

LAADS: Design and Implementation of a Sector-Aware Security Advisory Platform for Academic Institutions

Yerramsetty Chinna Rama Murthy Naidu¹, Siva Rama Krishna T²

¹PG Scholar, Department of Computer Science and Engineering, UCEK, JNTUK, Kakinada, Andhra Pradesh, India.

²Associate Professor, Department of Computer Science and Engineering, UCEK, JNTUK, Kakinada, Andhra Pradesh, India.

How to cite this paper:

Yerramsetty Chinna Rama Murthy Naidu¹, Siva Rama Krishna T², "LAADS: Design and Implementation of a Sector-Aware Security Advisory Platform for Academic Institutions", IJIRE-V7I4-171-179.



Copyright © 2026
by author(s) and
Fifth Dimension
Research

Publication. This work is licensed under the
Creative Commons Attribution International
License (CC BY 4.0).

<http://creativecommons.org/licenses/by/4.0/>

Abstract: Academic institutions increasingly hold data and infrastructure worth targeting, yet most lack a dedicated Security Operations Center capable of turning raw threat intelligence into advisories that are actually distributed to and acted on by their community. Existing Security Information and Event Management stacks handle log aggregation, and threat-intelligence platforms handle indicator storage, but neither produces institution-tailored, human-readable bulletins or delivers them across the channels an institutional audience actually uses. This paper presents the design and implementation of LAADS (LLM-Augmented Advisory Authoring and Dissemination System), a sector-aware security advisory platform built for this gap: a five-layer web system that authors advisories through three entry paths (manual authoring, AI-assisted extraction from a submitted article, and escalation of an external feed item), enforces sector-scoped multi-tenancy across five operational sectors at four independent layers, correlates advisories against an institutional asset inventory through a three-tier matching engine, and disseminates published advisories over six channels spanning real-time push, e-mail, chat webhooks, PDF export and share links. We describe the architecture, data model and enforcement mechanisms in detail and report functional verification of each subsystem. Quantitative evaluation of enrichment performance under production load is left as future work.

Key Words: Security Operations Center; Threat Advisory Management; Sector-Aware Multi-Tenancy; Role-Based Access Control; Multi-Channel Dissemination; Cyber Threat Intelligence; Asset Inventory Matching.

I.INTRODUCTION

The cyber-threat landscape facing academic institutions has shifted markedly over the last several years. Universities increasingly host high-value research data, administer payment and scholarship systems, and operate identity infrastructure whose compromise carries reputational and operational consequences, yet the historical assumption that such institutions attract little attacker interest no longer holds [1].

Operationally, most academic institutions in resource-constrained settings do not run a dedicated Security Operations Center (SOC). Security Information and Event Management (SIEM) platforms handle log aggregation and alerting well [2], and the operational difficulties of running them at scale are themselves well documented [3]. Threat-intelligence platforms such as MISP [4] and OpenCTI [5] provide rich indicator-sharing infrastructure, but neither produces the human-readable, institution-tailored advisories that a security function must ultimately distribute to faculty, students and administrators. Both categories of tooling presuppose a trained team able to curate and author such bulletins from raw data. Large language models have recently been explored as a way to reduce exactly this staffing assumption in security operations contexts [6], and offer a credible way to automate the extraction of a structured advisory from unstructured source text.

This paper presents the design and implementation of LAADS, a platform that fills this specific gap: the authoring, enrichment, and multi-channel dissemination of institutional security advisories. We deliberately scope LAADS to the advisory lifecycle — ingest, enrich, author, distribute, account — and do not address incident-response orchestration or case management, which remain the responsibility of an institution's broader security function. The system was implemented at the Department of Computer Science and Engineering, JNTU Kakinada. Its server side is written in Node.js against a document database; its client is a React single-page application that receives live updates over a WebSocket connection.

This paper makes the following contributions:

- 1) An end-to-end advisory lifecycle spanning three authoring entry paths — manual authoring, AI-assisted extraction from a submitted article, and escalation of an external feed item — and a six-channel dissemination layer with recipient-level engagement telemetry.
- 2) A four-layer sector-aware multi-tenant access model that partitions users, assets and advisories across five operational sectors and enforces that partition independently at the data, API, real-time and notification layers.

