JOURNAL OF DECISION MAKING AND HEALTHCARE
Volume 3 (2026), No. 2, 123–133
DOI:https://doi.org/10.69829/jdmh-026-0302-ta04
Tulipa Opera Scholarum

SECURITY LOG ANALYSIS OF A HEALTHCARE WEB APPLICATION USING LLM CLASSIFICATION AND UNSUPERVISED CLUSTERING

CHAWALIT CHANINTONSONGKHLA[1,*], PRASIT WONGSUPA[2], AND MOHAMMAD FARID[3]

[1] *School of Dentistry, University of Phayao, Phayao, Thailand*
[2] *Phayao Provincial Public Health Office, Phayao, Thailand*
[3] *Department of Mathematics, College of Science, Qassim University, Saudi Arabia*

Abstract. Healthcare web applications require periodic security auditing, yet many institutions lack dedicated security staff to review server logs. This study applied artificial intelligence (AI)-assisted log auditing to a healthcare web application at a Thai university. A total of 158,720 structured log entries collected over 163 days across three logging schema iterations were analyzed using two complementary methods. For threat classification, a large language model (LLM; GPT-5 Nano) classified batches of log entries along two dimensions (intent and impact) using structured JSON output. For pattern visualization, sentence embeddings of the log entries were projected to two dimensions using Uniform Manifold Approximation and Projection and clustered with Hierarchical Density-Based Spatial Clustering of Applications with Noise. The analysis identified 26 semantic clusters spanning six categories: attack traffic (10.4% of entries), security events (1.9%), probing (0.65%), routine operations (23.9%), system maintenance (3.3%), and mixed (0.9%), with the remaining 59.0% classified as noise. A single-day automated scanning flood accounted for the majority of attack-related entries. The results suggest that periodic large language model-assisted auditing can monitor web access patterns in production logs without requiring predefined signatures, and that embedding-based clustering provides a visual overview of traffic composition and security posture.

Keywords. Healthcare security, AI-assisted auditing, log analysis, large language models, text embeddings, unsupervised clustering.

1. Introduction

Healthcare institutions, including dental practices, increasingly deploy web applications with public-facing interfaces to support patient self-enrollment, appointment scheduling, and service access [1]. Moving these services from internal-only terminals to a public domain enables patients to interact with the hospital from personal devices, reducing barriers to care access. However, making a healthcare application reachable from the internet simultaneously exposes it to automated scanning. Internet-wide scanning tools can survey the entire IPv4 address space in minutes [2], meaning newly deployed servers are discovered and probed within hours. Healthcare is an attractive target for cybercrime because it holds valuable data and its defenses are often weak [3]. Security monitoring guidelines such as National Institute of Standards and Technology (NIST) SP 800-92 [4] recommend regular log review as a core security practice, yet many healthcare institutions, particularly smaller hospitals and university clinics, lack dedicated security operations staff to perform this review. Server logs accumulate continuously, and without periodic auditing, security-relevant events go undetected until a breach occurs.

*Corresponding author.
E-mail address: chawalit.ch@up.ac.th (C. Chanintonsongkhla), hippies_129@hotmail.com (P. Wongsupa), and m.jameel@qu.edu.sa (M. Farid)
Received: May 27, 2026, Revised: June 20, 2026, and Accepted: June 28, 2026.

Most research on web application security analysis uses synthetic or benchmark datasets. The HTTP Dataset CSIC 2010 [5], containing approximately 61,000 generated requests, remains the primary benchmark for web attack detection. The CICIDS2017 dataset [6] provides network-level features from a controlled testbed. These datasets have enabled methodological advances, but they were created in laboratory environments and do not capture the mix of legitimate traffic, automated integrations, and varied attacker sophistication found in production systems.

Large language models offer new approaches to security log analysis. LLMs can process batches of log entries within their context window, classify threats using structured output formats, and handle the heterogeneous log schemas common in production systems. Unlike signature-based systems that require predefined rules, LLMs can identify anomalous patterns based on the semantic content of request data. Institutions that cannot maintain continuously updated rule sets stand to benefit most from this approach.

This study applied AI-assisted log auditing to a healthcare web application deployed at a Thai university. The system logged application events and HTTP access requests in JSON Lines format over 163 days. Two complementary methods were applied: LLM-based batch classification assigned threat levels to log entries using a two-dimensional matrix (intent and impact), and sentence embedding with unsupervised clustering provided visual exploration of the classified entries. The study addresses three questions: (1) What types of security-relevant events appear in healthcare web application logs, and at what prevalence? (2) How does embedding-based clustering complement LLM batch classification for visualizing the threat landscape? (3) What practical considerations arise when deploying AI-assisted log auditing in data-sensitive healthcare environments?

