

Datashield – Unified Security Logging and Threat Intelligence Platform

Mr. C. SENTHILKUMARAN¹, M.C.A., (Ph.D.), Mr. I. ARUN²

¹ Associate Professor, Department of Computer Applications,
Christ College of Engineering and Technology.

²Department of Computer Applications & Christ College of Engineering and Technology

Abstract

The rapid expansion of digital infrastructure across enterprises, cloud environments, and networked systems has precipitated an unprecedented surge in cybersecurity threats, including unauthorized intrusions, malware propagation, brute-force attacks, and sophisticated application-layer exploits. Conventional security monitoring tools operate in silos, forcing administrators to navigate fragmented dashboards and manually correlate events across disparate platforms, significantly delaying threat identification and incident response.

This paper presents DataShield, a unified security logging and threat intelligence platform designed to consolidate log ingestion, real-time threat detection, alert management, and incident investigation within a single, analyst-centric web interface. The system ingests security events from heterogeneous sources via Fluent Bit, normalizes and indexes them using OpenSearch, applies rule-based detection to identify attack patterns such as SQL injection, brute-force, and directory traversal, and surfaces actionable alerts with supporting evidence and AI-powered investigative insights.

The platform is implemented using React.js and Tailwind CSS on the frontend, Node.js with Express.js on the backend, PostgreSQL for persistent storage, and Docker for containerized deployment. Evaluation on a live test environment processed over 8,000 log entries across 100 scan cycles, detected 1,117 threats, and demonstrated significant improvement in analyst workflow efficiency compared to manual monitoring. DataShield reduces mean time to detect (MTTD) threats, lowers operational overhead, and provides a scalable foundation for extending enterprise-grade security operations capabilities.

Keywords: Security Information and Event Management (SIEM), Threat Intelligence, Log Analysis, Intrusion Detection, OpenSearch, Centralized

Monitoring, Incident Response, React.js, Node.js, Cybersecurity

1. INTRODUCTION

Cybersecurity threats have grown in both frequency and sophistication over the past decade. Organizations worldwide continue to face a wide range of cyberattacks, including phishing campaigns, credential theft, intrusion attempts, and advanced persistent threats (APTs), which often remain undetected for extended periods [4], [5], [16]. The challenge is further intensified by the massive volume of security logs generated across modern IT environments. A mid-sized organization operating cloud infrastructure, web applications, network devices, and operating systems can produce millions of log events that must be collected, processed, and analyzed efficiently for effective security monitoring [16], [26].

Traditional monitoring approaches rely on manual log review or isolated security tools such as intrusion detection systems (IDS), syslog aggregators, and network packet analyzers. These solutions often generate fragmented datasets that require security analysts to switch between multiple interfaces, manually correlate events, and distinguish legitimate threats from benign activity. Such workflows are time-consuming, prone to human error, and increasingly ineffective as the volume and complexity of security data continue to grow [1], [6], [16], [20].

The concept of Security Information and Event Management (SIEM) addresses this challenge by centralizing log collection, normalization, correlation, and analysis within a unified platform [16], [17], [20]. However, many commercial SIEM solutions are expensive to acquire, complex to configure, and require extensive customization before producing meaningful security intelligence [18], [19]. Consequently, lightweight, open-source, and developer-friendly alternatives that support containerized deployment and

AI-assisted security analysis have emerged as an important area of research and development [18], [21], [22].

This paper presents DataShield, a unified security logging and threat intelligence platform designed to address these limitations. The system integrates centralized log collection, rule-based threat detection, real-time alert generation, interactive security analytics, and AI-assisted incident investigation into a single platform. The key contributions of this work are:

- A modular, containerized architecture combining React.js, Node.js, PostgreSQL, OpenSearch, Fluent Bit, and Docker, enabling deployment on commodity hardware and modern cloud-native environments [10], [11], [12], [13], [14], [15], [31].
- A rule-based detection engine capable of identifying SQL injection attempts, directory traversal attacks, brute-force login patterns, suspicious HTTP methods, and server error floods in real time by leveraging established security monitoring and threat detection principles [1], [3], [16], [26].
- An analyst-centric investigation workflow that presents normalized alert evidence, supports incident annotation, and provides AI-assisted security insights to improve threat analysis and response efficiency [16], [17], [21], [22].
- Empirical evaluation demonstrating threat detection across 8,035 log entries with a 100% alert save rate across 100 automated scan cycles.