A three-tier asset auto-matching engine that correlates incoming advisories against an institutional asset inventory by CVE identifier, vendor name and keyword overlap.

Section II surveys related work. Section III presents the system architecture. Section IV describes the advisory lifecycle and dissemination layer. Section V details the sector-aware access model, and Section VI the asset-matching engine. Section VII reports what has been implemented and functionally verified. Section VIII sets out the threat model and limitations, and Section IX concludes and outlines future work, including the quantitative evaluation this paper does not attempt.

II. RELATED WORK

A. SOC Automation and Threat Intelligence Platforms

Vielberth et al. [7] surveyed SOC functions and identified analyst overload as the dominant operational bottleneck; Islam et al. [8] reviewed security orchestration tooling more broadly, and Schlette et al. [9] compared incident-response platforms in detail. All three focus on response coordination rather than advisory production. MISP [4] and OpenCTI [5] remain the dominant open-source threat-intelligence platforms; both excel at indicator storage and federation but leave advisory authoring to a trained curator, and surveys of intelligence sharing [10] describe the same assumption of a skilled curating team. Our work targets this upstream gap directly.

B. Access Control and Multi-Tenancy

Role-based access control (RBAC) [11], later formalised as a NIST standard [12], and its attribute-based extension (ABAC) [13] establish the standard vocabulary for the access model we build on. Multi-tenant SaaS isolation has been studied extensively in the shared-schema setting [14]. Our five-sector model, with per-entity all/multiple markers, is best understood as a constrained ABAC overlay tuned for the small, hierarchical structure of an academic institution rather than the general enterprise case.

C. Vulnerability Data and Asset Correlation

The National Vulnerability Database (NVD) [15] is the canonical CVE corpus and is treated in LAADS as an authoritative oracle for enrichment, although systematic assessment of its record quality [16] cautions that entries vary in completeness, and automated severity scoring from vulnerability text has been explored as one response [17]. Active network vulnerability scanning [18] addresses the complementary problem of discovering exposed assets; our asset-matching engine instead correlates already-published advisories against a declared inventory, which is a lighter-weight and deployment-friendly alternative for institutions without a scanning program.

D. Automated Extraction of Structured Threat Data

Recovering structured facts from unstructured advisory prose predates the current generation of language models: TTPDrill [19] derived threat actions from narrative reports using an ontology-guided parser. More recent work applies staged language-model pipelines to the same extraction problem [20] and reports improved recovery of technique-level detail. Our authoring path uses a comparable extraction step to populate a structured advisory from a submitted article, but we treat it as one of three entry paths rather than as the object of study, and we make no claim about its extraction quality.

E. Positioning

Table no 1 places this work against the three classes of tooling an institution would otherwise combine. The distinction is not that any of them is deficient at its own task, but that none produces a bulletin tailored to one institution and delivers it to the individuals that institution employs. SIEM operates on log events, threat-intelligence platforms on indicators, and national CERT advisories on a national audience; the advisory tailored to a single campus, and the record of who received it, fall between them.

Dimension	SIEM	TIP	CERT	This work
Unit of distribution	events	IoCs	bulletin	bulletin
Tailored to an institution	no	no	no	yes
Drafting automated from prose	no	no	no	yes
Visibility model	RBAC	groups	public	sector
Per-recipient telemetry	no	no	no	yes
Local asset correlation	partial	partial	no	3-tier
Presumes trained staff	yes	yes	yes	reduced

Table no 1: Positioning against adjacent tooling.

