A Metadata-Driven Secure Data-Sharing Model for Controlled Access to Proprietary Information

Eddie Cabán Ferrer
Master in Computer Science
Advisor: Jeffrey Duffany, Ph.D.
Polytechnic University of Puerto Rico
Graduate Project EXPO, May 2026

Abstract *— A metadata-driven secure data-sharing model is proposed for controlling access to proprietary information exchanged between internal enterprise sources and external collaborators. The framework was designed to show that secure sharing can be governed through identity-aware authorization, tags-based policy enforcement, protected file storage, and persistent auditing rather than through static permissions or vendor-specific platforms. The implemented system integrates an identity provider, a policy evaluation engine, a relational database for metadata and audit records, object storage for protected files, and an application layer that coordinates token validation, authorization queries, storage access, and request measurement. Evaluation was conducted in three stages: a controlled baseline, a moderate workload stage, and a larger workload stage. Across these stages, the framework maintained correct access-control behavior, prevented unauthorized proprietary downloads, and preserved complete audit capture for protected actions. The results support the feasibility of a vendor-independent middleware approach for deterministic, auditable, policy-directed control of sensitive document sharing.*

Keywords *— Audit Logging, Cybersecurity, Data Access Control, Metadata-Driven Authorization.*

INTRODUCTION

Organizations increasingly depend on digital systems to create, manage, and exchange proprietary technical and business information. The literature on enterprise information management shows that proprietary information control must be understood as part of a broader organizational process that spans engineering, operational, and collaborative environments rather than as a simple file-storage problem [1]. In practice, that information rarely remains inside a single application or department. Instead, it moves across engineering, operational, and collaborative environments where different stakeholders require different levels of visibility and control. This fluid movement suggests that secure sharing must be treated as a multifaceted information governance challenge rather than a simple file transfer problem.

Current discourse in the field emphasizes a shift toward identity aware authorization and metadata-driven control, where proprietary files are classified at the point of intake and managed through protected object storage. One of the motivators for this shift is the mitigation of insider threats and accidental data leakage, which are often exacerbated by static permission models that fail to adapt to the data's lifecycle or context. Policy-enforced gateways and deterministic authorization provide the foundation for modeling the rigorous governance found in industrial Product Lifecycle Management (PLM) and Enterprise Resource Planning (ERP) systems. Furthermore, a robust emphasis on auditability ensures that every interaction with a proprietary file leaves a chronological, tamper-evident trail. This traceability is essential for forensic analysis and compliance verification.

As a result, this approach places the management of sensitive data at the critical intersection of secure system design, auditable enforcement, and strategic enterprise oversight. In this setting, identity-aware authorization, configuration-driven enforcement, and structured auditability emerge not merely as technical features, but as core governance mechanisms for controlling

sensitive information across organizational boundaries [2] [3] [4] [5].

PROBLEM STATEMENT

Businesses frequently need to share proprietary technical and corporate information with internal users, suppliers, contractors, and partner entities. This sharing often occurs through fragmented systems that do not combine authentication, authorization, object storage, and auditability in a consistent manner. Consequently, organizations face recurring risks including unauthorized disclosure, over-permissioned access, inconsistent policy enforcement, limited visibility into who accessed what, and difficulty proving compliance or accountability after an incident or review. Current literature indicates a gap between high-level enterprise governance needs and compact, testable implementations of secure proprietary data sharing. Prior work on attribute-based access control and governance-oriented security suggests that effective control depends not only on authenticated identity, but also on the consistent use of resource tags and evidentiary audit records to support deterministic and accountable access decisions [2] [3] [6]. This project responds to such gap by modeling a focused set of secure-sharing functions using an open-source software stack. In this design, authenticated users receive identity-bearing tokens, file objects are classified and associated with required access attributes, metadata and audit evidence are stored authoritatively, policy decisions are evaluated deterministically before release, and all successful and denied actions are recorded for later analysis.