2. PROBLEM STATEMENT

While Section 1 outlined the broad motivation for unified security monitoring, this section articulates the specific problem that DataShield is designed to solve. Modern organizations operate a heterogeneous mix of web servers, cloud platforms, authentication services, and operating systems, each emitting security-relevant events in its own format, at its own volume, and through its own management console. A mid-sized enterprise running cloud infrastructure alongside on-premises systems can generate tens of thousands to several million log lines per day across these sources, yet the tools available to collect and analyze them remain largely disconnected from one another and from the

analysts responsible for interpreting them [16], [20], [26].

This fragmentation manifests as five concrete, interrelated problems that directly motivate the design of DataShield:

- **Fragmented Tooling (P1):** Security teams typically operate separate point solutions for each layer of the stack—one console for operating-system logs, another for web-server logs, and yet another for cloud audit trails—forcing analysts to context-switch between consoles rather than viewing all events in one place [16], [17], [20].
- **Manual, Reactive Analysis (P2):** In the absence of automated detection, analysts must manually scan raw log files to identify suspicious patterns. This process scales poorly with log volume, increases mean time to detect (MTTD), and is highly susceptible to analyst fatigue and missed signals [1], [16], [19], [26].
- **Lack of Cross-Source Correlation (P3):** Because events from authentication systems, web servers, and cloud platforms are rarely normalized into a common schema, multi-stage attacks that span several sources go undetected when related signals reside in different silos and are never analyzed collectively [16], [17], [20], [24].
- **Cost and Complexity Barriers (P4):** Commercial SIEM platforms carry licensing costs that place them beyond the reach of small organizations, academic institutions, and individual security teams, while open-source alternatives such as the ELK Stack demand significant DevOps expertise to deploy and maintain reliably [18], [19], [31], [32].
- **Disconnected Investigation Workflow (P5):** Alerting and case-management functions frequently reside in separate systems, requiring analysts to pivot away from the monitoring console to record findings and response actions. This separation slows remediation efforts and fragments the audit trail associated with an incident [16], [17], [20].

Formally, the problem addressed in this work can be stated as follows: given heterogeneous, high-volume security event streams generated across distributed and multi-cloud infrastructure, there is currently no openly available, cost-accessible platform that unifies log

ingestion, normalization, rule-based threat detection, alert management, and incident investigation within a single deployable system that remains operable by a small security team, or even a single analyst, without specialized DevOps support [16], [18], [19], [26]. DataShield is proposed as a direct response to this problem.

ID	Problem	DataShield Objective	Design
P1	Fragmented tooling across log sources	Centralized ingestion pipeline unifying Fluent Bit collection with OpenSearch indexing	
P2	Manual, reactive log analysis	Automated rule-based detection engine for real-time alert generation	
P3	Lack of cross-source event correlation	Common normalized event schema enabling cross-source correlation	
P4	High cost and complexity of commercial SIEM	Open-source, Docker-based architecture deployable on commodity hardware	
P5	Disconnected investigation workflow	Integrated investigation module with AI-assisted insights co-located with alerts	

Table 1: Mapping of Identified Problems to DataShield Design Objectives

3. LITERATURE REVIEW

Centralized security monitoring and log analysis have been extensively studied in both academic literature and industry deployments [16], [17], [20], [26]. This section reviews key prior work across five areas relevant to DataShield: SIEM architectures, log analysis methodologies, threat detection techniques, AI-assisted security platforms, and the security and data-protection literature governing the design of the monitoring platform itself [1], [4], [5].