III. METHODOLOGY

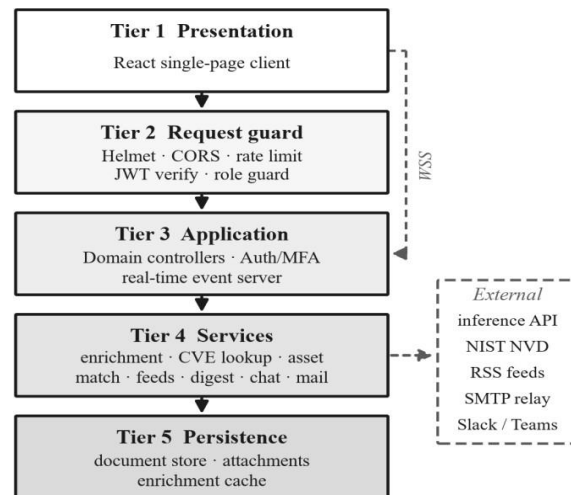
A. Overview

LAADS is organised into five layers. At the top, the browser client renders the operator surfaces. Requests from it pass through a chain of middleware that establishes who the caller is and what they may do, and only then reach the third layer, where the domain controllers, the authentication module and the real-time event server live. Those controllers delegate

longer-running work to a fourth layer of background services covering enrichment, CVE lookup, asset correlation, feed aggregation, digest scheduling, chat delivery and outbound mail. The fifth layer holds all durable state in a document database, together with an attachment store and a disk-backed enrichment cache. External dependencies — the inference provider, NVD, the configured feeds, the mail relay and the chat platforms — sit outside this boundary and are reached only from the service layer. Figure 1 renders the arrangement.

B. Backend and Frontend Stack

The backend uses Node.js with Express and Mongoose for MongoDB object–document mapping. Authentication follows a two-token strategy: a short-lived access token is held in browser memory only, never in persistent storage, while a refresh token travels in an HTTP-only, SameSite=Strict cookie, which together resist XSS-based token exfiltration and cross-site request forgery without a separate anti-CSRF token. Optional TOTP-based multi-factor authentication is available for administrator accounts. The client is built on React 19 with Vite and Tailwind CSS; React Query manages server state and invalidates its cache directly from incoming Socket.IO events, removing the need for client-side polling.



Requests descend; only Tier 4 crosses the boundary.

Figure 1: Five-layer architecture. Synchronous requests descend from the client through the request guard into the application layer; only the service layer crosses the trust boundary to external providers. The real-time channel connects the client directly to the event server.

C. Data Model

Twelve Mongoose collections anchor LAADS. Five core collections — Advisory, User, Asset, Comment and Notification— carry the primary domain data. Seven supporting collections carry audit and engagement telemetry (AuditLog, AdvisoryLog), the review surface (Acknowledgement, KnowledgeArticle), submitted-article and configuration state (ManualNews, PlatformSettings), and atomic identifier issuance (a single-document Counter). External feed items are intentionally not persisted; they live only in an in-process cache and are converted into Advisory documents on demand when an analyst acts on them. Table no 2 summarises the collections and the role each plays.

Collection	Role
Advisory	The bulletin; unit of distribution
User	Account, role and sector membership
Asset	Inventory entry with monitored CVEs
Comment	Threaded per-advisory discussion
Notification	Per-user in-app events
AuditLog	Authentication and mutation trail
AdvisoryLog	Share, export and acknowledgement telemetry
Collection	Role
Acknowledgement	Team-wide dismissal of an item
KnowledgeArticle	Runbooks and reference material
ManualNews	Analyst-submitted source articles
PlatformSettings	Webhook targets and severity thresholds
Counter	Atomic SOC-YYYY-NNNN issuance

Table no 2: Persistence model.

IV. ADVISORY LIFECYCLE AND DISSEMINATION

The five stages named in Section I — ingest, enrich, author, distribute and account — are the scope this paper claims, and Figure 2 shows how an item moves through them. The remainder of this section takes each stage in turn.

