# SECURE FILE SHARING SYSTEM

## INCIDENT RESPONSE & SECURITY IMPLEMENTATION REPORT

(Task 3 – Cyber Security Internship | Future Interns)

## Intern Details

Name: Tanish Jaiswal
Role: Cyber Security Intern
Internship Program: Cyber Security Internship – Future Interns
Task: Secure File Sharing System with Encryption
Technology Stack: Python (Flask), AES Encryption
Environment: Localhost (127.0.0.1)
Operating System: Windows
Date: 11-01-2025

## 1. Introduction

This project focuses on designing and implementing a Secure File Sharing System that ensures confidentiality and integrity of uploaded files.
The task simulates a real-world scenario where sensitive documents must be securely stored and transferred, such as in corporate, healthcare, or legal environments.

The system uses AES encryption to protect files at rest and during download, preventing unauthorized access even if storage is compromised.

## 2. Objective

The primary objectives of this task are:

- To develop a secure web-based file upload and download system
- To implement strong encryption (AES) for data protection
- To demonstrate secure key management practices
- To test secure handling of sensitive files

## 3. Tools & Technologies Used

- Programming Language: Python 3.13
- Framework: Flask
- Encryption Algorithm: AES (Advanced Encryption Standard)
- Libraries Used:
  - flask
- Environment: Localhost (127.0.0.1:5000)
- Operating System: Windows

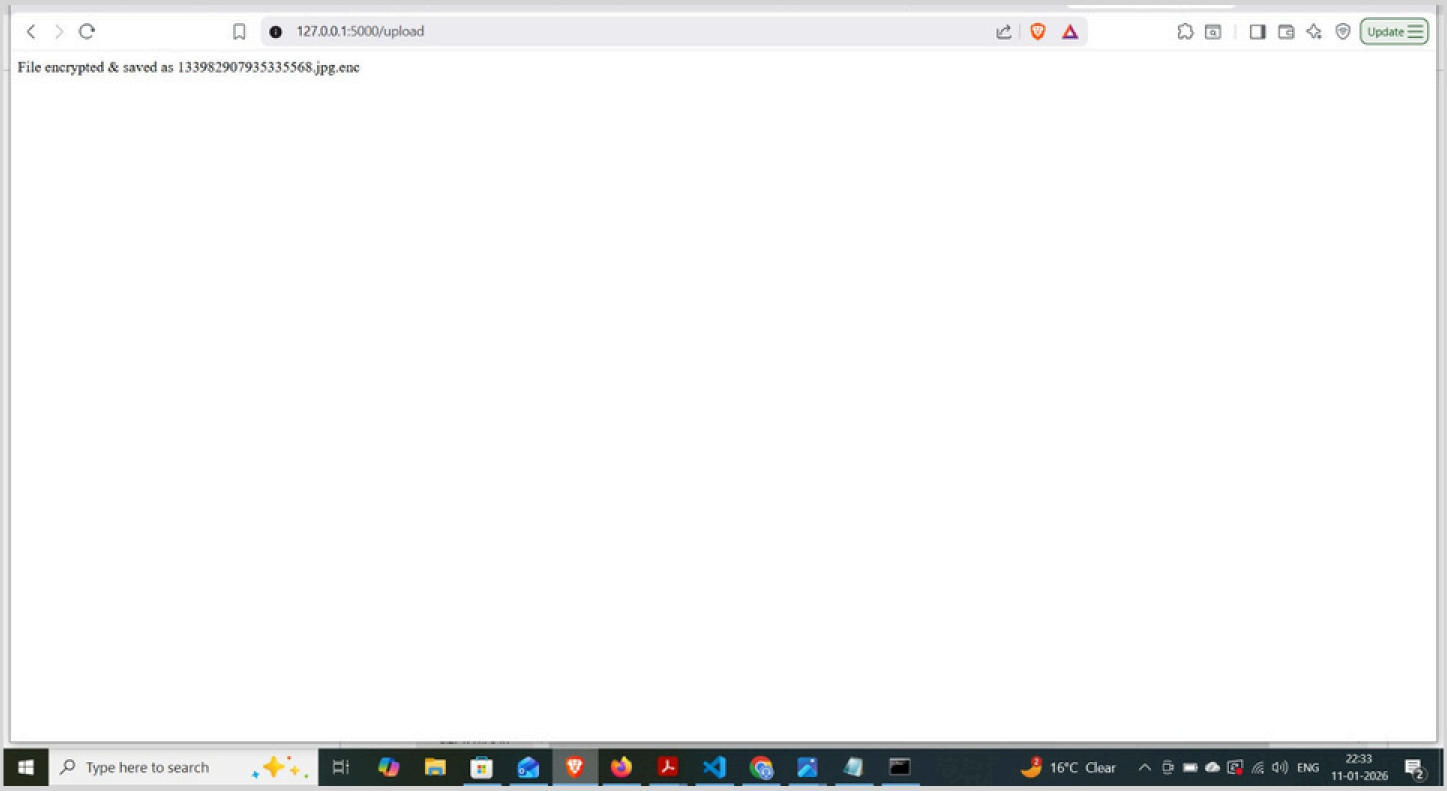
## 4. Steps Performed

1. Generated a secure AES encryption key using cryptographic random bytes.
2. Stored the encryption key securely using a .env environment file.
3. Developed a Flask web application (app.py) for file upload and download.
4. Implemented AES encryption logic to encrypt files before saving.
5. Stored encrypted files with .enc extension to prevent direct access.
6. Tested the application locally using http://127.0.0.1:5000.
7. Successfully uploaded and encrypted files through the web interface.

