Cyber Security Internship at Report FutureInterns

Name : K.KAVI PREETHY

G-MAIL : kavipreethy03@gmail.com

Task 3: Secure File Sharing System

TABLE OF CONTENT

1. Introduction

This project involved the development of a secure file upload and download web portal using Python, Flask, and AES encryption. The system ensures files are encrypted before storage and decrypted only when accessed by an authorized user. It demonstrates practical knowledge of secure coding, cryptographic handling, and backend web development.

2. Objectives

• Develop a secure web portal for file upload and download

• Apply AES encryption to protect files both at rest and during transmission

• Handle encrypted storage with proper key management

• Gain hands-on experience with Flask, Python, and cryptography libraries

3. Tools and Technologies Used

• Python 3 – Programming language for building the backend logic

• Flask – Micro web framework used to develop the file-sharing API

• PyCryptodome – Python library for AES encryption and decryption

• HTML/CSS – To build a basic user interface for file uploads and downloads

• Werkzeug – Provides secure handling of filenames

- PyCharm – IDE used for development and testing
- Kali Linux – Operating system used for development and testing environment

4. Architecture

- app.py – Handles all routes and web logic using Flask
- crypto_utils.py – Contains reusable encryption and decryption functions using AES
- templates/index.html – Frontend user interface for file uploads and downloads
- uploads/ – Stores encrypted versions of uploaded files

5. Functional Workflow

- User uploads a file through the web interface
- The file is encrypted using AES and stored with a .enc extension
- Encrypted files are stored in the uploads/ directory
- During download, the file is decrypted temporarily and served to the user

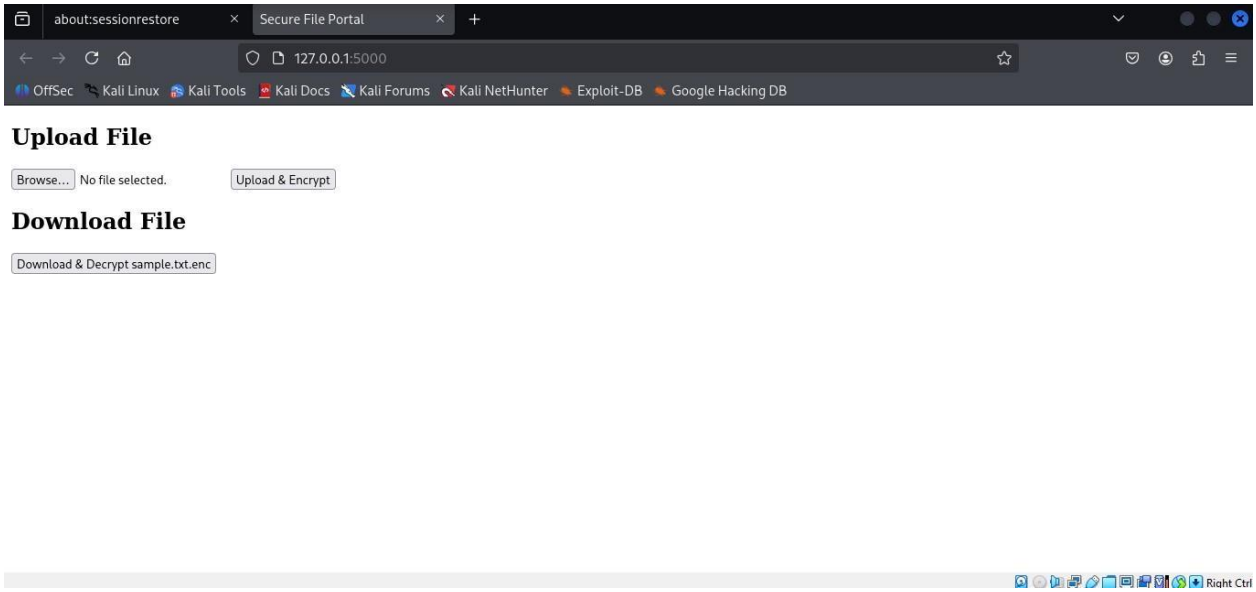
6. Module Functionality

- app.py – Manages Flask routes, file validation, encryption and decryption flow
- crypto_utils.py – Provides encryption and decryption functions using AES with padding
- index.html – Provides a simple web interface for upload/download

