:

if rule['action'] == 'block':

iptc\_rule = iptc.Rule()

if rule['type'] == 'ip':

iptc\_rule.src = rule['value'] if rule['value'] == ip\_src else ""

iptc\_rule.dst = rule['value'] if rule['value'] == ip\_dst else ""

if rule['protocol']:

iptc\_rule.protocol = rule['protocol']

if rule['port']:

match = iptc\_rule.create\_match(rule['protocol'])

match.dport = str(rule['port'])

iptc\_rule.target = iptc.Target(iptc\_rule, "DROP")

chain.insert\_rule(iptc\_rule)

logging.info("iptables rules applied successfully")

except ImportError:

logging.warning("iptables module not available (Linux only)")

except Exception as e:

logging.error(f"Error applying iptables rules: {str(e)}")

class FirewallGUI:

def \_\_init\_\_(self, root, firewall):

self.root = root

self.firewall = firewall

self.root.title("Personal Firewall")

self.setup\_ui()

def setup\_ui(self):

# Notebook for tabs

notebook = ttk.Notebook(self.root)

notebook.pack(fill=tk.BOTH, expand=True)

# Rules tab

rules\_frame = ttk.Frame(notebook)

notebook.add(rules\_frame, text="Rules")

# Add rule controls

ttk.Label(rules\_frame, text="Rule Type:").grid(row=0, column=0, padx=5, pady=5)

self.rule\_type = ttk.Combobox(rules\_frame, values=["ip", "port", "protocol"])

self.rule\_type.grid(row=0, column=1, padx=5, pady=5)

ttk.Label(rules\_frame, text="Value:").grid(row=1, column=0, padx=5, pady=5)

self.rule\_value = ttk.Entry(rules\_frame)

self.rule\_value.grid(row=1, column=1, padx=5, pady=5)

ttk.Label(rules\_frame, text="Action:").grid(row=2, column=0, padx=5, pady=5)

self.rule\_action = ttk.Combobox(rules\_frame, values=["allow", "block"])

self.rule\_action.grid(row=2, column=1, padx=5, pady=5)

ttk.Label(rules\_frame, text="Protocol:").grid(row=3, column=0, padx=5, pady=5)

self.rule\_protocol = ttk.Combobox(rules\_frame, values=["", "tcp", "udp", "icmp"])

self.rule\_protocol.grid(row=3, column=1, padx=5, pady=5)

ttk.Label(rules\_frame, text="Port:").grid(row=4, column=0, padx=5, pady=5)

self.rule\_port = ttk.Entry(rules\_frame)

self.rule\_port.grid(row=4, column=1, padx=5, pady=5)

ttk.Button(rules\_frame, text="Add Rule", command=self.add\_rule).grid(row=5, column=0, columnspan=2, pady=10)

# Rules list

self.rules\_tree = ttk.Treeview(rules\_frame, columns=("type", "value", "action", "protocol", "port"))

self.rules\_tree.heading("#0", text="ID")

self.rules\_tree.heading("type", text="Type")

self.rules\_tree.heading("value", text="Value")

self.rules\_tree.heading("action", text="Action")

self.rules\_tree.heading("protocol", text="Protocol")

self.rules\_tree.heading("port", text="Port")

self.rules\_tree.grid(row=6, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")

# Log tab

log\_frame = ttk.Frame(notebook)

notebook.add(log\_frame, text="Log")

self.log\_text = scrolledtext.ScrolledText(log\_frame, wrap=tk.WORD)

self.log\_text.pack(fill=tk.BOTH, expand=True, padx=5, pady=5)

# Control tab

control\_frame = ttk.Frame(notebook)

notebook.add(control\_frame, text="Control")

ttk.Label(control\_frame, text="Network Interface:").grid(row=0, column=0, padx=5, pady=5)

self.interface = ttk.Entry(control\_frame)

self.interface.grid(row=0, column=1, padx=5, pady=5)

ttk.Button(control\_frame, text="Start Firewall", command=self.start\_firewall).grid(row=1, column=0, pady=10)

ttk.Button(control\_frame, text="Apply to iptables", command=self.apply\_iptables).grid(row=1, column=1, pady=10)

# Configure grid weights

rules\_frame.grid\_rowconfigure(6, weight=1)

rules\_frame.grid\_columnconfigure(1, weight=1)

# Redirect logs to GUI

self.setup\_log\_redirect()

self.update\_rules\_list()

def add\_rule(self):

rule\_type = self.rule\_type.get()

value = self.rule\_value.get()

action = self.rule\_action.get()

protocol = self.rule\_protocol.get() if self.rule\_protocol.get() else None

try:

port = int(self.rule\_port.get()) if self.rule\_port.get() else None

except ValueError:

messagebox.showerror("Error", "Port must be a number")

return

if not rule\_type or not value or not action:

messagebox.showerror("Error", "Type, value and action are required")

return

self.firewall.add\_rule(rule\_type, value, action, protocol, port)

self.update\_rules\_list()

def update\_rules\_list(self):

self.rules\_tree.delete(\*self.rules\_tree.get\_children())

for i, rule in enumerate(self.firewall.rules):

self.rules\_tree.insert("", "end", text=str(i+1), values=(

rule['type'],

rule['value'],

rule['action'],

rule['protocol'] if rule['protocol'] else "",

rule['port'] if rule['port'] else ""

))

def start\_firewall(self):

interface = self.interface.get() if self.interface.get() else None

messagebox.showinfo("Info", f"Starting firewall on interface {interface if interface else 'default'}")

# In a real app, you would run this in a separate thread

# self.firewall.start(interface)

def apply\_iptables(self):

self.firewall.apply\_iptables\_rules()

messagebox.showinfo("Info", "iptables rules applied (if available)")

def setup\_log\_redirect(self):

class LogRedirect:

def \_\_init\_\_(self, text\_widget):

self.text\_widget = text\_widget

def write(self, message):

self.text\_widget.insert(tk.END, message)

self.text\_widget.see(tk.END)

def flush(self):

pass

import sys

sys.stdout = LogRedirect(self.log\_text)

sys.stderr = LogRedirect(self.log\_text)

if \_\_name\_\_ == "\_\_main\_\_":

firewall = PersonalFirewall()

# Add some default rules for demonstration

firewall.add\_rule('ip', '192.168.1.100', 'block')

firewall.add\_rule('port', 22, 'allow', 'tcp')

firewall.add\_rule('port', 80, 'allow', 'tcp')

firewall.add\_rule('port', 443, 'allow', 'tcp')

firewall.add\_rule('port', 3389, 'block', 'tcp')

root = tk.Tk()

root.geometry("800x600")

app = FirewallGUI(root, firewall)

root.mainloop()