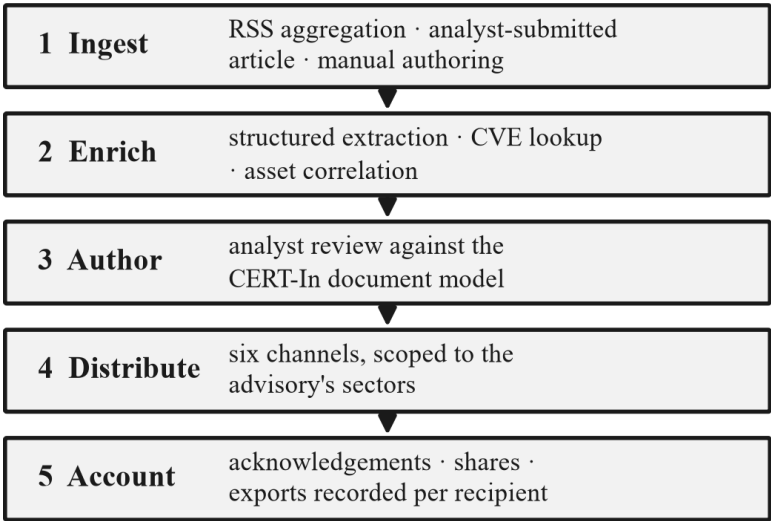


Figure 2: The advisory lifecycle. An item enters by one of three ingestion routes, is enriched and reviewed against a fixed document model, then fans out across the dissemination channels, with every resulting action recorded.

A. Advisory Document Model

Every advisory is a single document combining a CERT-In-compatible textual body — title, overview, target audience, risk and impact assessments, description, affected-systems list, solution and references — with metadata that routes, filters and audits it: severity, a sectors array, CVE identifiers, a CVSS score, indicators of compromise, and creation/modification fields. Each advisory receives an atomic, monotonically increasing SOC-YYYY-NNNN identifier issued through the Counter collection under an atomic increment, which guarantees collision-free issuance under concurrent inserts.

B. Authoring Entry Paths

Three workflows produce an advisory. In manual authoring, an analyst completes a templated form with field-level validation. In AI-assisted authoring, an analyst supplies only an article title and URL; LAADS fetches and sanitises the page text and invokes a large-language-model enrichment step to populate a structured draft against the same schema, which the analyst reviews before it is treated as published. In feed escalation, an external feed item — never itself persisted — is converted into a read-only advisory carrying an auto status when an analyst chooses to act on it. The LLM enrichment step draws on a two-tier model cascade with an automatic fallback to a smaller model, which exists to preserve availability rather than to optimise output quality; we describe its implementation but do not evaluate its performance in this paper.

C. Six-Channel Dissemination and Telemetry

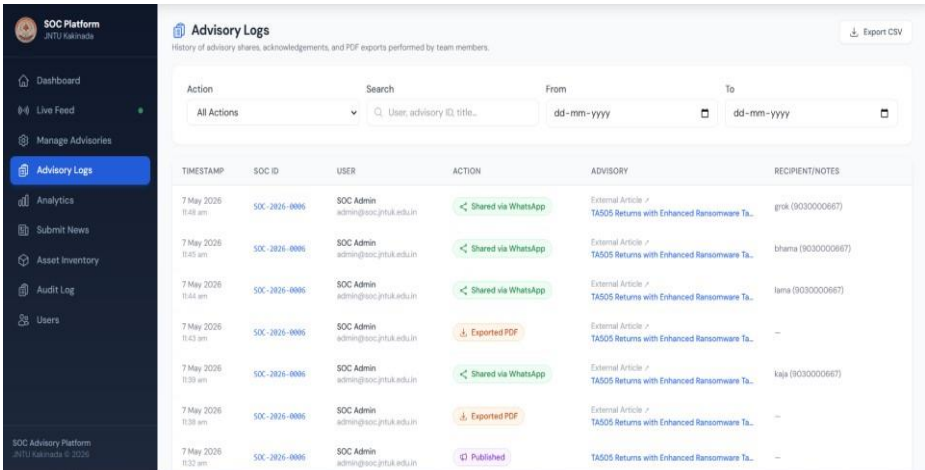


Figure 3: Advisory Logs view. Every share, acknowledgement and export is recorded with the performing user, optional recipient and a timestamp, filterable by action type, advisory, user and date range.

On publication, an advisory fans out over six independent, non-blocking channels: real-time WebSocket push to sector-scoped rooms, sector-matched e-mail with retry, severity-thresholded Slack and Microsoft Teams webhooks, a client-side CERT-In-styled PDF export, and pre-filled WhatsApp and Gmail share links — the last of these reflecting the observation that institutional stakeholders in our deployment context communicate operationally over channels beyond enterprise e-mail. A complementary cron-scheduled digest e-mails a severity-filtered daily and weekly summary to users who do not maintain an active session. Actions taken around dissemination are themselves durable: each share, each acknowledgement and each export is appended to a tamper-resistant log, which yields telemetry at the level of the individual recipient rather than the individual indicator. Figure 3 shows the resulting per-advisory trail as surfaced to administrators.