The objective is not to replicate a full commercial enterprise platform, but to model key subsets of the governance and control functions in PLM, ERP, and collaboration-oriented environments. The project is grounded in the idea that secure sharing must be treated as an information-governance problem rather than a simple file-transfer problem. Accordingly, the project is grounded in the view that secure proprietary sharing must be treated as a governance and enforcement problem, in which trusted identity attributes, protected resource tags, and auditable policy decisions are combined in a compact, testable implementation [2] [3] [4] [7].

METHODOLOGY

The model was implemented on a single Windows laptop using Docker Desktop for containerized services and a local SeaweedFS process for object storage (Figure 1). The implemented software stack consists of FastAPI, Keycloak, Open Policy Agent (OPA), PostgreSQL, SeaweedFS, and OWASP ZAP. Within this architecture, FastAPI serves as the orchestration layer that validates identity context from Keycloak, retrieves file metadata from PostgreSQL, submits authorization inputs to OPA, and, when permitted, brokers object access through SeaweedFS. Table 1 summarizes the principal software components and their roles in the framework.

Table 1
Software Stack Components and Their Roles in the Framework

Component	Role	Why It Matters
FastAPI	Application gateway and API layer	Implements upload, metadata, download, audit, and identity-check endpoints.
Keycloak	Identity provider	Issues user tokens and centralizes user and role management.
OPA	Policy engine	Evaluates decisional allow/deny decisions from token and file metadata.
PostgreSQL	Metadata and audit store	Stores file records, request metrics, and audit events.
SeaweedFS	Object storage	Persists uploaded files behind an S3-compatible interface.
OWASP ZAP	Security testing tool	Provides passive scanning and baseline web-surface validation.

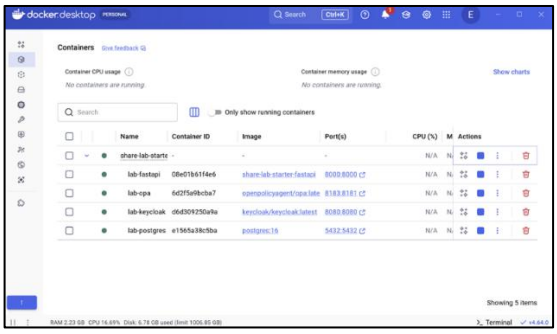
The software stack was instantiated as a hybrid local deployment in which Docker Desktop started FastAPI, Keycloak, Open Policy Agent, and

PostgreSQL as coordinated services, while SeaweedFS was launched separately on the Windows host as a weed.exe mini process for S3-compatible storage. PostgreSQL was provisioned as the project's persistent relational backend, and Keycloak was redirected from its default embedded development store to PostgreSQL through database environment settings so that realm, client, and user definitions would persist across restarts. This arrangement allowed the application, identity provider, policy engine, and relational backend to be managed together through the compose configuration while leaving object storage available as a host-level service with persistent local data retention.

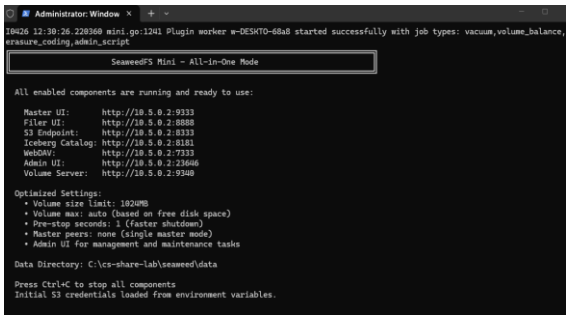
The identity and policy services were then prepared for use within that environment. Within Keycloak, the project realm, application client, test-user accounts, and role assignments were created to support the intended access scenarios. Open Policy Agent was started with the project Rego policy mounted into the container at launch, which allowed policy changes to be applied from the host-side source file without rebuilding the service image. In parallel, PostgreSQL remained available as the shared backend for application informational tags and persistent service state.

The application and storage layers were then initialized through explicit startup settings. FastAPI was supplied with environment variables defining the identity-provider context, policy endpoint, relational database connection, and object-storage endpoint, while SeaweedFS was started with local S3 settings and persistent host-side storage. During application startup, the required storage bucket was verified or created before testing began so that file operations targeted a known object-storage location. Once these service dependencies were reachable through their assigned local ports, the environment was ready for the framework workflow and later staged evaluation.