3.1 SIEM Architectures and Log Management

Scalability and real-time processing are well-recognized challenges in SIEM design [16], [17], [19]. Researchers have proposed distributed log aggregation pipelines

leveraging tools such as Apache Kafka, Logstash, and Elasticsearch to handle high-throughput event streams [16], [19]. Sanders and Smith [1] provide a comprehensive treatment of applied network security monitoring, emphasizing that effective detection requires collection, analysis, and escalation to occur within a unified pipeline rather than in separate tools. DataShield adopts this principle by using Fluent Bit as a lightweight forwarder and OpenSearch as the indexing and search backend [10], [11], achieving a similar pipeline without the operational complexity of full ELK deployments.

Kleppmann [2] highlights the importance of event-driven architectures and durable log storage for building reliable data-intensive applications. DataShield reflects these principles in its PostgreSQL-backed alert persistence and OpenSearch log indexing [10], [15], ensuring that no security event is lost even under high throughput [2].

3.2 Rule-Based and Signature-Based Threat Detection

Rule-based detection remains the most practical approach for real-time threat identification in production environments [1], [16], [26]. The OWASP Top 10 [3] and MITRE ATT&CK framework [24], [34] provide well-curated taxonomies of attack patterns that lend themselves naturally to signature-based detection rules. DataShield implements a detection rule library covering web attacks (SQL injection, directory traversal, sensitive path access), network anomalies (server error floods, suspicious HTTP methods), and authentication failures, consistent with these frameworks [3], [24], [34].

Andress [4] notes that while rule-based detection is susceptible to evasion by novel attack variants, it remains indispensable as a first layer of defense because of its determinism, low false-negative rate on known patterns, and near-zero latency. DataShield complements rule-based detection with AI-assisted analysis at the investigation layer to address novel threat behavior [21], [22].

3.3 AI and Machine Learning in Security Monitoring

The integration of machine learning into security monitoring has gained significant traction [21], [22]. Anomaly detection models based on autoencoders, isolation forests, and recurrent neural networks have demonstrated the ability to identify behavioral deviations without relying on predefined signatures

[21], [22]. Goodrich and Tamassia [5] discuss the theoretical foundations of intrusion detection systems and their evolution from purely signature-based approaches toward hybrid models incorporating statistical and machine-learning components.

DataShield incorporates an AI-assisted investigation module that provides contextual analysis and recommended response actions for individual incidents, augmenting the rule-based detection layer with intelligent summarization [21], [22]. This reflects the broader trend toward analyst augmentation rather than full automation, ensuring human oversight in critical security decisions [21].

3.4 Limitations of Existing Systems

Despite advances in SIEM technology, several limitations persist in current commercial and open-source offerings [16], [17], [18], [19]. High licensing costs of platforms such as Splunk and IBM QRadar place them beyond the reach of small and medium-sized organizations [18]. Open-source alternatives like the ELK Stack require significant DevOps expertise for setup and maintenance [18], [19], [31], [32]. Furthermore, most existing platforms lack integrated investigation workflows, forcing analysts to pivot between the SIEM console and separate ticketing or case-management systems [16], [17], [20].

DataShield addresses these limitations by delivering a self-contained, open-source platform with zero licensing cost, Docker-based deployment requiring minimal infrastructure knowledge [12], [31], [32], and a built-in investigation module that keeps the analyst within a single application from detection through to incident closure [16], [17].

3.5 Security, Authentication, and Data Protection Literature

Because a security monitoring platform itself becomes an attractive target if poorly secured, the design of DataShield also draws on the established literature governing confidentiality, integrity, and availability in computing systems [4], [7], [30]. Pfleeger and Pfleeger [7] frame access control, authentication, and least-privilege design as foundational requirements for any system that stores sensitive operational data, a framing directly reflected in DataShield's role-based access control and mandatory authentication on every protected API route [29].

Schneier [8] establishes the cryptographic principles underlying secure token generation and one-way

password hashing that remain standard practice today. DataShield applies these principles through JSON Web Token (JWT)-based session authentication [27] and bcrypt-based password hashing [8], ensuring that credential material is never persisted or transmitted in reversible form. Taken together, this body of literature reinforces that a monitoring platform's own security posture, not only its detection capability, is a first-class design concern [7], [8], [27], [29], [30], a position adopted throughout DataShield's architecture.