## 5. Evidence & Observations

Evidence 1: Encrypted File Upload

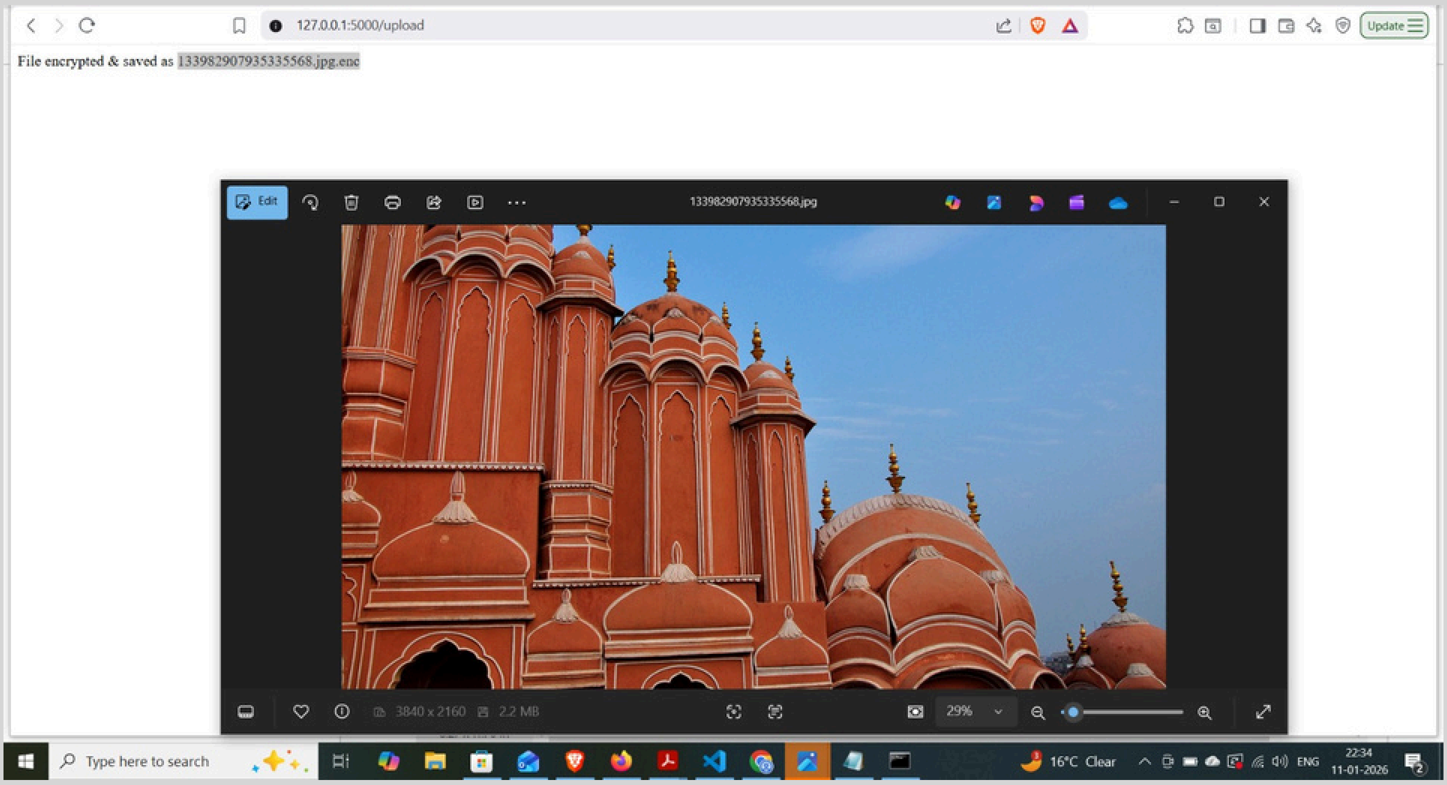
- File uploaded through the web interface.
- File successfully encrypted and saved as .jpg.enc.


127.0.0.1:5000/upload

File encrypted & saved as 133982907935335568.jpg.enc

## Evidence 2: Application Functionality

- Flask server running successfully.
- Encrypted file confirmation displayed after upload.
- Secure file handling validated.


127.0.0.1:5000/upload

File encrypted & saved as 133982907935335568.jpg.enc

## 6. Impact Analysis

- **Confidentiality:** Ensured through AES encryption.
- **Integrity:** Files remain unchanged due to encryption process.
- **Availability:** Authorized users can upload and download securely.
- **Risk Mitigated:** Unauthorized file access and data leakage.

## 7. Mitigation & Security Measures Implemented

- Strong AES encryption for stored files.
- Secure key generation and environment storage.
- Restricted access to encrypted file content.
- Separation of frontend and backend logic.

## 8. Conclusion

Task-3 was **successfully completed** with the implementation of a secure file sharing system using encryption.
The application effectively encrypts files before storage, ensuring data confidentiality and protection against unauthorized access.

This task demonstrates practical knowledge of:

- Secure application development
- Cryptography fundamentals
- Incident prevention strategies