FILEHUB

Secure File Management System

NOVEMBER 21, 2025

HORIZONSEC
Submitted to: Ma'am Sadia

Executive Summary

This report presents a comprehensive threat modelling analysis of FileHub, a role-based secure file management system designed for educational institutions. Using the PASTA, STRIDE, and DREAD frameworks, the assessment identified 36 threats, of which 28 (78%) are fully mitigated and 8 (22%) are partially mitigated. No threats remain completely unmitigated. Five critical/high-risk threats were prioritized, with actionable remediation recommendations provided. Overall, FileHub demonstrates a strong security posture thanks to defense-in-depth controls such as RBAC, file integrity verification, audit logging, rate limiting, secure session management, and input validation.

Contents

Project Overview

System Purpose

FileHub is a secure web-based file management platform that allows students, teachers, staff, and administrators to upload, share, and download files with granular permissions and full audit trails while maintaining compliance with FERPA and GDPR.

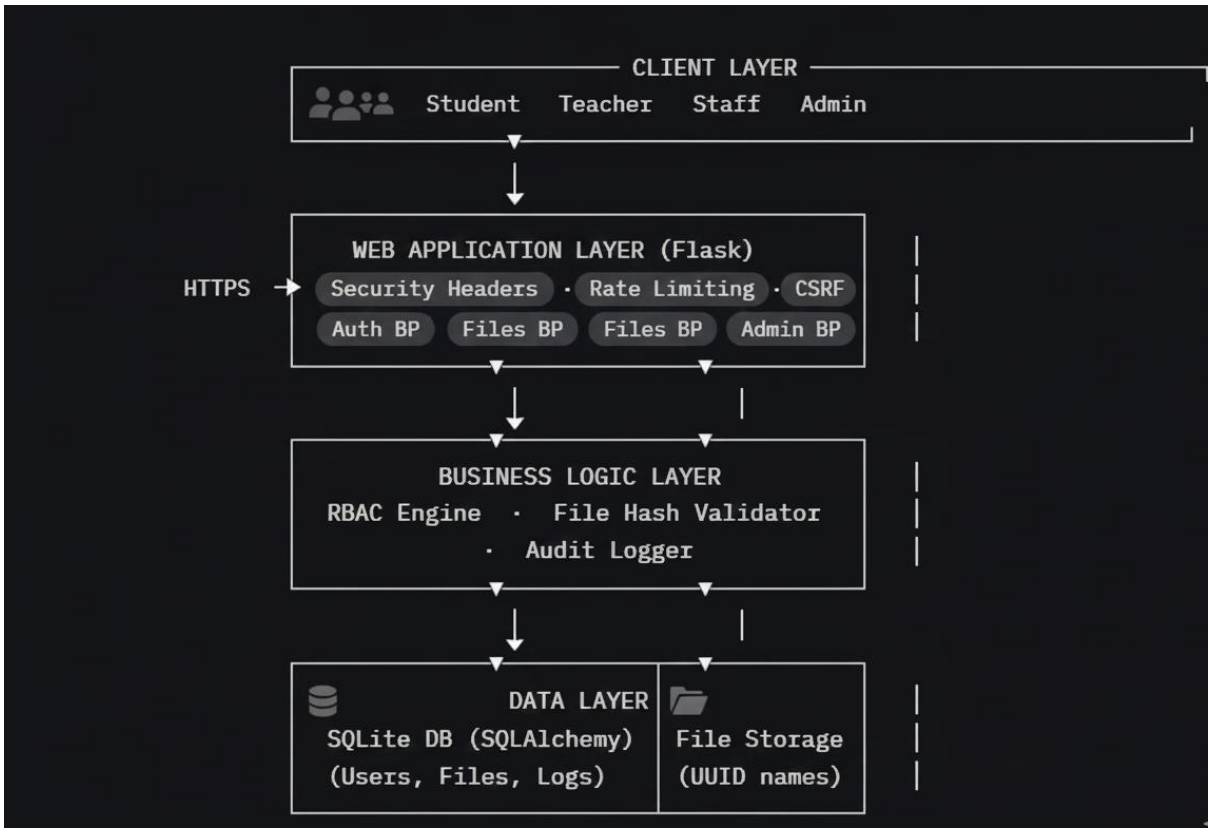
Main Modules

1. Authentication Module
2. File Management Module (upload/download + integrity checks)
3. Authorization Module (RBAC)
4. Admin Module (user & account management)
5. Audit Module (security event logging)