2. Related Work

2.1. **Web application log analysis.** Web application security analysis has traditionally relied on rule-based intrusion detection systems that match known attack signatures against incoming HTTP requests. The HTTP Dataset CSIC 2010 [5] remains the most widely used benchmark, while CICIDS2017 [6] provides a larger-scale benchmark with network-level features. Both datasets were generated in controlled environments and may not reflect the traffic mix found in production systems. Landauer et al. [7] surveyed system log clustering for anomaly detection and found that while most studies used real-world system logs, nearly 60% involved non-reproducible evaluations using private datasets, and few included confirmed attack scenarios.

2.2. **LLM applications in security.** Traditional unsupervised anomaly detection methods such as Isolation Forest [8] identify outliers by recursively partitioning feature space, achieving linear time complexity with efficient processing of high-dimensional data. However, these methods require numerical feature engineering and cannot directly process the semantic content of log messages. Recent work has explored using large language models for security tasks including vulnerability detection in source code, malware classification, and log summarization. Guan et al. [9] showed that LLMs can detect anomalies in system logs by processing the semantic content of log messages, achieving higher detection accuracy than traditional deep learning approaches after fine-tuning on labeled log data. Modern LLMs also support structured JSON output schemas, enabling systematic classification with machine-parseable results rather than free-text responses. The trade-off is computational cost per entry and the question of whether sensitive log data can be sent to cloud-based API providers, a consideration that is particularly acute in healthcare settings subject to data protection regulations.

2.3. **Healthcare IT security.** Healthcare institutions face security challenges compounded by regulatory requirements, legacy system dependencies, and the operational priority of system availability over security hardening. Coventry and Branley [3] identified healthcare as an attractive target for cybercrime for two reasons: it is a rich source of valuable data and its defenses are historically

weak. Increased connectivity, including public-facing patient portals and integration with messaging platforms, has expanded the attack surface beyond traditional network perimeters. The Thailand Personal Data Protection Act (PDPA) and comparable frameworks such as the Health Insurance Portability and Accountability Act (HIPAA) impose data handling requirements that affect how logs can be collected, stored, and analyzed, particularly when using cloud-based processing services. Studies analyzing real healthcare application logs remain scarce, with most healthcare cybersecurity research addressing network-level intrusion detection or policy frameworks rather than application-layer log analysis.

The intersection of healthcare log analysis, LLM-based classification, and embedding-based visualization has not been studied. This work addresses that gap by applying two complementary AI-assisted methods to 163 days of logs from a university healthcare web application.

3. Methods

3.1. **Application architecture.** The study system is a Django-based dental hospital information system deployed on a university virtual private server under a public domain name (dhos.up.ac.th). Prior to the web deployment, patients interacted with the hospital system only through staff-operated terminals on the local area network. The public deployment provides a memorable domain name rather than requiring access to internal network addresses (e.g., 10.99.x.x), enabling patient self-enrollment from personal devices without visiting the hospital in person. A Caddy reverse proxy handles HTTPS termination and routes requests to the application, which exposes two interfaces with different access policies (Figure 1).

The public interface serves a patient self-enrollment portal accessible via the internet. Patients authenticate through mobile one-time password (OTP) with SMS verification or institutional single sign-on. The portal implements automated bot verification (Cloudflare Turnstile), per-IP rate limiting on authentication endpoints, and phone-number lockout after repeated failed attempts. Through this interface, patients can register, view appointment schedules, and generate service QR codes.

The internal interface provides hospital staff with appointment scheduling, patient record management, and departmental queue coordination. Access is restricted to the local area network through IP-based middleware. Staff authenticate through Django session authentication, Microsoft Entra ID (OAuth 2.0) for institutional single sign-on, or optional passkey (WebAuthn) support. This interface is not directly reachable from the internet.

3.2. **Data collection.** Log data was collected from the production deployment from September 23, 2025 to March 4, 2026 (UTC+7). Two log sources were analyzed: the application event log (app.jsonl, 149,088 entries) and the HTTP access log (access.jsonl, 9,632 entries). Table 1 summarizes the dataset composition across the three structural periods.

An internal memorandum documenting the scope and data handling procedures was submitted to the hospital administration prior to analysis. The study analyzed only system-generated server logs. No patient data, personal health information, or identifiable personal data was collected, and no human subjects ethics approval was required [10].

3.3. **Logging structure and data anonymization.** The system generates structured logs in JSON Lines (JSONL) format through two sources: an application event log recording framework events, authentication actions, and errors; and an HTTP access log recording processed requests with response metadata.