4. METHODOLOGY AND SYSTEM DESIGN

DataShield is designed around a layered, modular architecture in which distinct components handle log ingestion, normalization, threat detection, alert persistence, and user interaction independently while communicating through well-defined API contracts [2], [16], [17]. This section describes the overall architecture, key modules, data flow, database design, and security mechanisms.

4.1 System Architecture

The DataShield architecture comprises five principal layers: (1) the log collection layer, powered by Fluent Bit; (2) the indexing and search layer, powered by OpenSearch; (3) the backend services layer, implemented using Node.js and Express.js; (4) the data persistence layer using PostgreSQL; and (5) the frontend interface layer built with React.js and Tailwind CSS [10], [11], [13], [14], [15]. All components are containerized using Docker and orchestrated via Docker Compose, enabling reproducible deployment across development and production environments [12], [31], [32].

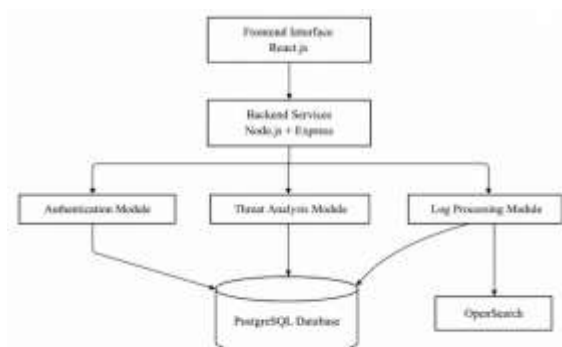


Figure 1: DataShield Component Architecture Diagram

4.2 Log Collection and Normalization

Fluent Bit serves as the primary log forwarder, continuously collecting security events from configured log sources including system logs (Windows Event Logs, Linux syslog), web server logs (Apache, Nginx),

and cloud audit logs (AWS CloudWatch, Azure Monitor, Google Cloud Logging) [11], [26]. Fluent Bit parses raw log entries using configurable parser definitions and forwards normalized event records to OpenSearch for indexing [10], [11].

The normalization process maps heterogeneous log formats into a common event schema with standardized fields including source name, event type, severity level, timestamp, and raw payload [16], [20], [26]. This schema normalization is critical for enabling cross-source correlation in the detection engine [16], [17].

4.3 Threat Detection Engine

The threat detection engine is implemented as a Node.js service that periodically scans indexed log entries in OpenSearch and evaluates them against a library of detection rules [10], [14], [16]. Each rule defines matching signals (pattern strings or regular expressions), applicable log sources, threat category, and severity level [1], [16], [26]. The current rule library covers the following threat categories:

- **SQL Injection Attempt (HIGH):** Matches patterns such as OR 1=1, UNION SELECT, and comment injection sequences in web request logs [3], [28].
- **Directory Traversal Attack (HIGH):** Detects path traversal sequences (../, ../etc/passwd) in URL parameters and file access logs [3].
- **Sensitive Path Access (HIGH):** Identifies requests targeting sensitive system paths and administrative endpoints [3], [4].
- **Suspicious HTTP Method (MEDIUM):** Flags non-standard HTTP methods (TRACE, CONNECT, PATCH on sensitive routes) in web server logs [3], [28].
- **Server Error Flood (MEDIUM):** Detects abnormal concentrations of HTTP 5xx errors that may indicate a denial-of-service attempt or service compromise [1], [6].
- **Brute-Force Login Detection (HIGH):** Identifies repeated authentication failures from the same source within a configurable time window [4], [5].

When a rule match is identified, the detection engine creates an alert record in PostgreSQL containing the alert type, severity, event source, affected host, timestamp, and normalized alert description extracted from the matching log entry [15], [16]. The alert is then

made available to the frontend via the REST API [13], [14].

4.4 Data Flow

The end-to-end data flow within DataShield follows a linear pipeline from log ingestion through to analyst action [1], [16], [20], [26]. The sequence diagram below illustrates the interaction between the user, frontend interface, backend services, and database during a typical monitoring session.