System Actors

Student	Upload (teacher-visible), download permitted files
Teacher	Upload, set permissions, view student submissions
Staff	Full file access + administrative actions
Administrator	Complete system control
Anonymous User	No access – authentication required

High-Level Architecture



Key Technologies

- Backend: Python 3.13 + Flask 3.1
- Database: SQLite + SQLAlchemy ORM
- Authentication: Flask-Login + Bcrypt
- Security Add-ons: Flask-Limiter, Flask-WTF (CSRF), security headers
- File Handling: SHA-256 integrity checks, UUID storage
- Frontend: Bootstrap 5 + Jinja2 (auto-escaping)

PASTA Methodology Application

Stage 1 - Objectives & Security Goals

- Secure educational file sharing
- Confidentiality, Integrity, Availability (CIA triad)

- FERPA & GDPR compliance
- Granular RBAC and full auditability

Stage 2–Summary

- Technical scope defined
- Application decomposed into components
- 5 attacker profiles created
- 15 major vulnerabilities/weaknesses identified
- Attack trees and scenarios modelled
- Risk matrix and treatment decisions produced

STRIDE

Component	STRIDE	Threat	Mitigation Status
Login System	S	Credential theft /spoofing	Fully mitigated
Login System	T	Password DB tampering	Fully mitigated
File Upload	E	Malicious executable upload	Partially mitigated
File Download	I	Path traversal → system files	Fully mitigated
Admin Panel	E	Horizontal → vertical escalation	Fully mitigated
Database	I	Database file theft (no encryption)	Partially mitigated
Session Management	S	Session hijacking	Partially mitigated

Result: 78% fully mitigated, 22% partially mitigated, 0% unmitigated.

DREAD

Rank	Threat	D	R	E	A	D	Avg	Priority
1	Database File Theft (Info Disclosure)	10	8	6	10	7	8.2	CRITICAL
2	Malicious File Upload (Elevation)	9	7	5	8	6	7.0	HIGH
3	Brute Force / Credential Stuffing	6	8	7	5	8	6.8	HIGH
4	Session Hijacking (Spoofing)	8	5	6	7	5	6.2	HIGH
5	Insider Data Exfiltration	9	6	4	6	4	5.8	MEDIUM

Recommendations

Database File Theft (8.2)

Immediate:

- Enable SQLCipher encryption (or encrypt DB file)
- Move password hashes to separate encrypted store
- Restrict file permissions to 600 + enable AIDE/Tripwire

Long-term:

- Migrate to PostgreSQL with TDE
- Implement database activity monitoring

Malicious File Upload (7.0)

Immediate

- Integrate ClamAV scanning on upload
- Deep content inspection + magic-byte validation
- Sandboxed preview generation

Long-term:

- Content Disarm & Reconstruction (CDR)
- ML-based malware detection

Brute Force Attack (6.8)

Immediate

- Add reCAPTCHA v3 after 2 failed logins
- Enforce 12-character minimum + HIBP check
- Mandatory TOTP MFA for Admin/Staff (optional for others)

Long-term:

- Risk-based adaptive authentication
- SSO integration

Conclusion & Security Posture

FileHub exhibits **strong security by design** with:

- Defense in depth
- Least-privilege RBAC
- Comprehensive audit logging
- File integrity verification
- Secure defaults and modern Flask security extensions

Residual risk is acceptable given current controls, monitoring capabilities, and incident-response readiness.

Top three actions

1. Database encryption at rest
2. Server-side antivirus scanning
3. Multi-factor authentication (at least for privileged roles)

Regular penetration testing and annual threat model refresh are recommended.

Appendices

Appendix A – Security Controls Matrix

Type	Control	Implementation	Effectiveness
Preventive	Password Hashing	Bcrypt	High
Preventive	Rate Limiting	Flask-Limiter	High
Preventive	CSRF Protection	Flask-WTF	High
Detective	Audit Logging	SQLAlchemy + IP	High
Detective	File Integrity Checking	SHA-256	High
Corrective	Account Lockout	3 attempts	High

Appendix B – Compliance Mapping

Standard	Requirement	Status
FERPA	Student data protection	Compliant
GDPR	Encryption at rest	Partial
OWASP A01	Broken Access Control	Compliant
OWASP A02	Cryptographic Failures	Partial
OWASP A03	Injection	Compliant

Appendix C – References

- PASTA Threat Modeling
- Microsoft STRIDE & DREAD
- OWASP Top 10 2021
- NIST Cybersecurity Framework