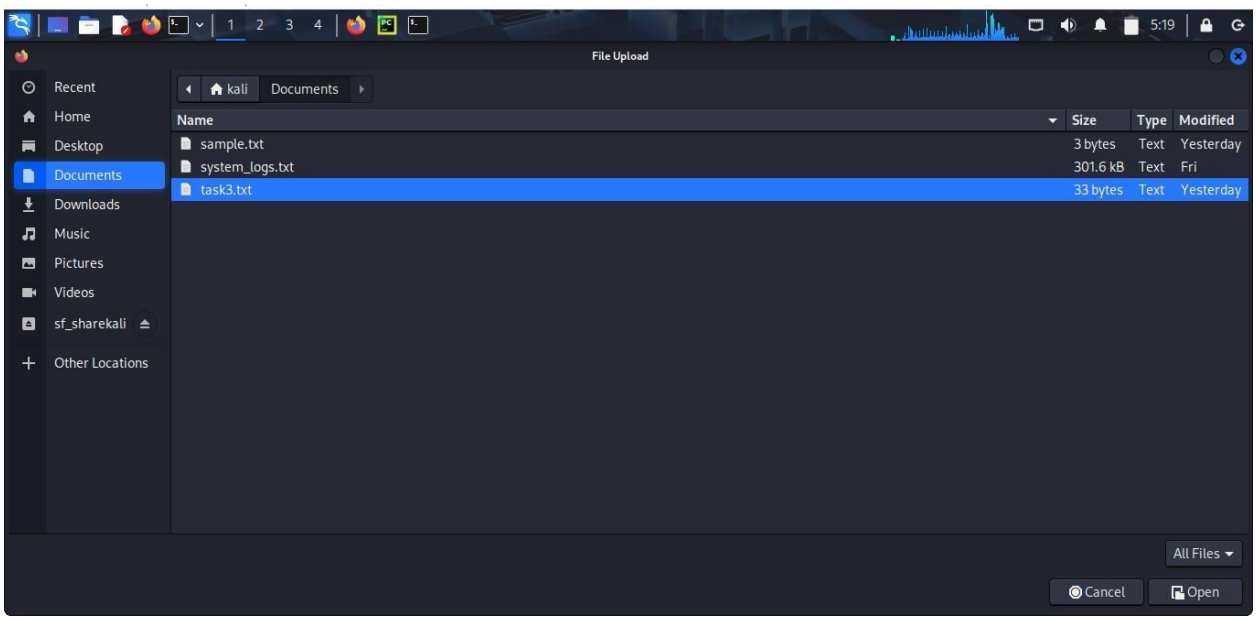
7. Testing and Validation

- Uploading various file types – Success
- File encryption validation – Encrypted .enc file confirmed
- File download and decryption – Matches original file
- Invalid file paths or names – Handled securely
- Directory traversal prevention – Implemented using secure_filename()

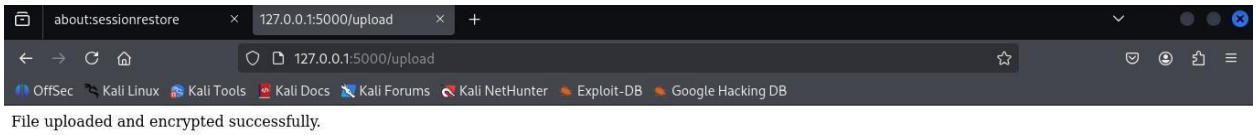
8.Output



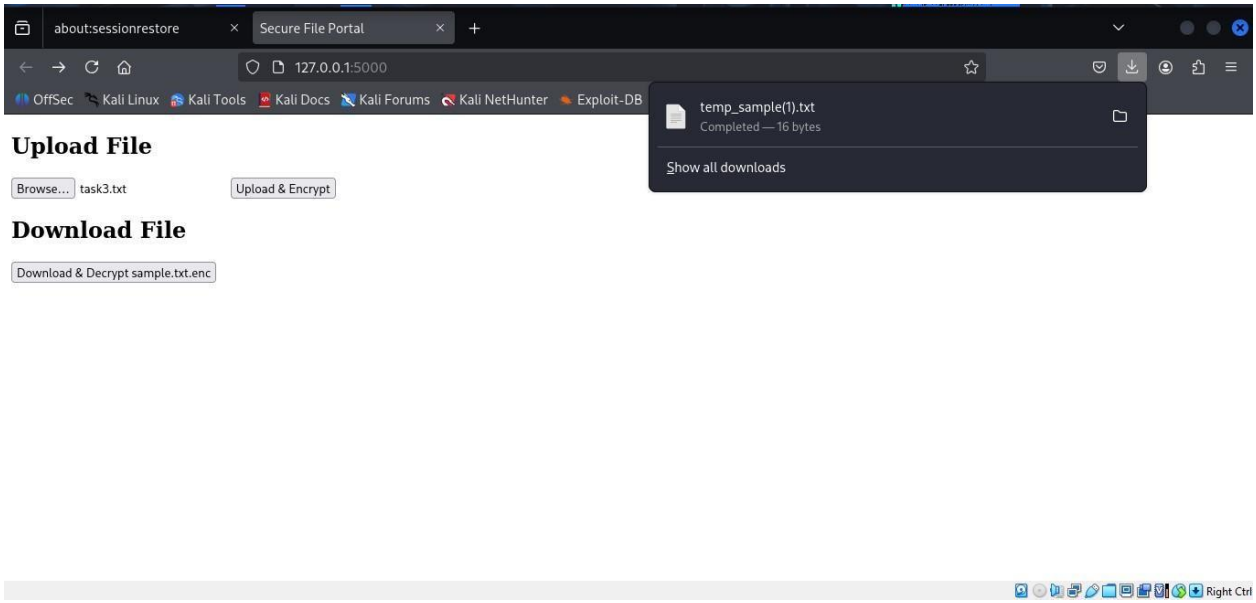
Figure(1):Home Page



Figure(2):Select File to upload



Figure(3):File uploaded successfully



Figure(4):Download the file

8. Security Measures Implemented

- AES Encryption – Files encrypted before storage
- Encrypted Storage – Only encrypted files are saved in the system
- Safe File Handling – Prevents unsafe filenames using secure_filename()
- Decryption On-Demand – Files decrypted only when downloading
- HTTPS Compatibility – Can be used behind HTTPS in production

9. Conclusion

- Demonstrated encryption at rest and in transit
- Gained hands-on experience with Flask and AES integration
- Implemented secure file handling and key management
- Project can be extended with authentication, AES-GCM, and database integration