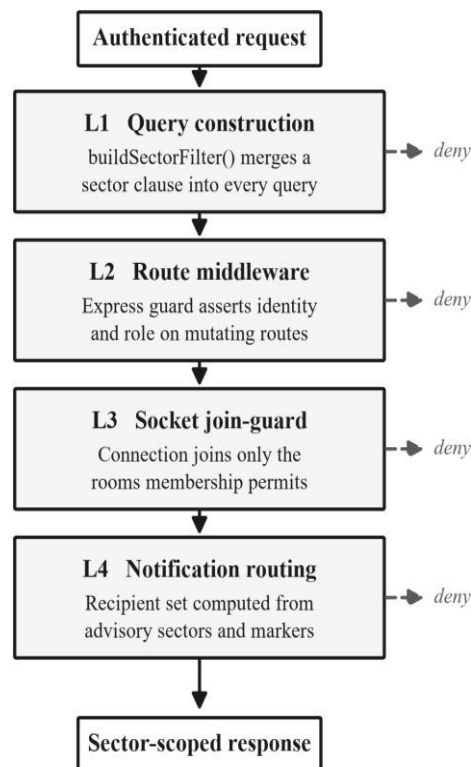
V. SECTOR-AWARE MULTI-TENANCY

A. Sector Vocabulary and Roles

LAADS partitions users, assets and advisories across five operational sectors. Two cross-cutting markers extend this vocabulary: users and assets may carry an all marker granting cross-sector presence, while advisories carry a multiple marker for items relevant to more than one sector. A conventional three-role model (admin, analyst, viewer) overlays the sector assignment on each user account. Table no 3 states the resulting visibility rule exhaustively; it is the whole of the read model, and every enforcement layer in Section V-B implements exactly this.

Viewer membership	Advisory in viewer's sector	Advisory in another sector	Advisory marked multiple
One sector	visible	hidden	visible
Several sectors	visible	hidden	visible
Marked all	visible	visible	visible

Table no 3: Sector visibility rule.



Each layer enforces the partition independently.

Figure 4: Four-layer sector enforcement. A request is filtered independently at query construction, route middleware, socket room join and notification routing; a denial at any layer yields an empty result rather than an explicit refusal.

B. Enforcement Layers

Sector isolation is enforced independently at four points rather than in a single choke point, as summarised in Figure 4. A reusable query-construction helper merges a sector filter into every advisory- and feed-returning database query. Express middleware verifies the requester's identity and role on every mutating route. The Socket.IO server joins each authenticated connection only to the rooms its sector membership permits, and rejects any runtime request to join a room outside that set. Finally, the notification-routing utility computes each event's recipient set as the union of the advisory's sectors and the multiple marker, always including all-marked users. Because the same authoritative filter logic underlies both read and write paths, the isolation guarantee does not depend on any single layer holding correctly.

VI. ASSET AUTO-MATCHING

A. Inventory Model

Each tracked asset carries a name, vendor, version, category, sector membership, a keyword list, a set of monitored CVE identifiers, and an optional link to a knowledge-base runbook. Figure 5 shows the corresponding capture workflow.

Figure 5: Asset capture workflow, recording vendor, category, sector membership, monitored CVE identifiers and a linked knowledge-base playbook.

B. Three-Tier Matching Algorithm

When an advisory is published, the matching engine applies a three-tier precedence rule in place of a weighted score, on the reasoning that CVE identity, vendor name and keyword overlap carry strictly decreasing evidentiary strength and any one signal suffices to flag an asset. The engine first matches on shared CVE identifiers, the strongest signal; assets not matched at that tier are tested against the advisory's vendor field; assets still unmatched are tested for keyword overlap between the advisory text and the asset's name and keyword list. The first tier that flags an asset wins, so no asset appears twice in the result. Flagged assets surface in a dedicated review view, converting a public feed into an institution-specific triage queue. Algorithm 1 states the rule directly, and Figure 6 shows the precedence order and the path taken by an asset that matches at no tier.