The application event log contains timestamp (ISO 8601), log level, source module, message text, client IP address, and authenticated user identifier. The HTTP access log adds request method, URL path, HTTP status code, username, and user type classification.
The dataset spans three structural periods:

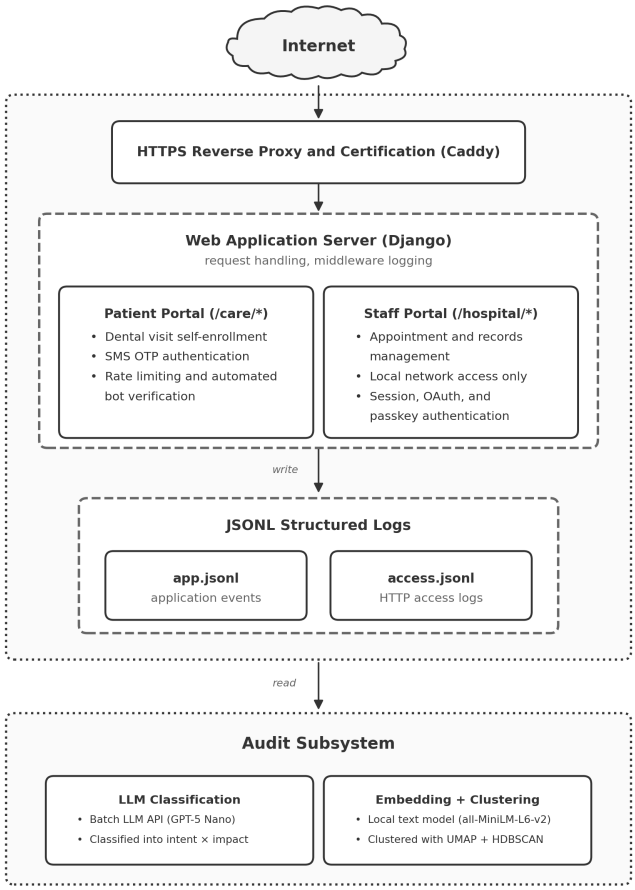

FIGURE 1. System architecture of the healthcare web application. The patient portal (left) receives internet traffic through an HTTPS reverse proxy with automated bot verification, rate limiting, and OTP authentication. The staff portal (right) is restricted to the local network. Both interfaces connect to the web application server, which generates JSONL-formatted structured logs. The audit subsystem (dotted border) operates as an external process that periodically reads these logs for analysis

TABLE 1. Dataset composition: log sources, entry counts, and characteristics of the three structural periods ($n = 158,720$)

Period	Days	Sources	Entry Count	Schema Notes
P1	18	app	4,907	Initial deployment, app events only, no IP capture
P2	74	app+access	42,117	Separate access logging enabled, stable schema
P3	71	app+access	111,696	Restructured routing, improved IP capture
Total	**163**	**app+access**	**158,720**	

Period 1 (September 23 to October 10, 2025): The initial deployment phase. Only the application event log was active. Authentication details were embedded inline within messages. IP addresses were not recorded from external clients.
Period 2 (October 11 to December 23, 2025): The HTTP access log was introduced as a separate access logging source with a stable schema. External IP addresses began to appear in a subset of entries.

Period 3 (December 24, 2025 to March 4, 2026): A code redeployment restructured URL routing and moved HTTP request details for authentication events into the application log. IP address capture improved, and log volume increased relative to earlier periods.

Prior to analysis, sensitive content was removed or anonymized. Period 1 logs containing mobile OTP codes and phone numbers were excluded. IP addresses were retained for internal analysis but are reported only as aggregate categories (private subnet, external, localhost) in this publication. Authentication tokens and session identifiers were stripped from all entries.

3.4. **LLM batch classification.** A standalone audit subsystem periodically retrieves these logs and classifies entries in batches using LLM API calls with structured JSON response schemas. Each batch receives a two-dimensional threat classification along intent and impact axes (each scored as none, low, medium, or high), together with a list of flagged anomalies and a statistical summary.

For each audit run, entries were grouped by source (application event or HTTP access) and processed in batches within the LLM's context window. Each batch was sent to the LLM API with a structured JSON output schema requesting: a list of anomalous entries with descriptions, a two-dimensional threat classification (intent: none/low/medium/high, impact: none/low/medium/high), and a statistical summary of the batch.

The LLM was provided with system context describing the application's architecture, expected traffic patterns (patient portal, LINE webhooks, kiosk API, staff management), and the private hospital network environment. This context enabled the model to calibrate threat classifications against the specific deployment, distinguishing expected automated traffic (health checks, monitoring probes) from genuinely anomalous requests.

