INTERNSHIP PROJECT PHASE - ELEVATE LABS

PERSONAL-FIREWALL using PYTHON

GITHUB: personal_firewall *(cli and gui version)*

KUMARASWAMY G S

(gudemaneswamy0506@gmail.com)

Introduction

This project details the development of a Personal Firewall, a security application designed to protect a computer by controlling network traffic. The project evolved from a basic IP rule checker into a more advanced, cross-platform firewall with both a Graphical User Interface (GUI) and a Command Line Interface (CLI). The final product is a functional and educational tool that demonstrates key cybersecurity concepts.

Abstract

This project is a comprehensive toolkit for building a personal firewall, encompassing a basic firewall simulator and an advanced version. The basic toolkit includes an ip_rule_checker.py for educational purposes and an ip_scanner.py to discover active devices. The advanced version, available as both a GUI (firewall.py) and a CLI (firewall_cli.py), dynamically applies system-level rules using native tools like iptables and netsh advfirewall. It features real-time traffic monitoring, configurable rules, and domain-based blocking by resolving hostnames to IP addresses. The project is an educational and practical tool, showcasing network packet inspection and system-level rule enforcement.

Tools Used

The project was developed primarily using Python 3. The following libraries and system tools were crucial for its functionality:

- Scapy: A powerful Python library for packet manipulation and sniffing, used for real-time traffic monitoring and network scanning.
- tkinter: Python's standard GUI library, used to build the user interface.
- socket: A standard Python library for determining the local machine's IP address and network prefix.
- iptables: The native firewall tool on Linux for enforcing rules.
- pfctl: The native firewall tool on macOS for applying system-level rules.
- netsh advfirewall: The command-line tool for managing Windows Firewall.
- psutil: A Python library used to gather system information for the network scanner.

- requests: A Python library for network communication.
- Npcap: A Windows-specific driver required for Scapy to function correctly.

Steps Involved in Building the Project

The project was built in a modular, iterative manner.

1. Basic Firewall Toolkit

- Firewall Simulator (ip_rule_checker.py): A script was created to simulate firewall behavior by loading rules from a rules.json file and filtering randomly generated packets.
- Network Scanner (ip_scanner.py): This script was developed using Scapy to perform a fast ARP scan. It automatically detected the network range and used multi-threading to gather detailed information (IP, MAC, hostname, manufacturer) about discovered devices.
-

2. Advanced GUI Firewall

- Core Functionality: The project was enhanced to be a functional firewall by integrating with native system commands like iptables, pfctl, and netsh to apply real rules.
- User Interface: A GUI was built with tkinter to provide a user-friendly way to control the firewall and view a live log of network activity.
- Concurrency: Multithreading and a message queue were implemented to keep the GUI responsive while network traffic was monitored.

3. Domain-Based Blocking and CLI Version

- **Domain Resolution**: The firewall's capabilities were expanded to include domain-based blocking by dynamically resolving domain names to their IP addresses, allowing for more intuitive rule creation.
- **CLI Implementation**: A Command Line Interface (`firewall_cli.py`) was developed to provide a more scriptable and automation-friendly alternative, incorporating features like command-line arguments and enhanced logging.

Conclusion

This project successfully evolved from a simple firewall simulator into a functional toolkit. By using system-level commands and libraries like Scapy, it demonstrates how to control and monitor network traffic. The development of both GUI and CLI versions, with features like domain blocking, showcases a strong foundation for future cybersecurity projects.