International Journal of Creative and Open Research in Engineering and Management
ISSN: 3108-1754 (Online)
Volume 02 Issue 11 November-2026 | Impact Factor: 3.5



Secure Vault Collaboration Suite: A Role-Based Platform for Secure File Management and Application Security Scoring

Dr. ANUPAMA B S
Department of CSE
Akshaya Institute of Technology, Tumakuru, Karnataka, India
anupamabsait@gmail.com

ANUSHA R N
Department of CSE
Akshaya Institute of Technology, Tumakuru, Karnataka, India
anushanagarajappa7@gmail.com

CHETHANA N
Department of CSE
Akshaya Institute of Technology, Tumakuru, Karnataka, India
Chethanan136@gmail.com

ABHILASH R
Department of CSE
Akshaya Institute of Technology, Tumakuru, Karnataka, India
abhireddyabhireddy114@gmail.com

HEMALATHA H N
Department of CSE
Akshaya Institute of Technology, Tumakuru, Karnataka, India
hemalathahn342@gmail.com

ABSTRACT

How to Cite this Article:
N, A. R., N, C., R, A. & N, H. H. (2026). Secure Vault Collaboration Suite: A Role-Based Platform for Secure File Management and Application Security Scoring. International Journal of Creative and Open Research in Engineering and Management, 02(11), 1-9. https://doi.org/10.55041/ijcope.v2i11.146

License:
This article is published under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.
© The Author(s). Published by International Journal of Creative and Open Research in Engineering and Management.







https://doi.org/10.55041/ijcope.v2i11.146

Teams that exchange documents through personal folders, mail attachments and chat links rarely know which copy is current, who may open it, or who touched it last. This paper presents Secure Vault Collaboration Suite, a web application that gathers registration, sign-in, permission-checked file handling and event recording into one managed workflow. Every protected request follows the same path: session validation, input checks, resource lookup, a server-side permission decision and an audit entry, so hiding a button never stands in for enforcement. Beyond file handling, the suite offers a security directory in which applications are searched, graded on a 100-point scale and broken down across five pillars: encryption, authentication, access control, data protection and vulnerability management. Each assessment closes with prioritised hardening steps. The prototype was built with Python and Flask over a SQLite store and exercised on Windows 11 in Google Chrome. A walkthrough of the working interface shows a dashboard governing twelve applications, nine of which reach the eighty-point adherence threshold, with a fleet average of 82/100. A worked example, a messaging application scoring 72 (Grade C), shows how a pillar profile turns into concrete remediation tasks. The paper also supplies a threat register, a role-permission model and a list of capabilities, including multi-factor authentication and encryption at rest, that remain future enhancements.

INDEX TERMS : Secure file collaboration, role-based access control, least privilege, audit logging, security posture assessment, threat modelling, password hashing.

I. INTRODUCTION

Sharing digital files is routine in coursework, research groups and offices. A team may need several people to draft, review and retrieve the same documents, yet the tools reached for first, such as local drives, mail threads and chat links, give little help in telling which version is current, who is entitled to open it and who changed it. A purpose-built collaboration platform must therefore do more than store and return files: it has to establish identity, decide permissions for each operation and leave a trail that can be reviewed later.



1. Scattered access control: Signing in does not stop one account from fetching another's private file if the server skips an ownership check, and a hidden download button protects nothing if its address still answers.

2. Unsafe uploads and secrets: Unchecked names, paths, sizes and types invite path manipulation and storage abuse, while plaintext passwords or verbose errors leak credentials.

3. Weak traceability: Without structured records of sign-ins, uploads, downloads and refusals, investigating an incident becomes guesswork.

4. Invisible application risk: Encryption, authentication and patch status of internal and third-party applications are seldom compared side by side, so plaintext transport or unsigned webhooks go unnoticed.

To close these gaps, this paper proposes Secure Vault Collaboration Suite, a modular prototype in which the server, not the page, is the final judge of every protected operation, and which pairs that rule with audit logging and a scored application-security directory. The contributions are a layered architecture, a threat register with a simple risk-scoring rule, and an illustrated walkthrough of the working interface.