Algorithm 1: Three-tier asset auto-matching.

Input: published advisory a , with CVE set C_a , vendor field v_a and text T_a ;
institutional asset inventory I

Output: flagged set F , each asset tagged with the tier that matched it

```

1:  $F \leftarrow \emptyset$ 
2: for each asset  $s \in I$  do
3:   if  $C_a \cap s.\text{monitoredCves} \neq \emptyset$  then      D Tier 1: CVE identity
4:      $F \leftarrow F \cup \{(s, 1)\}$ 
5:   else if  $v_a$  matches  $s.\text{vendor}$  then      D Tier 2: vendor name
6:      $F \leftarrow F \cup \{(s, 2)\}$ 
7:   else if  $\text{tokens}(T_a) \cap (s.\text{name} \cup s.\text{keywords}) \neq \emptyset$  then  D Tier 3: keyword overlap
8:      $F \leftarrow F \cup \{(s, 3)\}$ 
9:   end if      D the chain is exclusive: the first tier to match wins
10: end for
11: return  $F$ 

```

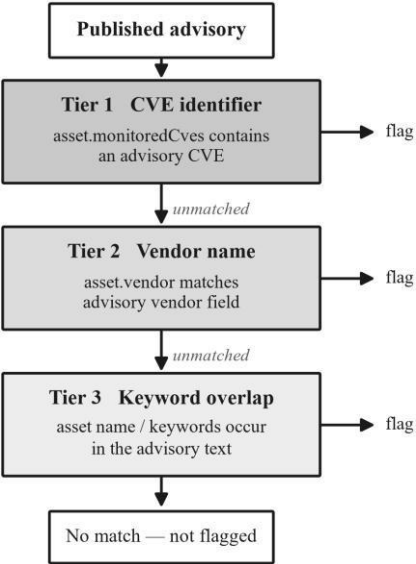


Figure 6: Three-tier asset matching precedence. Evaluation stops at the first tier that flags an asset, so no asset is reported twice.

VII. IMPLEMENTATION AND FUNCTIONAL VERIFICATION

A. What Is Built

LAADS as described above is implemented in full and deployed as a working system rather than a partial prototype: twelve Mongoose collections, seven backend services, four independent sector-enforcement layers, three advisory-authoring paths, and six dissemination channels are all present and operating together in the released codebase, alongside the two-tier LLM enrichment cascade and the three-tier asset matcher described in Sections IV–VI. Administrative actions across the platform are recorded in a queryable audit trail (Figure 7).

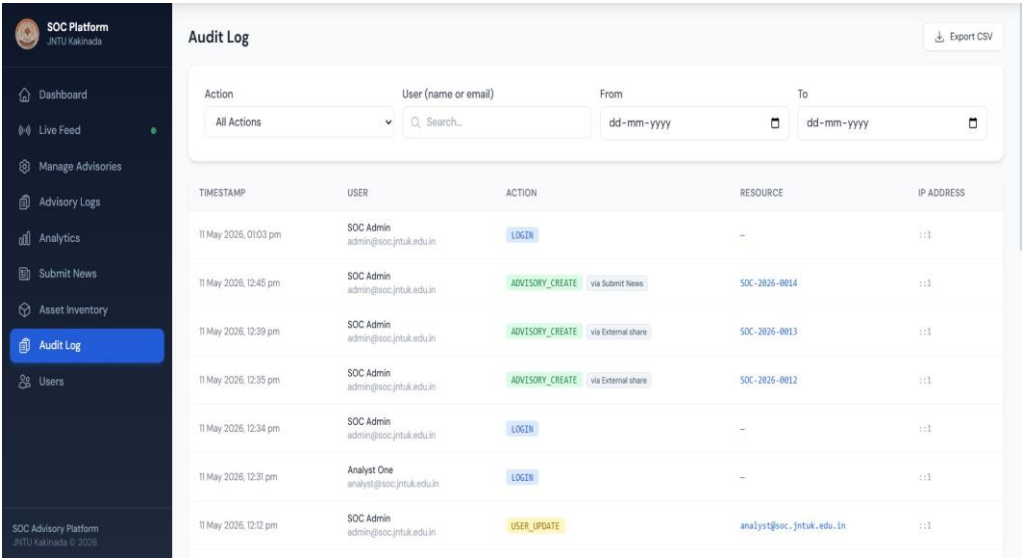


Figure 7: Audit Log view, recording each authentication, advisory-creation and user-management event with its subject, resource and source address.