The classification used GPT-5 Nano (gpt-5-nano-2025-08-07) with a maximum batch size of 5,000 log entries per API call. A total of 38 batches were processed: 36 for the application event log and 2 for the HTTP access log. Processing consumed approximately 5.9 million prompt tokens and 253,000 completion tokens.

3.5. **Sentence embedding and clustering.** Each log entry was also encoded into a dense vector representation using a sentence embedding model. The embedding input was constructed by concatenating the log level, source module, and message text as "level module message". Timestamps and IP addresses were excluded to focus the representation on semantic content.

The all-MiniLM-L6-v2 model, a distilled variant based on the MiniLM architecture [11] and distributed through the Sentence-Transformers framework [12], was used, producing 384-dimensional embedding vectors. This model was selected for its small size (approximately 80 MB), local deployability, and established performance on semantic textual similarity benchmarks. Running the model locally avoids transmitting log data to external API providers, a consideration under the Thailand Personal Data Protection Act (PDPA) and similar regulations governing server log data in healthcare environments.

The 384-dimensional embedding vectors were projected to two dimensions using Uniform Manifold Approximation and Projection (UMAP) [13] with a neighborhood size of 15, minimum distance of 0.1, and cosine distance. Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [14] was then applied to the two-dimensional projections with a minimum cluster size of 1,000 entries and minimum sample count of 10 to identify clusters of semantically similar entries without specifying a cluster count in advance. The minimum cluster size was set to 1,000 to produce an interpretable number of clusters from the 158,720 entries. Smaller values produced hundreds of micro-clusters that were difficult to annotate meaningfully. Clustering quality was assessed using the silhouette score computed on the clustered entries.

Each cluster identified by HDBSCAN was annotated by examining representative entries and classifying them based on the dominant request patterns. Clusters were assigned descriptive labels (e.g.,

"Automated scanning flood", "External IP access blocks", "Staff portal page views"). The LLM's per-batch anomaly flags served as an independent signal: clusters with high proportions of LLM-flagged entries were examined first as likely threat categories.

3.6. **Computational environment.** All analysis was conducted using Python 3 with Sentence - Transformers (v 5.3.0), UMAP-Learn (v 0.5.11), HDBSCAN (v 0.8.41), and Scikit-learn (v 1.8.0). UMAP projections used a fixed random seed for reproducibility. Sentence embeddings were generated in batches of 512 on a 20-core workstation.

4. Results

4.1. **LLM classification overview.** The LLM classified 38 batches covering 158,720 log entries across both log sources. Of the 38 batches, 18 (47%) were flagged as requiring attention based on the two-dimensional threat classification, while only 2 batches (5%) were classified as containing harmful actions. The intent axis showed the following distribution at batch level: none (14 batches), low (15), medium (1), and high (8). The impact axis showed: none (23), low (13), medium (2), and high (0). No batch received a high impact classification. Because no labeled ground truth existed for this dataset, these classifications reflect the LLM's assessment rather than confirmed attack labels.

The LLM identified 96 individual anomalies across all batches, spanning 83 distinct anomaly types. The most frequently identified anomaly types were external bot probing (4 occurrences across batches), HTTP host header manipulation (2), and GraphQL endpoint probing (2). WordPress-related probing appeared in 5 batches under different labels (login probes, admin panel discovery, path enumeration). Higher-severity anomalies included Java Naming and Directory Interface (JNDI)/Log4Shell injection attempts (intent: high), OTP rate limit abuse from an external IP (intent: high, impact: medium), SQL injection probes targeting authentication endpoints (intent: high), and an external IP authenticating to the patient portal and browsing internal staff pages (intent: medium, impact: medium).

The access log yielded the highest combined threat classification in the dataset (high intent, medium impact), while most application log batches received low or no threat classifications.

4.2. **Embedding cluster analysis.** Uniform Manifold Approximation and Projection (UMAP) and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) applied to the 158,720 log entry embeddings produced 26 clusters containing 65,083 entries (41.0%), with the remaining 93,637 entries (59.0%) classified as noise. The silhouette score computed on the 65,083 clustered entries was 0.428. Cluster sizes ranged from 1,012 to 14,165 entries. The largest cluster (14,165 entries, 8.9%) corresponded to the automated scanning flood, dominated by 404 Not Found responses to probed paths.