II. LITERATURE REVIEW

[1] The OWASP Top Ten lists the most frequent web application risks, among them broken access control, authentication failures, insecure configuration, injected input and weak cryptography. Its main lesson here is that risk is spread across the interface, application logic, storage and permission layers, so testing has to cover all of them.

[2] The OWASP File Upload Cheat Sheet recommends extension allow-lists, content-type verification, renamed storage objects, size caps and storage away from executable paths, and it advises against trusting client-supplied names.

[3] The OWASP Authorization Cheat Sheet separates authentication, which confirms who is asking, from authorization, which decides whether that identity may act on a given resource. It promotes least privilege, deny-by-default and a check on every request, which is the rule adopted for view, download, share and delete operations.

[4] The OWASP Password Storage Cheat Sheet advises slow, salted, adaptive hashing, with Argon2id preferred where available, so that a leaked database does not expose usable credentials. [5] NIST SP 800-63 frames digital identity as proofing, authenticator choice and lifecycle management, with assurance matched to risk, and [6] NIST SP 800-53 offers a control catalogue whose access-control, identification, audit and accountability families help structure requirements and test checklists.

[7]–[9] The Flask, SQLite and Python documentation describe the framework, relational store and language features used to realise the modules, including parameterised queries and predictable exception handling. [10] The OWASP Logging Cheat Sheet lists events worth recording, such as authentication attempts, uploads, edits to sharing rules and refused requests, and warns against writing passwords, tokens or file contents to logs.

[11] The OWASP Session Management Cheat Sheet covers identifier handling, cookie attributes and session expiry. [12] The CIS Benchmarks supply hardening baselines, [13] CVSS v3.1 defines how vulnerability severity is rated, [14] RFC 9106 specifies the Argon2 password-hashing function and [15] RFC 8446 specifies TLS 1.3.

None of these sources describes a complete workflow joining identity, permission checks, safe storage, auditing and posture scoring. Secure Vault Collaboration Suite attempts that, with claims limited to what the working build shows.

III. METHODOLOGY

A. System Architecture and Block Diagram

The suite is arranged in four layers. The browser interface sends requests to Flask routes; the security layer resolves the session and applies an authorization policy; the service layer performs file operations, posture scoring and audit recording; and the storage layer keeps metadata in SQLite and file contents in a controlled local directory. Fig. 1 summarises the flow.



Fig. 1: Layered architecture of Secure Vault Collaboration Suite.

B. Authentication and Session Handling:- Registration collects a name, a corporate email address, an organisational unit and a password of at least eight characters entered twice. Passwords are kept only as



salted adaptive hashes, and the sign-in page names Argon2id [14] and bcrypt. The role is assigned by the system and never read from a form field, so nobody can register as an administrator. Failed sign-ins return a single generic message, logout ends the session as session guidance advises [11], and three demonstration personas (Admin, Analyst and Auditor) are offered for quick demonstrations.

C. Authorization and File Operations:- Permission is decided from the role together with a per-file check. A request is refused when the caller is unauthenticated, the request names a missing file, or no permission can be found. Viewers may list and download permitted files, owners may additionally upload, share and delete their own files, and administrators act as authorised. Direct address requests pass through exactly the same check as button clicks. Table I gives the role matrix.

TABLE I. ROLE-PERMISSION MATRIX

Operation	Visitor	Viewer	Owner	Admin
Register account	Yes	No	No	Policy
Sign in	Yes	Yes	Yes	Yes
List permitted files	No	Yes	Yes	Auth.
Upload file	No	Policy	Yes	Auth.
Download file	No	Yes	Yes	Auth.
Change sharing rules	No	No	Own	Policy
Delete file	No	No	Own	Policy
Review audit records	No	No	Own	Auth.

Policy = policy-dependent; Auth. = as authorised; Own = own items only.