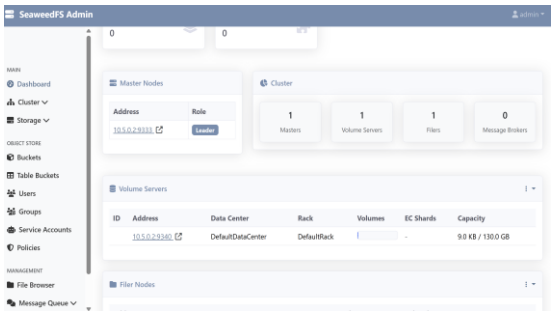
The selected stack was practical for the project because it avoided paid licensing costs, ran on commodity hardware, and isolated the central research contribution in the middleware and policy logic rather than in heavy enterprise infrastructure engineering. In this project, feasibility implies practical implementation and measurable security-governance behavior in a local framework environment rather than to production-scale enterprise deployment or full PLM/ERP integration.



(a) Docker Desktop Containerized Services



(b) SeaweedFS Local Startup Interface



(c) SeaweedFS administrative web interface

Figure 1
Local Deployment and Storage-Service Interfaces for the Implemented Model

The evaluation methodology was intentionally staged to separate initial correctness validation from progressively larger workload execution. First, the project established a fixed Baseline using a small seeded document set and a controlled correctness matrix to verify that authentication, authorization, storage access, and audit behavior matched the

intended design under directed test conditions. Second, after the instrumentation layer was added, the project executed Tier 1 to determine whether the same policy and audit behavior remained stable under a moderate multi-user workload. Third, the evaluation was extended to Tier 2, which increased the number of users, documents, and operations while preserving the same measurement structure so that results could be compared consistently across stages. This staged design allowed the methodology to distinguish between functional correctness, workload scalability, and comparative performance behavior within a single paradigm environment.

Framework Workflow

A user first authenticates through Keycloak and receives a signed JSON Web Token (JWT), which serves as the secure carrier of both identity and authorization-relevant claims. In this framework, the JWT payload contains the identity-aware attributes that drive policy evaluation, including username, role membership, department, and clearance level, in addition to standard issuance and expiration fields. These claims are later combined with request context and file metadata during authorization, allowing authentication and attribute delivery to occur through a single cryptographically protected token issued by the identity provider. A representative JWT payload used in this environment is shown below.

JSON

```
{
  "iss": "https://keycloak.local/realms/your-realm",
  "sub": "user-uuid-12345",
  "preferred_username": "jsmith",
  "email_verified": true,
  "roles": ["contractor", "external_partner"],
  "department": "Network Security",
  "clearance_level": "Level_2",
  "iat": 1714342400,
  "exp": 1714346000
}
```

When a user submits an upload, metadata, or download request, the FastAPI gateway validates the JWT signature using Keycloak's public key, confirms that the token is still valid, and extracts the relevant claims from the payload. FastAPI then reads the corresponding file metadata from PostgreSQL, such as classification or required access role, and constructs a structured authorization input for Open Policy Agent (OPA). OPA evaluates this input against the project's Rego policies by comparing the trusted claims carried in the JWT with the file-specific metadata stored by the application. If the policy evaluation returns allow, FastAPI either returns meta-information or retrieves the requested object from SeaweedFS and streams it to the requester; if the result is deny, the request is rejected. In both cases, policy-relevant file actions are written to the audit log. Figure 2 summarizes how the principal services in the framework interact to support authentication, claims validation, policy evaluation, metadata retrieval, protected object storage, and audit logging.