The clusters were grouped into six categories based on manual inspection of representative entries: attack (2 clusters, 16,546 entries), security (2 clusters, 2,967 entries), probing (1 cluster, 1,025 entries), operational (17 clusters, 37,968 entries), system maintenance (3 clusters, 5,185 entries), and mixed (1 cluster, 1,392 entries containing both external probes and internal rendering errors). Figure 2 shows the two-dimensional UMAP projection with clusters colored by category.

4.3. **Attack categories.** The two attack clusters together accounted for 16,546 entries (10.4% of total traffic). The larger cluster (14,165 entries) consisted of automated vulnerability scanning producing 404 Not Found responses to paths such as /login, /.env, /index.html, /api/, /wp-admin/, and /test.php. The smaller cluster (2,381 entries) contained more targeted external probes including requests for /.env files, WordPress login pages, and JavaScript directories, predominantly from external IP addresses.

Two security clusters contained the application's defensive responses: unauthorized access attempts (1,637 entries) from internal users attempting to reach restricted portal sections, and external IP access

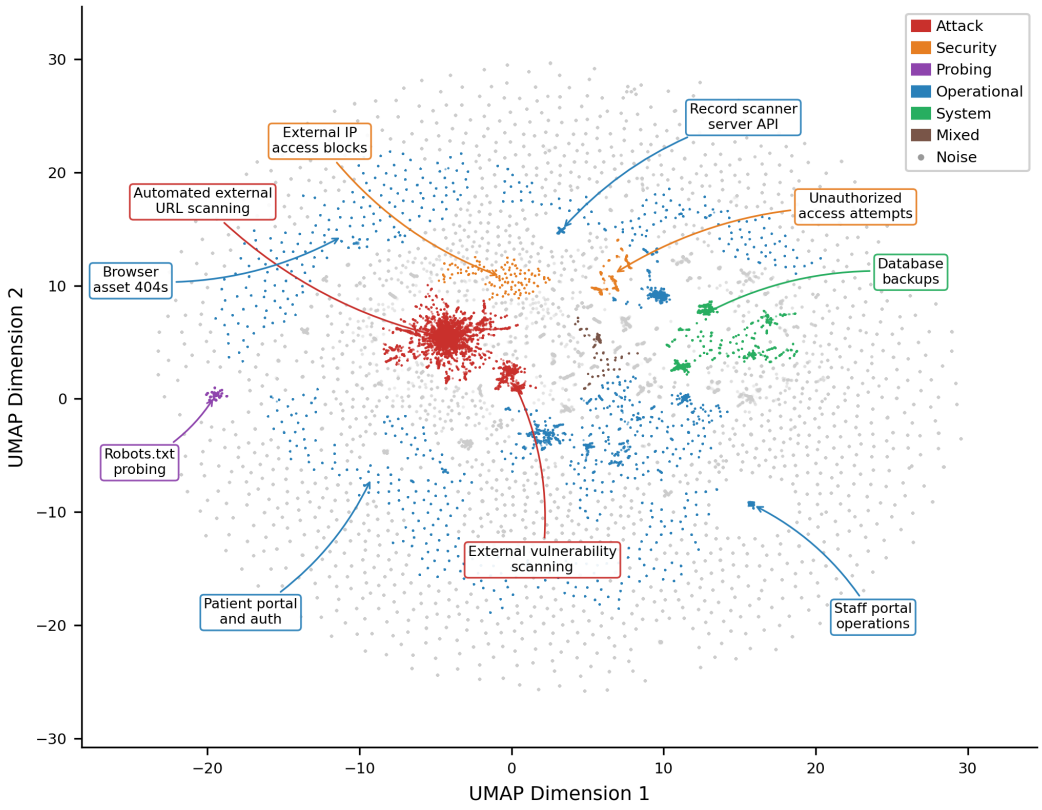

FIGURE 2. Two-dimensional UMAP projection of 158,720 log entry sentence embeddings, colored by HDBSCAN cluster category (26 clusters, silhouette score 0.428). Arrows indicate labeled clusters across six categories: attack, security, probing, operational, system, and mixed

blocks (1,330 entries) where the IP-based middleware denied requests from non-private addresses. A probing cluster (1,025 entries) consisted almost entirely of robots.txt requests.

The 17 operational clusters corresponded to the system's daily activities: staff portal navigation, record scanner server API calls, QR code processing, instructor approval workflows, student procedure requests, and patient portal authentication. Three system clusters contained database backup operations, configuration events, and submission processing. Table 2 summarizes the clusters by category, with operational and system clusters consolidated into broader groups for readability.