D. Upload Validation and Storage:- An upload is checked for size, extension and available content signature, and the display name is sanitised. The server then generates an unpredictable storage key, writes the content to a non-public directory and records the owner, name, size, type and timestamp as metadata. If the metadata write fails, the orphaned content is removed.

E. Audit Logging:- Each event stores an identifier, actor, action, target, timestamp and outcome. Passwords, session tokens and file contents are never written. The dashboard labels its audit stream as an immutable SHA-256 stream.

F. Threat Modelling and Risk Scoring:- Eight initial threats are registered (Table II). Each threat is scored by multiplying how likely it is by how damaging it would be, both on a 1 to 5 scale; products of 1–4 count as low, 5–9 moderate, 10–16 high and 17–25 critical. The band limits are a team convention rather than a standard, and scores are revisited once controls have been tested.

TABLE II. THREAT REGISTER AND PRIMARY CONTROL

Threat	Priority	Primary control
Weak credentials	High	Salted adaptive hashing
Missing permission check	High	Server-side policy on every call
Unsafe upload	High	Size, type and name validation
Path manipulation	High	Server-generated storage keys
Inadequate audit log	Medium	Structured event records
Unprotected transport	High*	HTTPS in deployment
Accidental deletion	Medium	Owner-only delete, confirmation
Sensitive error output	Medium	Generic user messages

*High for networked deployment.

G. Application Security Posture Assessment:- The directory lets a user search applications and narrow the list by category, risk status, minimum score and enforced multi-factor authentication. Each application receives a composite score out of 100 built from five pillars (encryption, authentication, access control, data protection and vulnerability management), a letter grade and a Secure, Moderate or High risk label. The interface states that scores derive from documented criteria, published CVE severity data [13] and CIS benchmark controls [12], and that sample data is labelled as simulated. In the example shown later, the composite equals the mean of the five pillar scores.

H. Test Strategy:- Testing begins with the normal user workflow and then moves to invalid inputs and denied operations. For every case the tester notes a case number, the starting state, the actions taken, the intended result, the observed result and a verdict. A passing upload proves nothing about authorization, so cross-user requests and direct resource requests are tried explicitly, using two test users and a sample document that is not confidential.

IV. RESULTS AND DISCUSSION

A. Experimental Environment

The prototype was run on Windows 11 (64-bit) with Python 3.14, the Flask framework and a SQLite database, using a local directory as file storage and Google Chrome as the client. The build was exercised on 27 October 2026 with non-sensitive sample files and the application's own simulated sample catalogue.

B. Access Screens

Fig. 2 shows the sign-in tab, with a corporate email field, a masked password field, the authenticate button and the three demonstration personas. Fig. 3 shows the

account-creation tab, which adds a name, an organisational unit and a confirmation field, and displays the eight-character minimum beside the password box. Personal details in both captures have been blurred.



Fig. 2: Sign-in screen with demonstration personas.

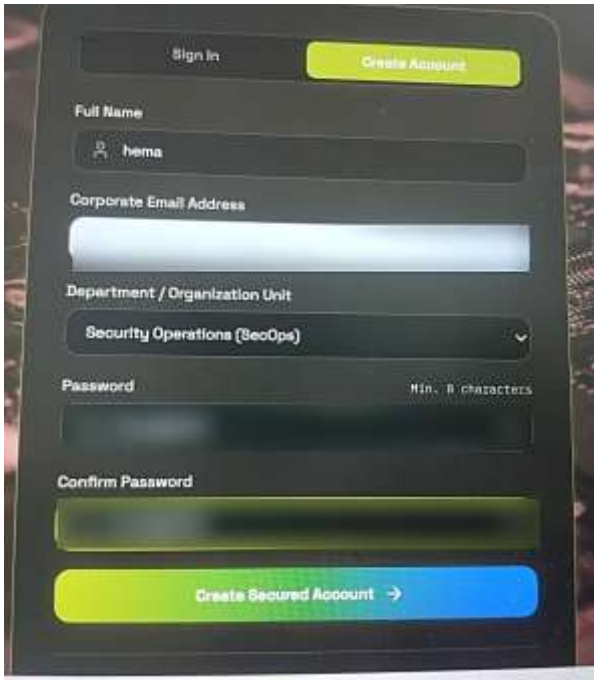


Fig. 3: Account-creation form with password rule.

