SECURE FILE SHARING SYSTEM REPORT

NAME: AFOLABI ADESOYE THEOPHILUS

Task 02: Secure File Sharing System

Program: Future Interns Cybersecurity Internship

DATE: 11/20/2025

Task Summary

The **Secure File Sharing System** is a web-based application that enables users to upload and download files securely. All uploaded files are encrypted using **AES-256 encryption** before storage, ensuring data confidentiality and protection against unauthorized access. Each user provides a **personal password**, which is used to derive a unique encryption key for every file, enhancing security and personalization. This project demonstrates the integration of **Python Flask** for backend development, cryptography for encryption and decryption, and modern **frontend technologies (HTML and CSS)** for a clean, responsive, and user-friendly interface.

Tools Utilized

- Python (Flask Framework): For backend web application
- Cryptography Library: For AES-256 encryption and decryption of files.
- HTML & CSS: For frontend design and responsive user interface.
- Linux Terminal (Kali): For setup, testing, and deployment of the application.

Procedure Overview

- Flask Server Setup: A Flask application was created and configured to handle file uploads, storage, and downloads securely.

- HTML Interface Design: A responsive HTML page with CSS styling was developed to allow users to upload, download, and manage files easily.

- AES Encryption Implementation: Using Python's cryptography library, uploaded files are encrypted with AES-256 (GCM mode) before being saved to storage.

- Password-Based Key Generation: Each user provides a personal password, from which a secure encryption key is derived using a key derivation function (KDF).

- File Storage: Encrypted files are stored in a dedicated directory, along with metadata for decryption reference.

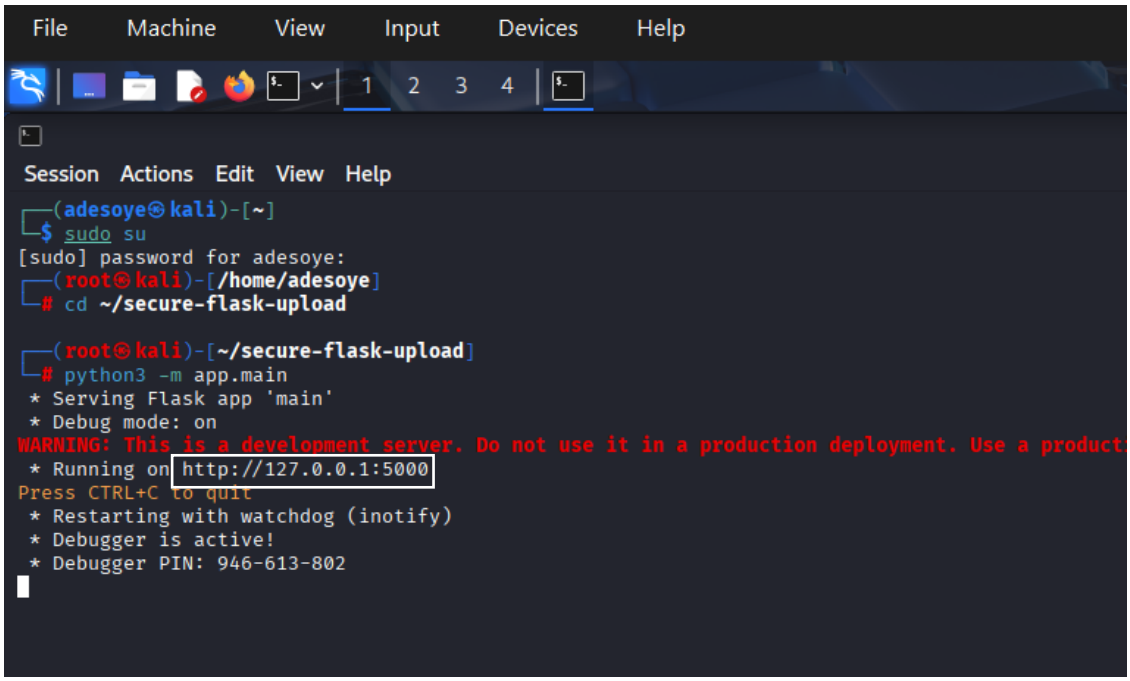
- Decryption Process: When a user requests a download, the system uses the stored metadata and the user's password-derived key to decrypt and serve the file securely.

- Flask Execution: The server is run locally (python3 -m app.main), allowing users to access the web app via http://127.0.0.1:5000.