4.4. **Log volume and scanning activity.** Log volume varied across the three structural periods. Period 1 (18 days) produced 4,907 entries, averaging 273 entries per day with only the application event log active. Period 2 (74 days) produced 42,117 entries (569 per day) after the HTTP access log was introduced. Period 3 (71 days) produced 111,696 entries (1,573 per day).

The most prominent feature was a scanning event on November 11, 2025, which produced 12,670 total entries (12,552 in the application event log, 118 in the access log), compared to a typical daily volume of 300 to 500 entries during Period 2 (Figure 3). The scanning activity was concentrated in a burst lasting approximately 90 minutes during the early morning hours (UTC+7). Of these entries, 12,304 were WARNING-level "Not Found" responses to probed paths including /login, /.env, /static/, /care/, /graphql, and common web framework endpoints (e.g., WordPress and Apache Struts2). The

TABLE 2. Cluster categories identified through HDBSCAN clustering and manual annotation (26 clusters from 158,720 entries). Attack, security, and probing clusters are shown individually. Operational and system clusters are consolidated into broader groups. IP addresses are masked

Category	Cluster Label	Count (%)	Representative Activity
Attack	Automated scanning flood (404 responses)	14,165 (8.9%)	404 responses to probed paths (/login, /.env, /wp-admin)
Attack	External vulnerability scanning	2,381 (1.5%)	External probes for config files and CMS endpoints
Security	Unauthorized access attempts (internal)	1,637 (1.0%)	Internal users accessing restricted portal sections
Security	External IP access blocks	1,330 (0.8%)	IP middleware denied non-private address requests
Probing	Robots.txt probing	1,025 (0.6%)	Automated robots.txt discovery requests
Operational	Staff portal and API operations (6 clusters)	12,932 (8.1%)	Staff page views and hospital API calls
Operational	Authentication and patient portal (5 clusters)	9,333 (5.9%)	Patient login, OTP verification, session management
Operational	Clinical and education workflows (4 clusters)	7,282 (4.6%)	Scanner processing, instructor approvals, student requests
Operational	Browser artifacts and redirects (2 clusters)	8,421 (5.3%)	Client-side 404s and route redirects
System	Database backups and maintenance (3 clusters)	5,185 (3.3%)	Scheduled database snapshots and config events
Mixed	Mixed external probes and rendering errors	1,392 (0.9%)	Combined external scanning and internal template errors
Noise	Unclustered entries	93,637 (59.0%)	Diverse semi-unique operational events

range of probed paths included authentication pages, configuration files, content management system paths, and Java framework endpoints.

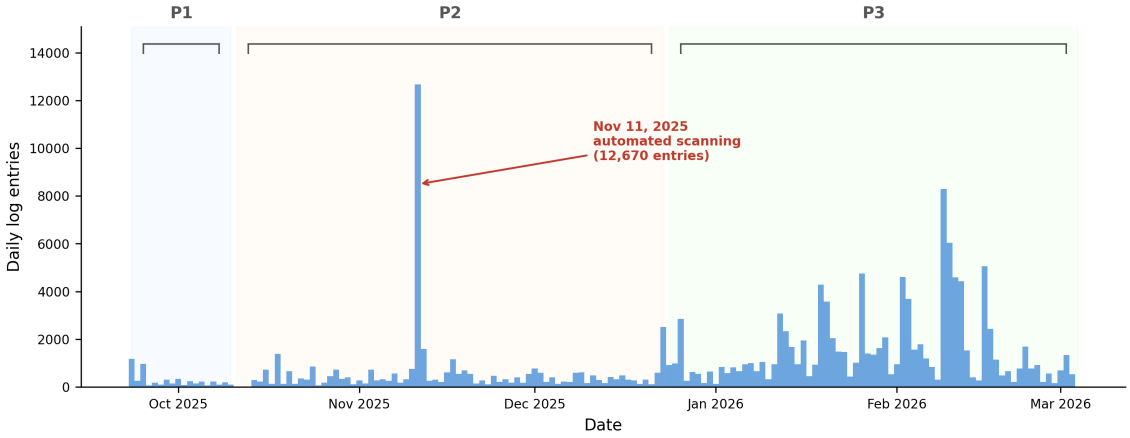

FIGURE 3. Daily log volume across the 163-day study period (158,720 entries). The three structural periods (P1, P2, P3) are shaded. The November 11, 2025 automated scanning event (12,670 entries) is annotated