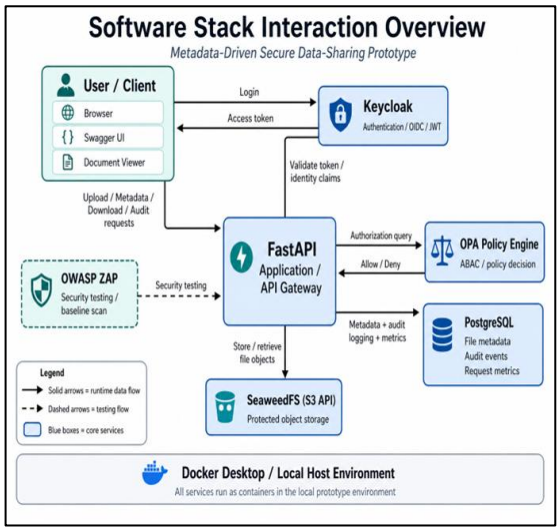
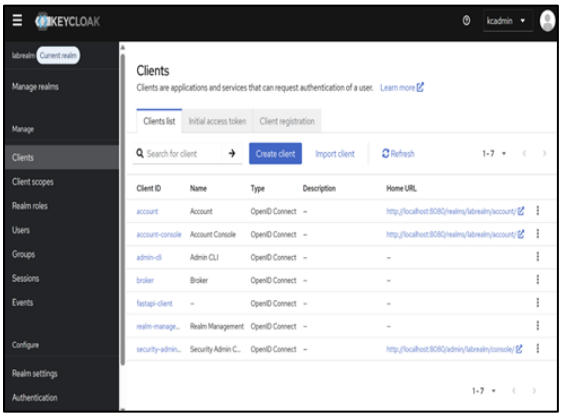


Figure 2
Software Stack Interaction Overview for the Metadata-Driven Secure Data-Sharing Model

This interaction is predetermined because the authorization result is always derived from the same combination of trusted identity claims and stored resource metadata. For any given request, the same JWT attributes and the same file informational tags will therefore produce the same binary outcome: allow or deny. Because Keycloak cryptographically signs the token, the application and OPA can treat claims such as department, role, and clearance level

as authoritative inputs rather than as user-supplied values. FastAPI thus acts as the integration layer that binds together identity from Keycloak, resource metadata from PostgreSQL, policy logic in OPA, protected object storage in SeaweedFS, and audit recording in PostgreSQL. This arrangement supports a governance-oriented access model in which decisions are based not on simple file permissions alone, but on a multidimensional comparison between authenticated identity attributes and protected resource metadata.



(a) Keycloak Realm and Client Administration Interface

(b) FastAPI Swagger/OpenAPI Interface for Protected Endpoints

Figure 3
Representative Identity-Management and API Interfaces Used in the Framework Workflow

Instrumentation

To support quantitative analysis, the application was extended with request-level timing middleware and a dedicated request_metrics table. For each request, the system records the endpoint, method, status code, latency, user identity when available, action type, policy decision, and file identifier. Protected actions such as upload, metadata retrieval, and download are also recorded separately in *audit_events*, allowing the project to distinguish between performance measurement and evidentiary security logging. For cross-stage latency comparison, request paths containing file-specific identifiers were normalized to shared route classes, and latency statistics were then recomputed from the raw per-request measurements within each normalized endpoint, method, decision, and status-code group. Together, these instrumentation elements provide the measurement foundation used in the Baseline, Tier 1, and Tier 2 evaluations.

The Baseline stage used four seeded files: two public files and two proprietary files uploaded by predefined personas. The baseline matrix then exercised authorized proprietary access, unauthorized proprietary access, public-file access, audit retrieval, no-token access, expired-token access, and role-mutation behavior. This stage was intentionally small and controlled so that correctness could be verified before introducing higher request volume and more complex multi-user activity. Its purpose was to establish that authentication, authorization, storage access, and audit behavior matched the intended design under directed test conditions.

The Tier 1 workload expanded the evaluation to 10 users, 100 uploaded files, and 331 client-side operations. A dedicated benchmark client automated file generation, upload, metadata checks, download attempts, audit retrieval, and identity checks while preserving the same access-control logic validated during Baseline. Tier 1 therefore served as the first operational workload stage: large enough to produce measurable request, latency, and audit data, but still limited enough to expose anomalies and support controlled debugging before further scaling.

