PROJECT INFORMATION

Project Name: Personal Firewall Using Python

Developer: Meet Pandit

Project Duration: October 2025

Status: Completed and Operational

PERSONAL FIREWALL PROJECT

Network Security Implementation

Elevate Labs Project Documentation

TABLE OF CONTENTS

1. Abstract

2. Introduction

3. Tools and Technologies Used

4. Steps Involved in Building the Project

5. Project Architecture

6. Implementation Details

7. Testing and Validation

8. Conclusion

9. Future Enhancements

10. References

═══════════════════════════════════════════════════════════════════

1. ABSTRACT

This project presents the development and implementation of a Personal Firewall system designed to monitor, filter, and control network traffic on a Windows-based computer system. The firewall operates at the network layer, intercepting packets in real-time and evaluating them against user-defined security rules.

The system utilizes Python programming language along with Scapy library for packet sniffing and analysis. It provides a command-line interface for managing firewall rules and monitoring network activity. The firewall can block or allow traffic based on various criteria including IP addresses, port numbers, protocols (TCP/UDP/ICMP), and traffic direction (incoming/outgoing).

Key features include:

• Real-time packet capture and analysis

• Rule-based traffic filtering

• Detailed logging of network activities

• Support for multiple protocols (TCP, UDP, ICMP)

• Configurable interface selection

• Administrator-level security controls

This implementation serves as an educational tool for understanding network security principles and demonstrates practical application of packet filtering techniques.

2. INTRODUCTION

2.1 Background

In today's interconnected digital landscape, network security has become paramount. Every device connected to the internet is constantly exposed to potential threats including unauthorized access, malware, data theft, and network attacks. A firewall acts as the first line of defense, monitoring and controlling network traffic based on predetermined security rules.

2.2 Purpose and Objectives

The primary objective of this project is to develop a functional personal firewall that:

1. Monitors all incoming and outgoing network traffic

2. Implements rule-based packet filtering

3. Provides real-time visibility into network activities

4. Blocks malicious or unwanted connections

5. Allows legitimate traffic to pass through

6. Logs all network events for security analysis

2.3 Problem Statement

Standard operating systems come with built-in firewalls, but understanding their inner workings and customizing them for specific needs can be challenging. This project addresses the need for:

• Educational understanding of firewall mechanisms

• Customizable security rules tailored to specific requirements

• Transparent packet inspection and logging

• Hands-on experience with network security implementation

2.4 Scope

This personal firewall implementation focuses on:

• Packet-level inspection and filtering

• IPv4 network traffic (TCP, UDP, ICMP protocols)

• Windows operating system environment

• Command-line interface for management

• Rule-based access control

The project does not cover:

• Application-layer filtering

• Deep packet inspection (DPI)

• Intrusion detection/prevention systems (IDS/IPS)

• IPv6 protocol support

• Graphical user interface

3. TOOLS AND TECHNOLOGIES USED

3.1 Programming Language

Python 3.x

• Chosen for its extensive networking libraries

• Cross-platform compatibility

• Rich ecosystem of security-related packages

• Ease of development and maintenance

• Strong community support

3.2 Core Libraries and Frameworks

a) Scapy (Packet Manipulation Library)

• Version: Latest stable release

• Purpose: Packet capture, analysis, and manipulation

• Features used:

- AsyncSniffer for non-blocking packet capture

- Protocol layer parsing (IP, TCP, UDP, ICMP)

- Packet summary and metadata extraction

b) Npcap (Packet Capture Driver)

• Version: 1.79

• Purpose: Low-level packet capture on Windows

• WinPcap API-compatible mode enabled

• Required for administrator-level packet sniffing

c) PSUtil (System and Process Utilities)

• Purpose: Network interface information retrieval

• Features used:

- Local IP address detection

- Network interface enumeration

- System resource monitoring

3.3 Development Environment

• IDE: Visual Studio Code

• Operating System: Windows 10/11

• Terminal: Windows PowerShell (Administrator mode)

• Version Control: Git (optional)

• Python Package Manager: pip

3.4 Additional Tools

• Windows PowerShell: Command execution and testing

• Network Testing Tools: ping, curl, web browsers

• Text Editor: For rule configuration files

• Command-line Interface: For firewall management

4. STEPS INVOLVED IN BUILDING THE PROJECT

4.1 PHASE 1: Environment Setup

Step 1: Python Installation

• Download Python 3.x from python.org

• Install with "Add Python to PATH" option enabled

• Verify installation: python --version

Step 2: Project Structure Creation

Create the following directory structure:

personal\_firewall\_1/

├── src/

│ ├── \_\_init\_\_.py

│ ├── cli.py # Command-line interface

│ ├── sniffer.py # Packet sniffing logic

│ ├── rules.py # Rule evaluation engine

│ └── logger\_util.py # Logging configuration

├── requirements.txt # Python dependencies

└── README.md # Project documentation

Step 3: Install Required Dependencies

Create requirements.txt with:

scapy

psutil

Install dependencies:

pip install -r requirements.txt

Step 4: Install Npcap Driver

• Download Npcap from https://npcap.com/#download

• Run installer as Administrator

• Enable options:

✓ Install Npcap in WinPcap API-compatible Mode

✓ Support raw 802.11 traffic

• Restart computer after installation