C. Dashboard and Risk Overview

After sign-in, the dashboard (Fig. 4) greets the security officer and summarises twelve applications under governance, nine classed as secure (75% adherence at a score of 80 or more), three needing attention and an average score of 82/100, a split charted in Fig. 5. A second panel (Fig. 6) lists the weakest workloads: a legacy FTP gateway at 42/100 for plaintext transmission on port 21 and a public webhook bridge at 58/100 for missing HMAC signature verification, each with a stated action. The activity audit panel read "No recent activities recorded" in this capture, so audit visibility should be re-shown with populated events in a later evaluation.



Fig. 4: Home dashboard with posture summary.

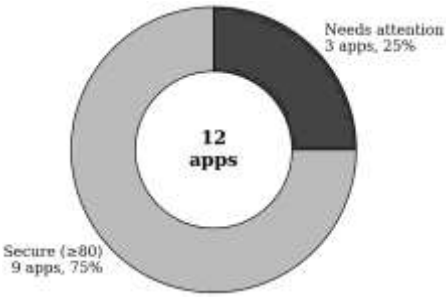


Fig. 5: Share of the twelve governed applications by posture.



Fig. 6: High-risk workloads and recommended actions.

D. Directory, Scoring and Recommendations

The directory (Fig. 7) combines free-text search with category chips and risk, score and MFA filters. Searching for a messaging application opens its report (Fig. 8): a composite of 72, Grade C, Moderate Risk, last evaluated on 12 October 2026. The five-pillar matrix (Fig. 9), plotted in Fig. 10, shows encryption as the strongest pillar and data protection as the weakest, which is where remediation effort should go first.



Fig. 7: Application security directory and filters.



Fig. 8: Application report with composite score and grade.



Fig. 9: Five-pillar security matrix for the selected application.

Fig. 10 plots the same scores. The spread between the strongest pillar (encryption, 82) and the weakest (data protection, 65) is 17 points, and three pillars fall below the mean of 72: access control at 70, vulnerability management at 68 and data protection at 65. A single composite would hide this uneven profile, which is why the pillar view is shown beside it.

Reading the profile against the 80-point adherence threshold used on the dashboard gives a simple way to rank the work. Data protection is 15 points short of the line, vulnerability management 12, access control 10 and authentication 5, while encryption already clears it, so the four weaker pillars carry a combined shortfall of 42 points. Ordering tasks by shortfall puts data protection first, followed by vulnerability management, access control and authentication.

As an illustration only, if each of the four weaker pillars were lifted exactly to 80 while encryption stayed at 82, the mean would become 80.4 and the application would cross the threshold. This is a worked calculation, not a measured result, but it shows why the pillar view is more actionable than the composite alone: a team can see which control family to improve, and by how much, before the grade changes.