The Tier 2 workload preserved the Tier 1 benchmark structure but increased the scale to 20 users, 500 uploaded files, and 1,520 client-side operations. The same client workflow was retained

so that Baseline, Tier 1, and Tier 2 remained directly comparable: upload paths, tag requests, download attempts, audit retrieval, and identity checks were all executed under the same policy model, with privileged users assigned the proprietary_access role and unprivileged users left without it. Tier 2 was therefore the first stage intended to test whether the model could sustain a substantially larger protected document set and request volume without changing the middleware design or authorization logic.

EVALUATION AND RESULTS

The subsequent evaluation verifies that the metadata-based policy and audit mechanisms maintain zero instances of unauthorized access and full audit capture as operational scale grows, validating the deterministic reliability of the system across three escalating workload phases.

Baseline

The Baseline stage consisted of 25 controlled cases. All 25 cases produced the expected outcomes, yielding an observed policy accuracy of 100.00%. Authorized users successfully retrieved metadata and downloaded proprietary files when their role assignments satisfied policy, while unprivileged users were denied access to proprietary content. Public files remained accessible to unprivileged users. Unauthenticated and expired-token scenarios correctly returned 401 Unauthorized responses. Role-mutation behavior also matched the intended design: after owner1 lost the proprietary_access role, access to previously uploaded proprietary content still succeeded because of the uploader-override rule, while a new proprietary upload was denied and a new public upload was allowed.

Table 2
Baseline Correctness Summary Across Controlled Case Categories

Category	Cases	Observed Result	Outcome
Seed uploads	4	4/4 succeeded	Pass
Authorized proprietary access	8	All returned 200 and allowed	Pass
Unauthorized proprietary access	5	All returned 403 and denied	Pass
Public-file access	4	All returned 200 and allowed	Pass
Audit endpoint	1	Returned 200	Pass
No-token / expired-token cases	3	All returned 401	Pass
Role-mutation behavior	4	Matched uploader-override design	Pass

All Baseline cases matched the expected outcomes recorded in the correctness matrix, with no observed false allows or false denies in the executed cases.

Tier 1 Workload

The Tier 1 workload produced 100 uploaded documents, 331 client-side logged operations, 332 server-recorded requests, and 300 audit rows, which is consistent with the designed mix of audit-eligible actions. The observed client pass rate was 99.70% (330/331). The single client-side failure occurred on the /me endpoint and was caused by a token-audience mismatch rather than by an authorization-policy error. Across the protected file operations, Tier 1 preserved the core Baseline findings: unauthorized proprietary downloads succeeded in 0.00% of measured cases, false allow and false deny rates remained at 0.00%, and audit completeness for audit-eligible actions remained 100.00%. Tier 1 produced a moderate multi-user workload dataset under the same policy model and audit structure used in Baseline.

Tier 2 Workload

The Tier 2 workload increased the evaluation to 500 uploaded documents and 1,520 client-side operations, while preserving a 100.00% client pass rate (1,520/1,520). The server recorded 1,520 requests, of which 1,040 were allowed and 480 were denied, with 0 authentication failures in the clean rerun. Audit logging also remained consistent at scale: the system produced 1,460 audit rows, exactly matching the number of audit-eligible protected actions generated during the run. Most importantly,

Tier 2 preserved the central security result of the project: unauthorized proprietary download success remained 0.00% under the larger workload. Tier 2 produced the largest completed workload dataset in the evaluation, consisting of 500 uploaded documents, 1,520 client-side operations, and 1,460 audit rows under the same middleware and authorization configuration used in the earlier stages.

Comparative Evaluation of Baseline, Tier 1, and Tier 2

Across the three completed stages, the recorded outcomes remained consistent in terms of policy correctness, unauthorized-access prevention, and audit completeness. Baseline established decisional correctness under controlled conditions, while Tier 1 and Tier 2 demonstrated that the same policy and audit behavior remained stable under increasingly larger workloads. Across all completed stages, the core security findings remained unchanged: false allow rate = 0.00%, false deny rate = 0.00%, unauthorized proprietary download success rate = 0.00%, and audit completeness = 100.00%.

