Keylogger Monitoring System – Project Report

Abstract

This project demonstrates a secure keylogging and monitoring system built for educational and cybersecurity research purposes. The solution records keyboard input, encrypts it, and transmits the data to a mock server for safe monitoring and visualization. It emphasizes encryption, local logging, and secure client-server communication.

Introduction

The main goal of this project is to simulate how encrypted keystroke data can be captured and transmitted securely for testing or monitoring authorized systems. Keylogging is typically associated with malicious activity, but in controlled environments, it serves as an important research tool for studying security threats and data protection. The project consists of three core modules: crypto_utils.py (encryption), keylogger.py (data capture), and mock_server.py (data monitoring).

Tools Used

• Python 3.10+
• Libraries: cryptography, pynput, requests, http.server
• Environment: Localhost testing (port 8080)
• Encryption: Fernet (AES-based symmetric encryption)

Steps Involved in Building the Project

1. Setup and Encryption Design – Created an Encryptor class using the cryptography library and generated secure keys with Fernet.generate_key().
2. Keylogger Implementation – Used pynput to record keystrokes, stored logs in an encrypted file, and transmitted data in batches to the mock server with a kill switch feature.
3. Server and Monitoring Dashboard – Built a lightweight HTTP server to receive, decrypt, and display keystrokes on an auto-refreshing web interface.
4. Testing and Execution – Ran the mock server on port 8080, started the keylogger separately, and verified encrypted transmission and real-time visualization.

Conclusion

This project successfully demonstrates how encryption can secure sensitive event data transmission between a client and server. While a real keylogger poses ethical and legal risks, this implementation serves as a controlled, educational simulation to understand encryption workflows, data handling, and secure client-server communication. Future improvements could include authentication mechanisms, secure sockets, and advanced analytics.