5. DISCUSSION

Rule-based intrusion detection systems require predefined signatures that must be maintained as new attack patterns emerge. The LLM-based approach used in this study classified log entries based on the semantic content of requests, identifying patterns such as Java Naming and Directory Interface (JNDI)/Log4Shell injection attempts and SQL injection probes without requiring specific signatures for each variant. The LLM also identified contextual anomalies that would be difficult to detect with pattern matching alone. The external IP browsing internal staff pages and the OTP rate limit abuse were both classified based on the system description provided in the prompt, which allowed the model to distinguish expected internal traffic from suspicious external access. The absence of any high-impact classifications across all 38 batches suggests that the application's layered defenses (bot verification, rate limiting, IP-based access control) prevented the identified attacks from succeeding. These classifications are model-dependent and may vary with different LLMs or prompt formulations. Exposing the

application to the internet for patient self-enrollment introduced measurable security exposure: 10.4% of all log entries originated from attack traffic, and a single-day scanning event generated more entries than any typical day of legitimate use. The range of probed paths (authentication pages, configuration files, content management system paths, Java framework endpoints) suggests automated scanning tools executing predefined lists of common web application attack vectors rather than targeted probing of this specific application.

The traffic composition of this system differs substantially from commonly used benchmarks. The CSIC 2010 dataset [5] contains approximately 61,000 requests generated using automated tools against a single web application. The dataset in this study contains 158,720 entries generated by a production system serving multiple client types, including patient mobile browsers, messaging platform webhooks, desktop scanner clients, automated health monitoring probes, database backup processes, and external scanners with differing sophistication levels. The embedding-based clustering represented this heterogeneity: 17 of 26 identified clusters corresponded to distinct operational traffic patterns, while only 5 clusters contained security-relevant activity. Methods trained or evaluated on benchmark datasets may not account for this ratio, particularly the volume of legitimate automated traffic (hospital API polling, scanner integrations, backup snapshots) that can superficially resemble scanning activity.

The 59% noise rate in HDBSCAN clustering reflects the diversity of log messages in a production system, where many entries are semi-unique operational events that do not form dense semantic groups. HDBSCAN labels entries outside dense regions as noise by design [14]. The clustered 41% contains the most repetitive patterns in the data, and the silhouette score of 0.428 indicates that these clusters are moderately well separated in the UMAP projection. For practical purposes, even a moderate silhouette score is sufficient when the goal is visual exploration and category-level summarization rather than precise entry-level classification.

The approach uses a two-stage design that balances classification capability with data protection requirements. For the classification stage, a cloud-based LLM API (GPT-5 Nano) processed all 158,720 entries in 38 batches consuming approximately 40 minutes, providing batch-level threat assessments without requiring predefined rules. For the exploration stage, a locally deployable embedding model (all-MiniLM-L6-v2, 384 dimensions) generated sentence embeddings entirely on local hardware, avoiding transmission of raw log data to external providers. This local processing aligns with the Thailand Personal Data Protection Act and similar regulations governing healthcare data. The combination of periodic cloud-based classification with local embedding-based visualization could serve as a foundation for automated auditing systems in healthcare institutions, where security review is necessary but dedicated security staff may not be available. Whether security research analyzing system-generated log data requires institutional ethics review remains context-dependent [10]. Macnish and van der Ham [15] argued that current governance mechanisms for cybersecurity research ethics are inadequate, and a review of 1,154 top-tier security papers found ethics reporting practices remain inconsistent across the field [16].

The three structural periods in the dataset illustrate a practical challenge for longitudinal log analysis: logging infrastructure itself evolves alongside the application it monitors. The transition from single-source logging (Period 1) to dual-source logging with improved IP capture (Period 3) affected which analyses could be performed on which portions of the data. Systems deploying AI-assisted auditing should anticipate schema evolution and design their analysis pipelines to accommodate format changes across time.

For healthcare institutions without dedicated security operations centers, periodic AI-assisted auditing offers a practical middle ground between no monitoring and continuous real-time analysis. The batch processing model used here, where accumulated logs are audited at regular intervals, aligns with the resource constraints of smaller institutions. The embedding-based visualization (Figure 2) provides

a compact overview of the security landscape that non-specialist administrators can use to prioritize review: attack-related clusters can be identified visually and investigated first, while routine operational clusters require no immediate action. Institutions adopting this approach should consider the trade-off between audit frequency and detection latency. Daily audits would detect scanning events within 24 hours, while weekly or monthly schedules increase the gap between occurrence and detection. Supplementary real-time alerting for volume anomalies would complement periodic batch auditing and reduce this latency.

6. Limitations

This study analyzed logs from a single institution and application, limiting generalizability to other healthcare settings. The LLM classification and embedding-based clustering were used primarily for exploration and visualization of log patterns rather than as a validated detection system. No labeled ground truth exists for this dataset, so the classifications and cluster annotations represent the LLM's assessment rather than confirmed security labels. The LLM classification depends on the specific model and prompt used, and different configurations may produce different results. Validating these methods against labeled datasets and alternative approaches remains outside the scope of this exploratory study but would strengthen future implementations.

