

Cyber Security Internship – Task 3 Report

Intern Name: Akshita Sharma

Internship Program: Cyber Security – Future Interns

Task Title: Secure File Sharing System

Task Type: Web Development + Encryption

Status:  Completed

GitHub :

https://github.com/AkshitaSharma300/FUTURE_CS_03

Introduction:

In today's digital landscape, secure data sharing is a fundamental need. Unauthorized access to files during transfer or storage can lead to serious data breaches. As part of the Future Interns Cyber Security Internship, this task challenged me to build a secure file sharing web application that protects data using AES encryption. It was designed to ensure confidentiality for files both in transit and at rest.

Task Objective:

Developed a secure portal for uploading and downloading files, where all file data is encrypted using AES (Advanced Encryption Standard). The system should encrypt files during upload and decrypt them securely during download through a simple web interface, thereby ensuring end-to-end confidentiality.

Tools & Technologies Used:

Component	Technology
Programming	Python
Web Framework	Flask
Encryption Library	PyCryptodome (AES)
Frontend	HTML
Version Control	Git & GitHub
Platform	Localhost (Browser-based)

Core Features Implemented:






- **AES Encryption on Upload:** Files are encrypted using AES-128 (CBC mode) with a random key and IV.
- **AES Decryption on Download:** Encrypted files can be decrypted only using the same algorithm and IV structure.
- **Web-Based Interface:** Simple HTML+Flask interface for ease of use.
- **Secure Handling of Files:** Decrypted files are temporarily saved in a separate folder and are overwritten with each use.
-

Security Overview:

- **Encryption Standard:** AES-128 in CBC mode ensures robust symmetric encryption.
- **Key Management:** A new 128-bit key and a unique IV (Initialization Vector) are generated for each file. The IV is stored with the encrypted data to allow correct decryption.

- **Data Padding:** AES requires input data to be a multiple of 16 bytes. PyCryptodome's padding utility securely handles this during encryption and unpadding during decryption.
- **Confidentiality:** Files are unreadable on disk without access to the decryption key. No plaintext files are stored permanently.
- **File Separation:** Encrypted files are stored in the uploads/ folder, while decrypted files are stored temporarily in decrypted/, which can be periodically cleared for hygiene.
- **Attack Surface Minimization:** No sensitive keys are exposed through the user interface. Input validation ensures only existing filenames are processed.
- **Improvements Possible:** For future enhancement, HMAC-based integrity checks and user authentication can be added to prevent tampering and misuse.

Project Outcome:

-  Successfully developed a secure file sharing system
-  Applied encryption techniques in a real-world web app
-  Learned and implemented AES (CBC) encryption/decryption
-  Used Flask to create a clean, functional web portal
-  Uploaded code to GitHub for version control and sharing

Conclusion:

- This task gave me hands-on experience in developing a real-world cybersecurity solution. It deepened my understanding of file-level encryption, secure web design, and best practices for data confidentiality. This project not only fulfilled the internship requirement but also strengthened my foundation as a future cyber security analyst.