B. Functional Verification

We verified each mechanism functionally rather than quantitatively, exercising the paths described above directly and confirming the behaviour each is designed to produce. Table no 4 summarises the mechanisms checked and the outcome observed for each. What this establishes is narrow but definite: on the inputs we tried, each path did what it was built to do. It says nothing about how much of the input space those cases cover, nor about behaviour once the system is placed under sustained load. A quantitative evaluation of the enrichment cascade — latency, token consumption and behaviour under quota exhaustion — is outside the scope of this paper and is identified as future work in Section IX.

Mechanism	Verified behaviour
Manual authoring	Advisory created end-to-end with atomic SOC-YYYY-NNNN identifier
AI-assisted authoring	Structured draft correctly returned from a submitted article URL

Feed escalation	Advisory created with auto status and enforced read-only
Sector scoping	Cross-sector reads refused at all four enforcement layers
Cascade fallback	Primary-to-fallback tier transition returns a parseable result
Six-channel dissemination	All six channels deliver; actions logged in AdvisoryLog
Asset auto-matching	Tracked assets correctly flagged via CVE, vendor and keyword precedence

Table no 4: Functional verification summary.

VIII. THREAT MODEL AND LIMITATIONS

A. What the Sector Model Defends Against

The sector partition is designed to stop a legitimate, authenticated user from casually reaching material outside their remit— an analyst in one sector browsing another sector's advisories, or a notification reaching a recipient who should not have been told. Enforcing it at four independent points means a defect in any single layer does not by itself expose data, because the query filter, the route guard, the room join and the notification router each derive the same restriction from the same membership record.

It is equally important to state what the model does not do. It is not a substitute for database-level isolation. All sectors share one schema and one deployment, so an attacker who obtains database credentials, compromises an administrator account, or achieves code execution inside the application process is outside the model entirely. Where cross-sector confidentiality is a hard regulatory requirement rather than an operational preference, separate instances or row-level security in the database should be used in addition. The AI-assisted authoring path additionally trusts the content of a page an analyst submits: a hostile source could attempt to steer extraction, which is why the resulting draft is presented for review rather than published unread.

B. Limitations

Four limits bound what this paper establishes. First, verification is functional: the mechanisms were exercised and behaved as designed, but we report no measurement of how well they perform, and Section IX names that as the next step. Second, the matcher's precedence rule deliberately favours recall over precision, on the reasoning that a reviewer dismissing a false positive is cheaper than a missed advisory; that trade is a design choice and has not been quantified. Third, the deployment described here is a single node with one institution's data, so nothing is claimed about behaviour under concurrent load or at multi-institution scale. Fourth, the five-sector vocabulary is fixed and flat; an institution whose structure is hierarchical, or which needs sectors to nest, would require the model to be generalised.

IX. CONCLUSION AND FUTURE WORK

We presented the design and implementation of LAADS, a sector-aware security advisory platform that unifies advisory authoring, LLM-assisted enrichment, institutional access control, asset correlation and multi-channel dissemination in a single system built for resource-constrained academic institutions. Each mechanism was verified functionally against the behaviour it is designed to produce. Three directions remain for future work. First, and most significantly, a quantitative evaluation of the enrichment cascade — latency and token consumption by model tier, behaviour under real quota exhaustion, and extraction reliability — was deliberately left outside the scope of this paper and is the immediate next step. Second, the asset-matching engine has been verified functionally but not evaluated against a larger, labelled inventory. Third, the sector model could be extended to support hierarchical or nested sectors for institutions with a more complex organisational structure than the five-sector vocabulary presently assumes.