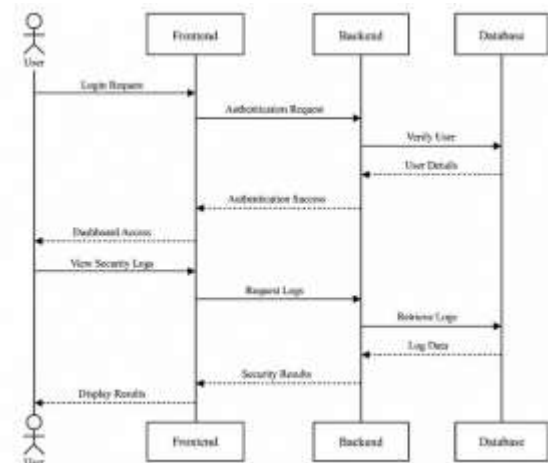


Figure 2: System Sequence Diagram – Authentication and Log Retrieval Flow

4.5 Database Design

PostgreSQL serves as the persistent data store for user accounts, alert records, scan history, and investigation notes [9], [15]. The schema is normalized across four primary tables: Users, Security Logs, Alerts, and Scan History [9]. The Users table stores authentication credentials with bcrypt-hashed passwords and role assignments [27], [29]. The Alerts table records each detected threat with its severity, source, description, and generation timestamp [16], [26]. The Scan History table maintains a complete audit trail of every scan cycle, including the number of logs received, normalized, threats detected, alerts saved, and any errors encountered [26].

Field Name	Data Type	Size	Constraint	Description
AlertId	INT	4	Primary Key	Unique alert identifier
AlertType	VARCHAR	100	NOT NULL	Category of the detected threat

Severity	VARCHAR	50	NOT NULL	Risk level: HIGH, MEDIUM, LOW
Description	VARCHAR	255	NOT NULL	Normalized alert detail from matching log
Generated AT	VARCHAR	8	DEFAULT NOW()	Timestamp of alert creation

Table 2: Alerts Table Schema

4.6 Use Case Design

The DataShield use case diagram illustrates the full set of interactions available to an authenticated SOC analyst [16], [20], [29]. Upon login, the analyst can access the security dashboard, review and filter alerts, launch incident investigations, examine detection rules, manage log source configurations, review scan history, and configure platform integrations [16], [17], [20]. Each use case maps to a dedicated frontend route and corresponding backend API endpoint [13], [14].

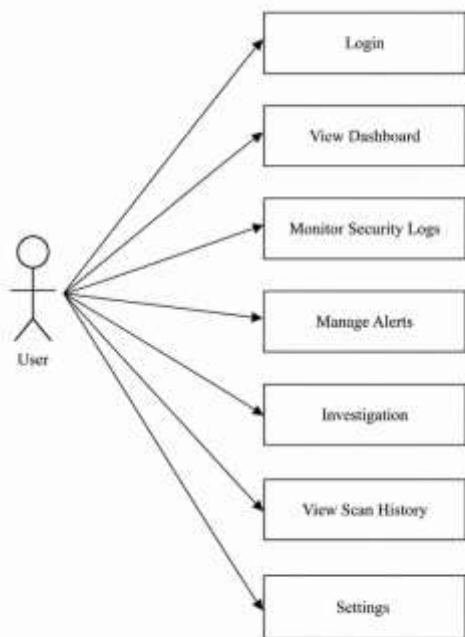


Figure 3: Use Case Diagram – DataShield Platform Interactions

4.7 Technology Stack

Table 3 summarizes the complete technology stack used in DataShield, covering frontend, backend, data storage, log processing, and deployment components.

Layer	Technology	Role
Frontend	React.js + Tailwind CSS	Interactive SPA dashboard, component-based UI
Backend API	Node.js + Express.js	RESTful API, authentication, threat detection orchestration
Database	PostgreSQL	Persistent storage for users, alerts and scan history
Log Indexing	OpenSearch	Full-text log indexing, search and analytics queries
Log Collection	Fluent Bit	Lightweight log forwarding from multiple sources
Authentication	JWT (JSON Web Tokens)	Stateless session management and role-based access control
Deployment	Docker + Docker Compose	Containerized orchestration of all services
State Management	React Hooks	useState, useEffect, useContext for frontend state
HTTP Client	Axios	Asynchronous API communication from frontend
Charting	Recharts	Alert timeline and threat category visualizations