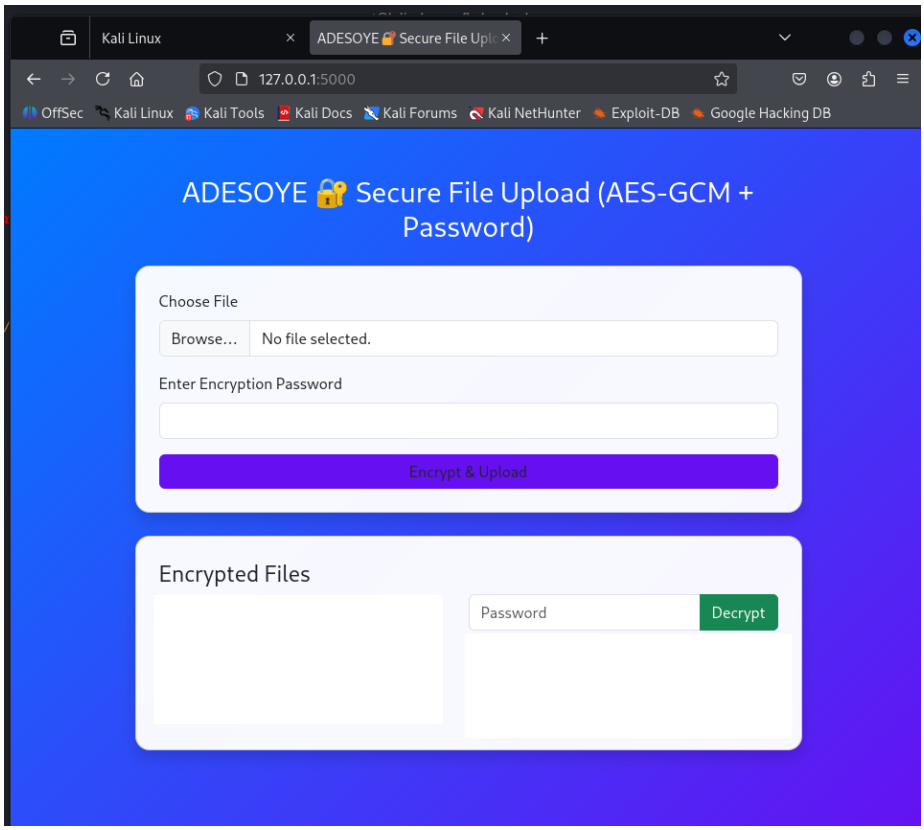
IMPLEMENTATION STEPS

- **Environment Setup**
  - Created project folder: secure-flask-upload
  - Initialized virtual environment:
    - python3 -m venv venv
    - source venv/bin/activate
  - Installed dependencies:
    - pip install Flask cryptography
- **AES Key & Password Setup**
  - Generated a 32-byte AES-256 key.
  - Integrated personal password input to derive a secure key for each encryption and decryption process.
- **Backend (Flask App)**
  - Flask routes implemented:
    - / --- Homepage (list uploaded files)
    - /upload --- Upload + Encrypt file
    - /decrypt/<filename> --- Decrypt file using personal password
- **Frontend (HTML + CSS)**
  - Clean, responsive UI with:
    - File upload form and password field
    - File listing table
    - Decrypt button for each file
- **Encryption & Decryption**
  - Used AES-256 GCM mode for encryption and integrity verification.
  - Uploaded files are encrypted before saving and can be decrypted back using the provided password.
- **Running the Application**
  - Navigate to project directory and run the Flask app:
    - cd ~/secure-flask-upload
    - python3 -m app.main
    - Access the system via: http://127.0.0.1:5000