Table 3
Comparative Outcome Summary for Baseline, Tier 1, and Tier 2

Metric	Baseline	Tier 1	Tier 2
Directed / client-side cases	25	331	1,520
Documents uploaded	4	100	500
Client pass rate	100.00%	99.70%	100.00%
Unauthorized download success rate	0.00%	0.00%	0.00%
False allow rate	0.00%	0.00%	0.00%
False deny rate	0.00%	0.00%	0.00%
Audit completeness	100.00%	100.00%	100.00%
Audit rows recorded	25	300	1,460
Server-side requests recorded	111	332	1,520
Authentication failures (401)	3	2	0
Notes	Directed interactive baseline	One /me mismatch only	Normalized Tier 2 rerun

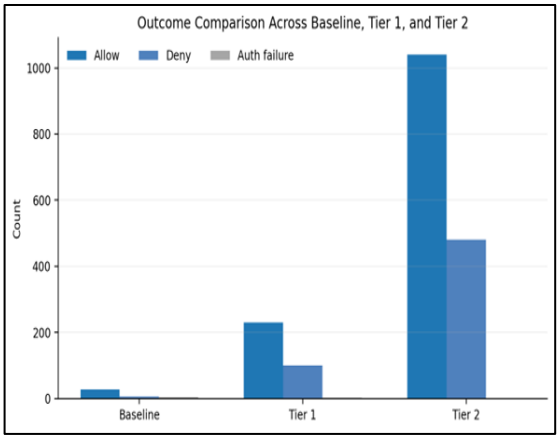


Figure 4
Outcome Comparison Across Baseline, Tier 1, and Tier 2

The comparison results also show that Tier 2 remained internally coherent at the decision level. The clean rerun produced 1,040 allowed responses and 480 denied responses, with 0 authentication failures, confirming that the model continued to admit authorized requests and block unauthorized protected actions even after the workload increased by a factor of five in uploaded documents relative to Tier 1. The Tier 2 audit total of 1,460 rows is especially significant because it exactly matches the number of audit-eligible protected actions in the run, confirming that protected operations were still fully captured in the audit trail under heavier use.

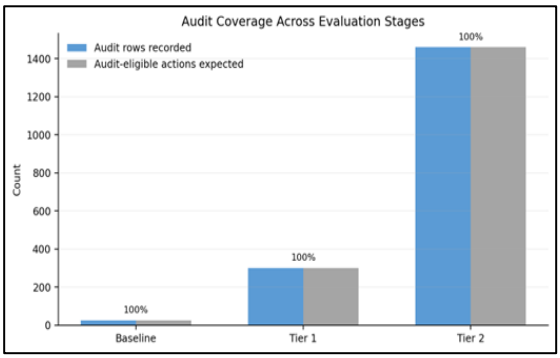


Figure 5
Audit Rows Recorded Versus Audit-Eligible Actions Across Baseline, Tier 1, and Tier 2

From a performance perspective, the normalized latency classes show a measurable but controlled increase from Tier 1 to Tier 2. Average upload allow latency increased from 40.81 ms to 74.18 ms; metadata allow from 24.30 ms to 50.71 ms; metadata deny from 23.77 ms to 51.04 ms;

download allow from 31.10 ms to 67.19 ms; download deny from 24.18 ms to 50.85 ms; and audit retrieval from 9.11 ms to 19.29 ms. Tier 2 median and p95 values remained bounded for the main protected operations, including upload allow at 74.27 ms median and 92.16 ms p95, metadata allow at 52.35 ms median and 62.63 ms p95, metadata deny at 52.61 ms median and 64.10 ms p95, download allow at 67.54 ms median and 83.06 ms p95, and download deny at 51.41 ms median and 63.81 ms p95. Although Tier 2 latency was higher than Tier 1 across the normalized protected endpoint classes, the measured values remained practical for a single-laptop proof-of-concept and did not alter the correctness of the authorization outcomes.