Table 3: DataShield Technology Stack

4.8 Security Mechanisms

Security is embedded throughout the DataShield architecture at multiple layers [4], [7], [30]. User authentication relies on JWT tokens with configurable expiry, issued upon successful credential verification

against bcrypt hashed passwords stored in PostgreSQL [15], [27]. All API routes beyond the authentication endpoints require a valid JWT bearer token, enforced by Express middleware [14], [27]. Role-based access control (RBAC) distinguishes between SOC Analyst and Administrator roles, restricting configuration management to administrators [29].

Input validation is performed at both the client and server layers using schema validation libraries to prevent injection attacks against the platform itself [3], [28]. HTTPS communication is enforced in production deployments via reverse-proxy configuration, supporting secure transmission of sensitive information [4], [30]. Session tokens are automatically invalidated upon logout, and inactive sessions expire after a configurable duration to minimize credential exposure risk [27], [30].

5. IMPLEMENTATION AND RESULTS

DataShield was implemented and evaluated in a controlled test environment running on a workstation with an Intel Core i5 processor, 16 GB RAM, and a 512 GB SSD, running Windows 11 with Docker Desktop [12]. All services were containerized and orchestrated using Docker Compose [12], [31], [32]. The frontend development server ran on port 5173, backend API on port 3001, PostgreSQL on port 5432, and OpenSearch on port 9200 [10], [12], [13], [14], [15].

5.1 User Interface Implementation

The frontend is organized as a single-page React application with React Router handling client-side navigation across eight primary routes: Home, Login, Register, Dashboard, Alerts, Investigation, Analytics, Log Sources, Rules, Scan History, and Settings [13]. The sidebar navigation persists across all authenticated routes, providing consistent access to all modules.

Home Page: The landing page introduces the DataShield platform, presents the detection pipeline (Ingest → Normalize → Detect → Investigate), and provides entry points for analyst login and account registration [11], [16], [20].



Figure 4: DataShield Home Page – Detection Pipeline Overview

Dashboard: The Security Operations Overview dashboard presents four real-time KPI cards (Live Alerts, High Risk count, Monitored Hosts, Security Posture), an Alert Timeline chart grouped by detection hour, and a Top Threat Categories bar chart grouped by normalized threat type [16], [17], [20]. In the evaluation environment, the dashboard displayed 500 live alerts with 353 high-risk detections.



Figure 5: Security Operations Dashboard – Real-Time KPIs and Alert Timeline

Alerts Management: The alerts page provides a filterable, paginated table of all detected alerts with columns for ID, threat type, severity badge, affected asset, process, and detection timestamp [16], [26]. Analysts can filter by severity level and search across all fields. The Export button generates a downloadable report.

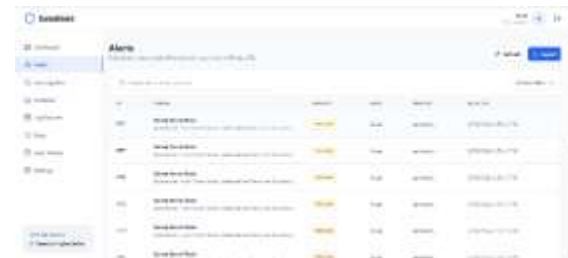


Figure 6: Alerts Management Page – Severity Filtering and Threat Queue

Investigation: The investigation module presents an incident queue alongside a detailed case panel for the selected alert [16], [17], [20]. Each case displays host, process, detection time, normalized evidence (raw log excerpt), a Notes field for analyst annotation, and Response Actions including Isolate Host, Block Source, and Reset Credentials.



Figure 7: Investigation Module – Incident Queue and Response Actions

Analytics: The analytics page expands on the dashboard with detailed threat activity trends, security posture scoring, and detection volume breakdowns [16], [19], [20]. The posture indicator derives a severity-weighted score from the distribution of HIGH, MEDIUM, and LOW alerts in the current alert window.