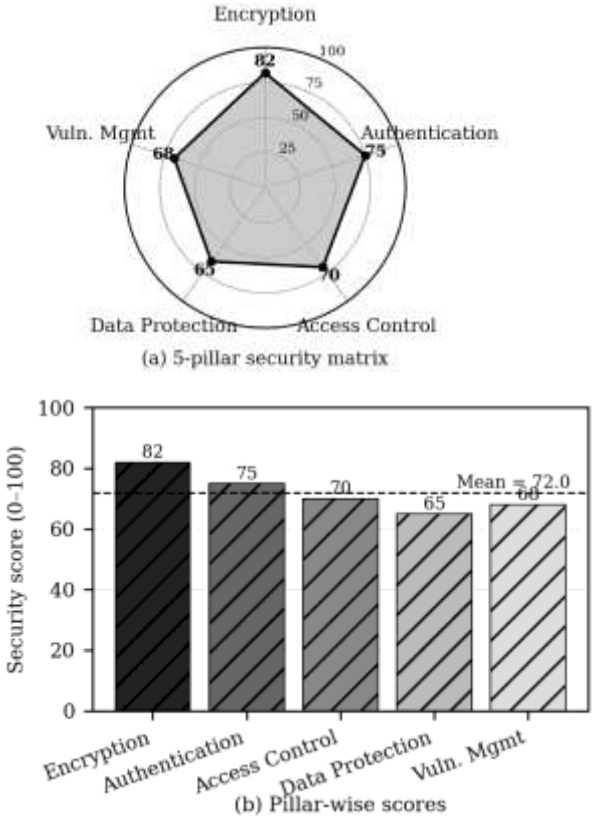


Fig. 10: Pillar scores of the example application as (a) a radar plot and (b) a bar chart with the mean of 72.

Two high-priority hardening records follow (Fig. 11): enabling end-to-end encrypted cloud backups and enforcing a six-digit two-step verification PIN. Both appear with a priority tag, a pending status and the exact remediation path.



Fig. 11: Tailored hardening recommendations.

E. Discussion and Limitations

The evidence offered here is a functional walkthrough, not a benchmark, so no throughput or accuracy figures are claimed. Scores come from the application's simulated sample data, as its own disclaimer states. Several capabilities named in the interface, including TLS 1.3 [15], AES-256-GCM and MFA enforcement, need independent verification before they are treated as implemented, while multi-factor sign-in, encryption at rest, end-to-end encryption, version history, malware scanning and cloud deployment are planned as future



work. Negative tests, such as one account requesting another account's file, should be recorded with expected and actual outcomes in the next evaluation round.

Read against the reviewed guidance, the prototype is designed to address broken access control [1], [3], upload risks [2], credential risks [4] and accountability [6], [10]; code-level review and independent testing would strengthen each claim.

V. CONCLUSION

This paper presented Secure Vault Collaboration Suite, a layered prototype that combines authenticated access, permission checks made on the server, safe upload handling and structured audit records with a scored application-security directory. The threat register and risk rule make weaknesses visible early, and the interface walkthrough shows how a pillar profile becomes ranked remediation work. The prototype should not yet be called production-ready: tighter identity controls, encryption, version tracking, file scanning and scalable deployment remain to be built and tested.

REFERENCES

[1] OWASP Foundation, "OWASP Top Ten web application security risks." [Online]. Available: https://owasp.org/www-project-top-ten/

[2] OWASP Foundation, "File Upload Cheat Sheet." [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html

[3] OWASP Foundation, "Authorization Cheat Sheet." [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

[4] OWASP Foundation, "Password Storage Cheat Sheet." [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

[5] National Institute of Standards and Technology, "Digital Identity Guidelines," NIST SP 800-63. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/63/4/final

[6] National Institute of Standards and Technology, "Security and Privacy Controls for Information Systems and Organizations," NIST SP 800-53, Rev. 5. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final

[7] Pallets Projects, "Flask documentation." [Online]. Available: https://flask.palletsprojects.com/

[8] SQLite Consortium, "SQLite documentation." [Online]. Available: https://www.sqlite.org/docs.html

[9] Python Software Foundation, "Python 3 documentation." [Online]. Available: https://docs.python.org/3/

[10] OWASP Foundation, "Logging Cheat Sheet." [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html

[11] OWASP Foundation, "Session Management Cheat Sheet." [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

[12] Center for Internet Security, "CIS Benchmarks." [Online]. Available: https://www.cisecurity.org/cis-benchmarks

[13] FIRST.Org, Inc., "Common Vulnerability Scoring System v3.1: Specification Document." [Online]. Available: https://www.first.org/cvss/v3.1/specification-document

[14] A. Biryukov, D. Dinu, D. Khovratovich, and S. Josefsson, "Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications," RFC 9106, Internet Engineering Task Force, Sep. 2021. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9106

[15] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Internet Engineering Task Force, Aug. 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8446