Table 4
Normalized Latency Comparison Across Baseline, Tier 1, and Tier 2

Action Class	Baseline Avg (ms)	Tier 1 Avg (ms)	Tier 2 Avg (ms)	Tier 2 Median (ms)	Tier 2 p95 (ms)
Upload allow	78.61	40.81	74.18	74.27	92.16
Metadata allow	151.95	24.30	50.71	52.35	62.63
Metadata deny	32.82	23.77	51.04	52.61	64.10
Download allow	50.35	31.10	67.19	67.54	83.06
Download deny	27.95	24.18	50.85	51.41	63.81
Audit allow	25.43	9.11	19.29	20.95	23.26
/me allow	2.32	2.02	2.76	2.17	3.36
Auth failure (/me)	n/a	159.97	n/a	n/a	n/a

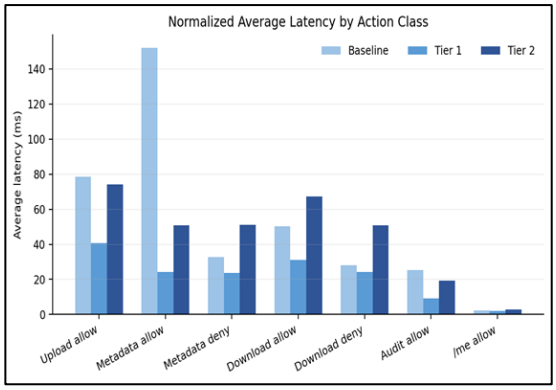


Figure 6
Normalized Average Latency by Action Class Across Baseline, Tier 1, and Tier 2

Across the three completed stages, the model preserved zero observed false allows, zero observed false denies, zero unauthorized proprietary download successes, and full audit completeness, while Tier 2 introduced higher but still bounded latency relative to Tier 1.

DISCUSSION

The implemented methodology moved from a correctness-first Baseline to two progressively larger operational workloads without changing the underlying access-control model. This progression is significant because it shows that the project did not rely on one-off demonstrations or manually curated success cases. Instead, the same metadata-driven authorization logic was carried from directed verification into Tier 1 and then into Tier 2, where it continued to enforce deterministic, auditable access control at larger scale.

The combined Baseline, Tier 1, and Tier 2 evaluation supports three conclusions. First, the framework was functionally correct for the tested access-control scenarios, including authorized proprietary access, unauthorized proprietary access, expired or invalid token handling, and audit capture. Second, the framework remained reliable as the workload increased from a 25-case correctness matrix to 100 uploaded files and then to 500 uploaded files without introducing observed false allows or unauthorized proprietary download successes. Third, the measured latency increase at Tier 2 indicates real scaling cost, but not a breakdown in the architecture or policy model.

The strongest security result is that unauthorized proprietary downloads succeeded in 0% of measured cases across all evaluated stages. This means the middleware framework did not simply log suspicious activity after the fact; it actively prevented unauthorized retrieval before object delivery while continuing to record auditable evidence of the attempted action.

The Baseline metrics were collected in a directed interactive setting and therefore include small-sample interactive effects, while Tier 1 and

Tier 2 were structured workload runs and are more suitable for direct latency comparison. The single Tier 1 /me authentication mismatch should likewise be interpreted as an implementation or configuration artifact rather than as a policy-logic failure, since the protected file decisions recorded during Tier 1 remained correct. These constraints do not invalidate the broader findings, but they do define the appropriate scope of the performance claims.

Overall, the comparative evaluation supports the project's central feasibility claim. Baseline established correctness under controlled conditions, Tier 1 indicated that the same logic remained stable under moderate workload conditions, and Tier 2 showed that the model could scale to a substantially larger protected document set and request volume without introducing incorrect authorization outcomes or audit loss. Although latency increased at Tier 2, this increase does not negate the central result that the framework can enforce fixed, auditable, metadata-driven access control in a realistic local proof-of-concept environment.