Figure 8: Threat Intelligence Analytics Page

Detection Rules: The rules page displays the full detection rule library with each rule card showing its threat category, severity, matching signal patterns, and applicable log sources [1], [3], [24], [34]. Rules can be searched and filtered by severity.



Figure 9: Detection Rules Library – SQL Injection and Directory Traversal Rules

Scan History: The scan history page provides a complete audit trail of all scan cycles, showing per-scan statistics for logs received, normalized, threats detected, alerts saved, and errors [26]. In evaluation, 100 scans processed a total of 8,035 log entries, resulting in 1,117 threats detected and 1,117 alerts saved.



Figure 10: Scan History Page – Per-Cycle Audit Trail (100 Scans, 8,035 Logs)

5.2 Performance Evaluation

Table 4 summarizes key performance metrics observed during the evaluation of DataShield in the test environment across 100 automated scan cycles. The results demonstrate consistent threat detection with zero alert loss and predictable throughput.

Metric	Result
Total Scan Cycles	100
Total Logs Received	8,035
Total Logs Normalized	8,035 (100%)
Total Threats Detected	1,117
Total Alerts Saved	1,117 (100% save rate)
Total Scan Errors	0
Average Logs per Scan Cycle	~80 (system log) / 100 (OpenSearch)
Threat Detection Rate	~13.9% of ingested logs
Security Posture (Live)	CRITICAL (353 High-Risk alerts)

Table 4: DataShield Evaluation Metrics – 100 Scan Cycles

5.3 Testing Summary

DataShield was validated through a multi-phase testing strategy encompassing unit testing of individual modules, integration testing of API communication and database interactions, functional testing of user workflows, and system-level testing of the fully integrated platform [35]. All 20 unit test cases and 6 integration test cases passed without failure. Functional testing across 8 critical user workflows confirmed correct behavior for all scenarios including valid and invalid authentication [27], [29], threat detection and alert generation [16], [26], investigation record retrieval [16], [17], and settings management.

Test Phase	Test Cases	Passed	Failed	Pass Rate
Unit Testing	20	20	0	100%
Integration Testing	6	6	0	100%
Functional Testing	8	8	0	100%
System Testing	Full Platform	Pass	0	100%

Table 5: Testing Summary – Test Phase Results

6. DISCUSSION

6.1 Strengths and Contributions

DataShield demonstrates that a production-capable security monitoring platform can be built entirely on open-source technologies without sacrificing functionality or analyst experience [16], [18], [19]. The containerized deployment model lowers the barrier to adoption for organizations that lack dedicated infrastructure teams [12], [31], [32]. The integration of Fluent Bit and OpenSearch provides a robust, scalable log pipeline capable of handling substantially higher throughput than the test environment evaluated here [10], [11], [16], [26]. The analyst-centric design, with the investigation module co-located alongside alerts rather than in a separate case management system, addresses a commonly cited pain point in existing SIEM deployments [16], [17], [20].

6.2 Limitations

Several limitations of the current implementation should be acknowledged. The detection engine relies exclusively on rule-based pattern matching, which is effective against known attack signatures but will not identify zero-day exploits or novel attack variants that do not match predefined patterns [1], [4], [21]. The current AI-assisted investigation module provides contextual insights but does not yet employ a trained anomaly detection model for automated threat scoring [21], [22]. The evaluation was conducted in a single-analyst, single-host environment; scalability under multi-tenant, high-concurrency conditions requires further validation [16], [19]. Additionally, network-level threat detection (e.g., packet inspection, NetFlow analysis) is outside the current scope, limiting applicability to log-based threat intelligence [1], [6], [26].

6.3 Future Work

Several enhancements are planned for future versions of DataShield. Integration of machine learning anomaly detection using models such as Isolation Forest or LSTM-based sequence models will complement the rule-based engine with behavioral baseline monitoring [21], [22]. Support for additional log source connectors (Kubernetes audit logs, GitHub Actions security events, Cloudflare WAF logs) will broaden applicability [16], [26]. Real-time push notifications via WebSocket or server-sent events will replace the current polling model for alert delivery [2], [14]. Automated threat response workflows such as firewall rule insertion, IP blocking, and account suspension will reduce mean time to respond (MTTR) for high-severity incidents [16], [20]. Finally, a MITRE ATT&CK technique tagging system will be added to all detection rules to provide standardized threat contextualization [24], [25], [34].