This paper is confined to the design of LAADS and to functional verification of the mechanisms it comprises. A companion manuscript, under review separately, reports the measured evaluation of the enrichment cascade — per-tier latency, token consumption and behaviour under quota exhaustion — and makes no design contribution beyond what is described here. The two are distinct contributions concerning one system.

Acknowledgment

The authors thank the Department of Computer Science and Engineering, JNTU Kakinada, for supporting this work.

References

1. A. Ulven and G. Wangen, "A systematic review of cybersecurity risks in higher education," *Future Internet*, vol. 13, no. 2, art. 39, 2021.
2. A. González-Granadillo, S. González-Zarzosa, and R. Diaz, "Security Information and Event Management (SIEM): Analysis, trends, and usage in critical infrastructures," *Sensors*, vol. 21, no. 14, art. 4759, 2021.
3. M. Cinque, D. Cotroneo, and A. Pecchia, "Challenges and directions in Security Information and Event Management (SIEM)," in *Proc. IEEE Int. Symp. Software Reliability Engineering Workshops (ISSREW)*, 2018, pp. 95–99.
4. C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "MISP: The design and implementation of a collaborative threat intelligence sharing platform," in *Proc. ACM Workshop on Information Sharing and Collaborative Security (WISCS)*, 2016, pp. 49–56.
5. OpenCTI Platform, "Open Cyber Threat Intelligence Platform: Architecture and design," *Filigran, Tech. Rep.*, 2022.
6. A. Habibzadeh, F. Feyzi, and R. E. Atani, "Large language models for security operations centers: A comprehensive survey," *Journal of Electrical and Computer Engineering*, 2026.
7. M. Vielberth, F. Böhm, I. Fichtinger, and G. Pernul, "Security Operations Center: A systematic study and open challenges," *IEEE Access*, vol. 8, pp. 227756–227779, 2020.
8. C. Islam, M. A. Babar, and S. Nepal, "A multi-vocal review of security orchestration," *ACM Computing Surveys*, vol. 52, no. 2, art. 37, 2020.

- 2019.
9. D. Schlette, M. Caselli, and G. Pernul, "A comparative study on cyber threat intelligence: The security incident response perspective," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2525–2556, 2021.
10. T. D. Wagner, K. Mahbub, E. Palomar, and A. E. Abdallah, "Cyber threat intelligence sharing: Survey and research directions," *Computers & Security*, vol. 87, art. 101589, 2019.
11. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.
12. D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Transactions on Information and System Security*, vol. 4, no. 3, pp. 224–274, 2001.
13. V. C. Hu, D. R. Kuhn, and D. F. Ferraiolo, "Attribute-based access control," *IEEE Computer*, vol. 48, no. 2, pp. 85–88, 2015.
14. R. Krebs, C. Momm, and S. Kounev, "Architectural concerns in multi-tenant SaaS applications," in *Proc. Int. Conf. Cloud Computing and Services Science (CLOSER)*, 2012, pp. 426–431.
15. H. Booth, D. Rike, and G. A. Witte, "The National Vulnerability Database (NVD): Overview," *NIST, Tech. Rep. ITL Bulletin*, 2013.
16. A. Anwar, A. Abusnaina, S. Chen, F. Li, and D. Mohaisen, "Cleaning the NVD: Comprehensive quality assessment, improvements, and analyses," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 4255–4269, 2022.
17. G. Spanos, L. Angelis, and D. Toloudis, "Assessment of vulnerability severity using text mining," in *Proc. Pan-Hellenic Conf. on Informatics (PCI)*, 2017, art. 49.
18. A. Tundis, W. Mazurczyk, and M. Mühlhäuser, "A review of network vulnerabilities scanning tools," in *Proc. Int. Conf. on Availability, Reliability and Security (ARES)*, 2018, art. 65.
19. G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "TTPDrill: Automatic and accurate extraction of threat actions from unstructured text of CTI sources," in *Proc. Annual Computer Security Applications Conference (ACSAC)*, 2017, pp. 103–115.
20. H. R. Kim, D. Lee, I. Lee, S. Lee, and S. Lee, "Multi-step LLM pipeline for enhancing TTP extraction in cyber threat intelligence," *IEEE Access*, vol. 13, pp. 179696–179710, 2025.