CONCLUSION

This project demonstrated the feasibility of a metadata-driven secure-sharing framework implemented as a working middleware model using an open-source, vendor-independent software stack. The completed system integrated Keycloak for identity management, Open Policy Agent for deterministic policy enforcement, PostgreSQL for metadata and audit persistence, SeaweedFS for protected object storage, and FastAPI as the coordinating application layer. Across Baseline, Tier 1, and Tier 2, the framework maintained correct access-control behavior, prevented unauthorized proprietary downloads, and preserved complete audit capture for audit-eligible protected actions. These results show that proprietary document sharing can be governed through identity-aware, tags-driven, and auditable authorization decisions rather than through static permissions or platform-specific controls alone.

The value of the project extends beyond the specific framework that was implemented. Although the motivating context was enterprise-style information governance, the same control model is also relevant to smaller organizations that repeatedly manage sensitive client, case, or regulated records and require traceable, policy-based release over time. In this sense, the project contributes not only a local proof-of-concept, but also a general framework for controlled sharing in environments where accountability, repeatability, and explainable access decisions are essential. At the same time, the implementation approach suggests practical architectural advantages: by combining containerized services with a host-level object store, the model reflects a modular and comparatively portable stack that can be managed locally and adapted for later migration to broader deployment environments. While this does not by itself establish enterprise-scale readiness, it shows that identity management, policy enforcement, metadata persistence, and protected storage can be integrated without relying on a monolithic commercial platform. Such an approach is especially relevant in paperless engineering, manufacturing, and other record-driven workflows where sensitive documents must be shared, revised, and governed over time while reducing exposure associated with fragmented file-handling practices.

FUTURE WORK

Building on the successful localhost evaluation, prospective work should extend the framework into more distributed and reproducible operating settings. Because the present study was conducted on a single laptop with co-located services, the reported results establish local feasibility and provide a strong basis for examining how authentication, policy enforcement, meta-information retrieval, object storage access, and audit logging behave across multiple virtual machines or cloud instances under more realistic deployment conditions. Future development could also strengthen reproducibility through automated environment reset, seeded

dataset generation, scripted workload replay, and formalized figure generation so that the completed evaluation stages can be reproduced consistently. In parallel, the framework can be expanded through richer audit analytics, more expressive attribute-based policy conditions, secure document preview capabilities, viewer-layer controls, and integration-friendly application programming interfaces. Taken together, these directions position the project not as a limited endpoint, but as a validated foundation for a more mature, reproducible, scalable, and extensible metadata-driven secure-sharing framework.

REFERENCES

[1] S. D. Murthy, "The Strategic Architecture of Modern Manufacturing: PLM and ERP as Complementary Digital Systems," in *Journal of Information Systems Engineering and Management*, vol. 11, no. 1s, 2026.

[2] A. Chiquito, U. Bodin, and O. Schelen, *Automated Management of Attribute-Based Policies for Access Control using Tag-Matching*, 2023.

[3] V. Biagi and A. Russo, "Data Model Design to Support Data-Driven IT Governance Implementation," in *Technologies*, vol. 10, no. 106, 2022. Doi: 10.3390/technologies10050106.

[4] S. D., Minavathi, and M. S. D., "Towards Resilient PLM Architectures: Cyber Threats, Security Mechanisms, and Industrial Implications," in *IITM Journal of Management and IT*, vol. 16, no. 2, Dec. 2025.

[5] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," in *National Institute of Standards and Technology (NIST)*, Gaithersburg, MD, Special Publication (SP) 800-207, 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-207.

[6] A. Teymurova, "Metadata Management in Data Governance: A Comprehensive Scientific Analysis," in *Azerbaijan State University of Economics (UNEC)*, Baku, Azerbaijan, 2026. [Online]. Available: teymurova.arzu.rovshan.2023@unec.edu.az.

[7] A. O. Adebayo, "The SAIS-GRC Framework: Engineering Trust and Secure, Agile Systems for Proactive AI Governance and Compliance," in *Iconic Research and Engineering Journals*, vol. 8, no. 5, pp. 1475-1480, Nov. 2024. Doi: 10.64388/IREV7I5-1713349.