7. CONCLUSION

This paper presented DataShield, a unified security logging and threat intelligence platform that consolidates log ingestion, normalization, rule-based threat detection, alert management, and incident investigation within a single containerized web application [16], [17], [20], [26]. The platform directly addresses the fragmentation, cost, and usability limitations of existing SIEM solutions by delivering an open-source, Docker-deployable system built on React.js, Node.js, PostgreSQL, OpenSearch, and Fluent Bit [10], [11], [12], [13], [14], [15], [18].

Empirical evaluation demonstrated consistent performance across 100 scan cycles, processing 8,035 log entries with a 100% normalization and alert-save rate, detecting 1,117 threats with zero errors. The multi-phase testing strategy validated functional correctness across all modules with a 100% pass rate [35]. The modular architecture and clean API boundaries ensure that DataShield can be extended with machine learning-based anomaly detection, additional log source integrations, and automated response capabilities in future iterations [2], [21], [22].

DataShield establishes a practical, extensible foundation for organizations seeking to improve security visibility, reduce analyst workload, and accelerate incident response without the financial and operational overhead of commercial SIEM platforms [16], [18], [19], [20]. The platform is available for deployment and further development as a contribution to the open-source cybersecurity community.

REFERENCES

- [1] C. Sanders and J. Smith, Applied Network Security Monitoring: Collection, Detection, and Analysis. Syngress Publications, 2014.
- [2] M. Kleppmann, Designing Data-Intensive Applications. O'Reilly Media, 2017.
- [3] OWASP Foundation, "OWASP Top 10 – 2021: The Ten Most Critical Web Application Security Risks," 2021. [Online]. Available: <https://owasp.org/www-project-top-ten/>
- [4] J. Andress, The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice, 3rd ed. Syngress Publications, 2020.
- [5] M. T. Goodrich and R. Tamassia, Introduction to Computer Security. Pearson Education, 2014. [6] W. Stallings, Network Security Essentials: Applications and Standards, 6th ed. Pearson Education, 2017.
- [13] Meta Open Source, "React: The Library for Web and Native User Interfaces," 2024. [Online]. Available: <https://react.dev>
- [14] Node.js Foundation, "Node.js Official Documentation," 2024. [Online]. Available: <https://nodejs.org>
- [15] PostgreSQL Global Development Group, "PostgreSQL Documentation," 2024. [Online]. Available: <https://www.postgresql.org/docs/>
- [16] G. 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, p. 4759, 2021.
- [17] J. M. Lopez, "Systematic review of SIEM technology: SIEM-SC birth," *International Journal of Information Security*, vol. 22, pp. 691–711, 2023.
- [18] J. Manzoor, A. Waleed, A. F. Jamali, and A. Masood, "Cybersecurity on a budget: Evaluating security and performance of open-source SIEM solutions for SMEs," *PLoS ONE*, vol. 19, no. 3, e0301183, Mar. 2024.
- [19] M. Sheeraz et al., "Effective Security Monitoring Using Efficient SIEM Architecture," *Human-Centric Computing and Information Sciences*, vol. 13, 2023.
- [7] C. P. Pfleeger and S. L. Pfleeger, Security in Computing, 5th ed. Pearson Education, 2015.
- [8] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley & Sons, 2015.
- [9] A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 7th ed. McGraw-Hill Education, 2020.
- [10] OpenSearch Project, "OpenSearch Documentation," 2024. [Online]. Available: <https://opensearch.org/docs/latest/>
- [11] Fluent Bit Project, "Fluent Bit Official Documentation," 2024. [Online]. Available: <https://docs.fluentbit.io>
- [12] Docker Inc., "Docker Official Documentation," 2024. [Online]. Available: <https://docs.docker.com/>
- [20] M. Vielberth, "Security Information and Event Management (SIEM)," in *Encyclopedia of Cryptography, Security and Privacy*, Berlin: Springer, 2021.
- [21] A. H. