Title: Secure File Storage System with AES

Introduction:

In today's digital world, securing sensitive files is essential to prevent unauthorized access and tampering. This project focuses on developing a secure file storage system using AES (Advanced Encryption Standard) to ensure confidentiality and integrity of files stored on a local machine.

Abstract:

This project implements a secure file storage application that allows users to encrypt and decrypt files using AES-256 encryption. The system supports uploading, encryption with a strong key, metadata logging (filename, hash, timestamp), and secure retrieval. It ensures that only authorized users with the correct key can access the data. Hash verification is used to detect file tampering.

Tools Used:

- Python
- cryptography module (Fernet)
- Tkinter (for GUI)
- hashlib (for SHA-256 hashing)

Steps Involved in Building the Project:

1. Setup Environment: Installed Python and necessary modules (cryptography, tkinter, hashlib).
2. User Interface: Developed a Tkinter-based GUI for file upload, encryption, decryption, and viewing metadata.
3. Encryption Module:
   - Used Fernet (AES in CBC mode with HMAC) from the cryptography module.
   - Generated secure keys and stored them.

- Encrypted files to .enc format.

4. Metadata Logging:

- Logged filename, timestamp, and SHA-256 hash.

5. Decryption Module:

- Allowed file selection and decrypted if correct key is given.

6. Integrity Check:

- Used SHA-256 hash to verify file integrity.

Conclusion:

This AES-based secure file storage system ensures the confidentiality, integrity, and accessibility of files in a local environment. Future enhancements could include cloud support and multi-user access.

Title: Personal Firewall using Python

Introduction:

Cybersecurity threats can arise from unauthorized network activity. A personal firewall helps monitor and control such traffic. This project builds a lightweight firewall using Python and scapy.

Abstract:

The personal firewall uses packet sniffing, rule-based filtering, and real-time logging. Users can define IPs, ports, and protocols to block or allow. A simple GUI allows live monitoring of traffic. Suspicious activity is logged for further review.

Tools Used:

- Python
- scapy
- tkinter (optional GUI)
- iptables (for system-level enforcement on Linux)

Steps Involved in Building the Project:

1. Used scapy to sniff real-time network packets.
2. Defined allow/block rules for IPs and ports.
3. Implemented packet filtering based on rules.
4. Logged suspicious packets to a text file.
5. Created a basic GUI for toggling rules and monitoring traffic.
6. Integrated iptables (optional) for system-level blocking.

Conclusion:

This personal firewall empowers users to control their network traffic in real-time. The rule-based system is extendable to detect port scans and other anomalies.