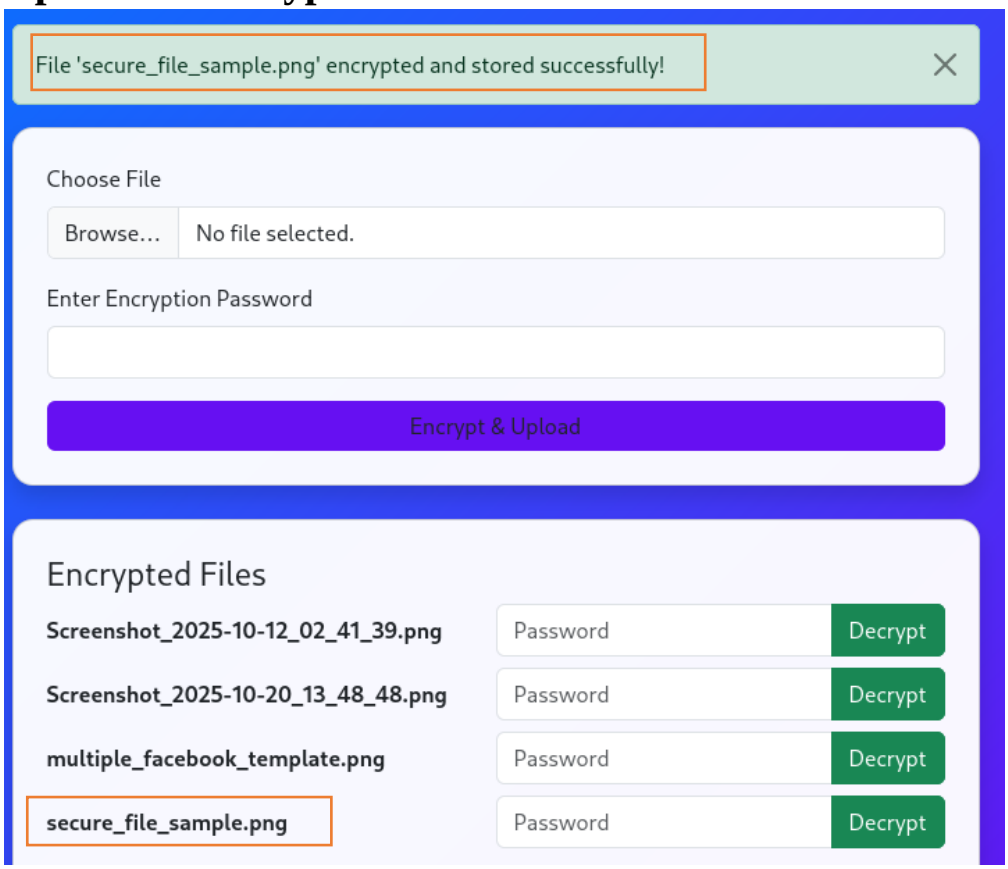
1. Flask server running



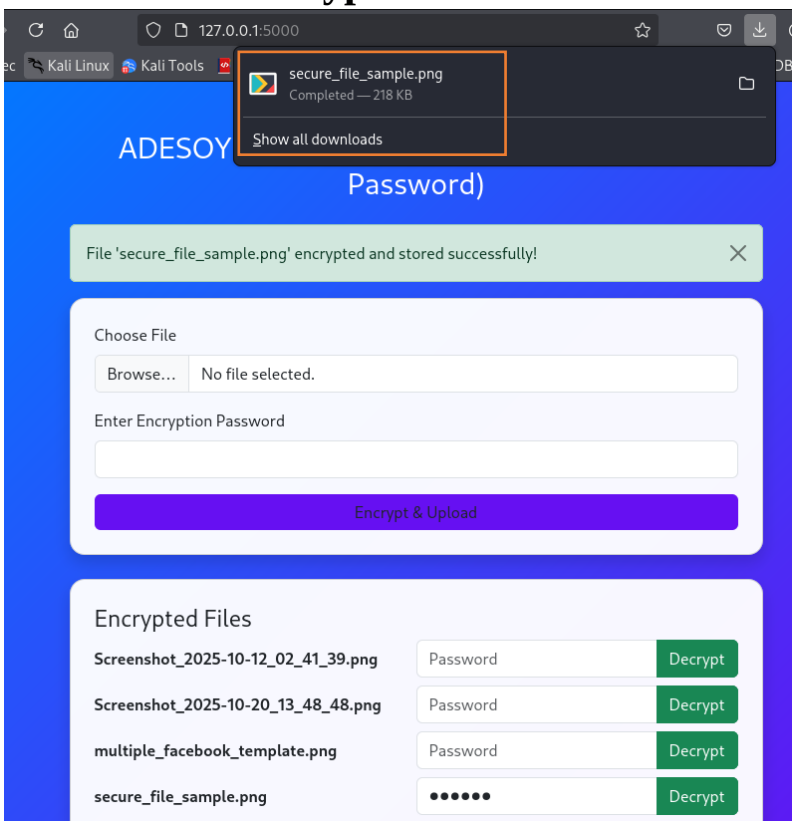
2. Homepage with Upload and encrypt form (http://127.0.0.1:5000)



