

Security Overview Report


Project Title: SecureShare - Encrypted File Sharing System

Internship Task: Cybersecurity Internship - Task 3 (FUTURE INTERNS)

Developer: Priyanshu Shekhar Choudhury

Objective

To design and implement a secure file upload and download system where all user data is encrypted both at rest and in transit, ensuring only authenticated users can interact with the system securely.

 **Note:** This application is intended for desktop use only, optimized for desktop browsers and workflows.

Encryption Approach

1. Encryption at Rest

To ensure that files stored on the server are protected from unauthorized access or tampering, we use the AES (Advanced Encryption Standard) algorithm in EAX mode. EAX provides both confidentiality and message integrity, making it an ideal choice for secure file storage. Each uploaded file is immediately encrypted upon submission using a shared symmetric 16-byte key. The final encrypted data consists of the nonce, authentication tag, and ciphertext, which are all stored in a single binary `.enc` file.

AES is considered highly secure and is widely adopted in both government and enterprise-level systems. In our system, this ensures that even if an attacker gains access to the server's storage, they will not be able to retrieve usable data without the secret key.

2. Encryption at Transit

Secure transmission of data over the internet is ensured using HTTPS. The application is deployed on a secure platform like Render.com, which automatically provisions TLS certificates. This ensures that all user interactions, including login, file upload, and file download, are encrypted in transit using Transport Layer Security (TLS). This protects against man-in-the-middle (MITM) attacks and packet sniffing.

Having HTTPS in place also builds user trust and meets standard compliance requirements for handling sensitive data.

User Authentication & Access Control

The system includes robust authentication and session handling features to ensure that only authorized users can access the platform. Users register with a unique username and password, which is hashed using Werkzeug's `generate_password_hash()` before storing in the SQLite database.

Flask session management is used to maintain user state across pages. Access to routes like `/dashboard`, `/upload`, `/download`, and `/delete` is restricted to authenticated users only. Unauthorized users attempting access are redirected to the login page with an appropriate flash message.

This ensures a high level of security and user accountability.

File Handling Security

Secure file handling is central to the SecureShare application. All uploaded files are sanitized using `secure_filename()` to remove dangerous characters and ensure compatibility.

Immediately after upload, the files are encrypted and stored with a `(.enc)` extension. This ensures no sensitive data resides unencrypted on the server.

The download process provides encrypted files only. No server-side decryption is performed to keep the application lightweight and secure. Users can optionally decrypt files locally using a custom tool.

File deletion includes a JavaScript-powered popup for confirmation, preventing accidental data loss and adding an extra layer of control for the user.

User Experience & Feedback

A clean and responsive UI has been implemented using Bootstrap. Pages have been designed to support a desktop-first approach, making them highly usable for office or administrative environments.

Flash messages notify users of key actions like registration success, failed login attempts, successful uploads, and deletion confirmations. Validation checks ensure duplicate usernames are not allowed and password fields are filled properly. These features contribute to a user-friendly and secure platform.

Tools & Technologies Used

- Flask (Python Web Framework)
- SQLite (Lightweight Relational Database)
- HTML5, CSS3, Bootstrap (Frontend UI)
- PyCryptodome (AES Encryption)
- Render.com (Deployment with HTTPS)
- Git & GitHub (Version Control & Code Hosting)

Summary

SecureShare is a well-architected, lightweight, and secure file-sharing platform that successfully fulfills the following security and functionality requirements:

- AES-based encryption of files at rest
- HTTPS-based encryption in transit
- Secure user authentication and role-restricted access

- Flash alerts and validation for better user interaction
- Responsive, desktop-optimized user interface

This project is suitable for small organizations and teams looking to exchange documents securely over the web.

✦ **Future Improvements**

- Assign per-user encryption keys for individualized security
- Implement server-side decryption with audit logging
- Integrate Flask-Login or JWT for scalable authentication
- Add malware scanning and file type validation
- Provide real-time email notifications on activity