7. Conclusion

LLM batch classification and embedding-based clustering applied to log entries from a healthcare web application identified semantic clusters spanning attack, security, probing, operational, and system categories, with attack-related traffic accounting for 10.4% of all entries. The two-stage design combines periodic cloud-based LLM classification with locally deployable sentence embedding, keeping raw log data on local infrastructure while providing both threat assessment and visual exploration of traffic patterns. This approach offers a practical complement to rule-based monitoring for healthcare institutions without dedicated security staff, and could serve as a foundation for automated auditing systems in similar settings. Future work should validate the approach across multiple institutions and investigate integration with real-time alerting to reduce detection latency.

Statements and Declarations

The authors declare no conflict of interest. AI tools (GPT-5 Nano) were used for the log classification stage described in the Methods section. The authors take full responsibility for the content of this manuscript.

Acknowledgments

The authors acknowledge the School of Dentistry, University of Phayao for supporting this research.

References

[1] E. Carini, L. Villani, A. M. Pezzullo, A. Gentili, A. Barbara, W. Ricciardi, and S. Boccia. The impact of digital patient portals on health outcomes, system efficiency, and patient attitudes: updated systematic literature review. *Journal of Medical Internet Research*, 23(9):Article ID e26189, 2021.

[2] Z. Durumeric, M. Bailey, and J. A. Halderman. An internet-wide view of internet-wide scanning. In *Proceedings of the 23rd USENIX Security Symposium*, pages 65-78, San Diego, California, 2014. University of Michigan, Michigan.

[3] L. Coventry and D. Branley. Cybersecurity in healthcare: a narrative review of trends, threats and ways forward. *Maturitas*, 113:48–52, 2018.

[4] K. Kent and M. Souppaya. *Guide to Computer Security Log Management (NIST Special Publication 800-92)*. National Institute of Standards and Technology, Gaithersburg, USA, 2006.

[5] C. T. Gimenez, A. P. Villegas, and G. A. Maranon. *HTTP Dataset CSIC 2010*. Information Security Institute of CSIC, Madrid, Spain, 2010. https://www.isi.csic.es/dataset/. Accessed Oct 15, 2024.

[6] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the International Conference on Information Systems Security and Privacy*, pages 108-116, ICISSP 2018, Madeira, Portugal, 2018. Science and Technology Publications, Portugal.

[7] M. Landauer, F. Skopik, M. Wurzenberger, and A. Rauber. System log clustering approaches for cyber security applications: a survey. *Computers and Security*, 92:Article ID 101739, 2020.

[8] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *Proceedings of the 2008 IEEE International Conference on Data Mining*, pages 413-422, ICDM 2008, Pisa, Italy, 2008. IEEE, New York.

[9] W. Guan, J. Cao, S. Qian, J. Gao, and C. Ouyang. LogLLM: log-based anomaly detection using large language models. *ArXiv Preprint ArXiv:2411.08561*, 2024.

[10] E. Buchanan, J. Aycock, S. Dexter, D. Dittrich, and E. Hvizdak. Computer science security research and human subjects: emerging considerations for research ethics boards. *Journal of Empirical Research on Human Research Ethics*, 6(2):71-83, 2011.

[11] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. MiniLM: deep self-attention distillation for task-agnostic compression of pre-trained transformers. *ArXiv Preprint ArXiv:2002.10957*, 2020.

[12] N. Reimers and I. Gurevych. Sentence-BERT: sentence embeddings using Siamese BERT-networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 3982–3992, EMNLP-IJCNLP 2019, Hong Kong, China, 2019. Association for Computational Linguistics, Pennsylvania, USA.

[13] L. McInnes, J. Healy, and J. Melville. UMAP: uniform manifold approximation and projection for dimension reduction. *ArXiv Preprint ArXiv:1802.03426*, 2018.

[14] R. J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 160–172, PAKDD 2013, Gold Coast, Australia, 2013. Springer Berlin, Heidelberg.

[15] K. Macnish and J. van der Ham. Ethics in cybersecurity research and practice. *Technology in Society*, 63:Article ID 101382, 2020.

[16] H. S. Ramulu, H. Schmitt, B. Rerich, R. Gonzalez Rodriguez, T. Kohno, and Y. Acar. Ethics in computer security research: a data-driven assessment of the past, the present, and the possible future. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 3162–3176, Taipei, Taiwan, 2025. Association for Computing Machinery, New York.