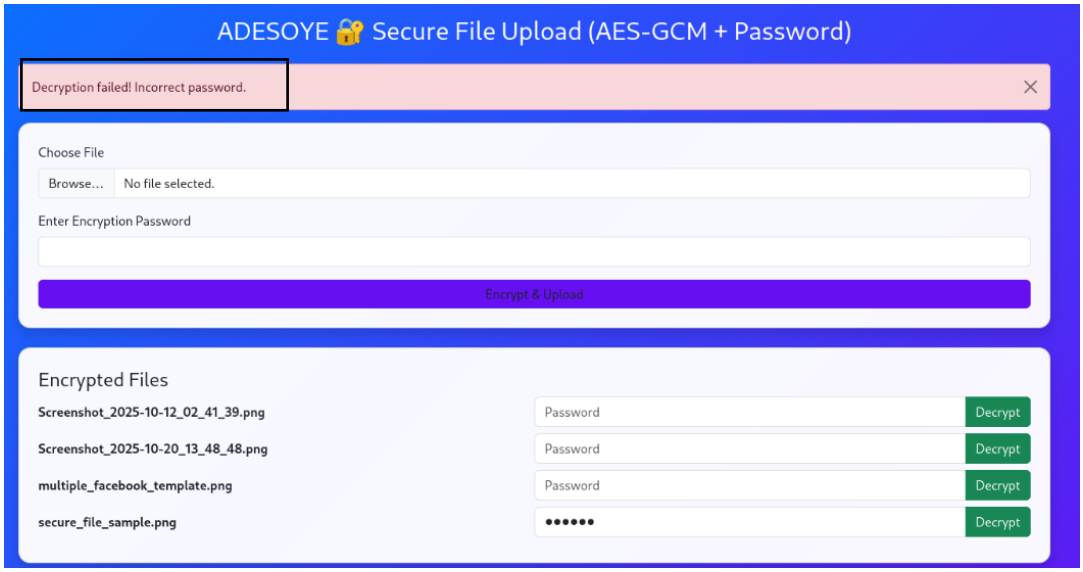
3. Upload & Encrypt



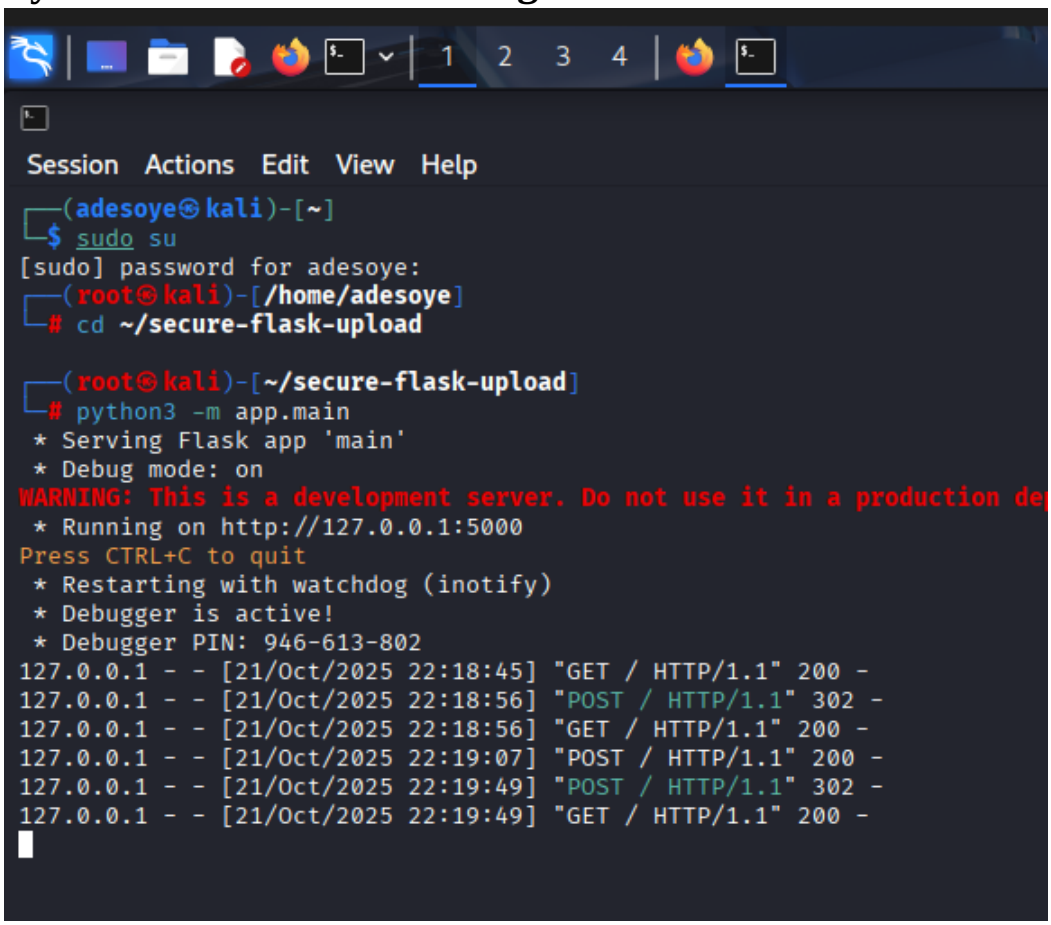
4. Downloaded decrypted file



5. Decryption error caused wrong password



6. Python Flask server Running



Conclusion:

The Secure File Sharing System successfully demonstrates how Flask and AES-256 encryption can be combined to protect user data through password-based encryption and secure file handling. It ensures confidentiality, integrity, and a simple user experience for secure file uploads and downloads.

Limitation:

The system currently lacks multi-user authentication, cloud integration, and advanced key management features, which could be added to enhance scalability and real-world deployment.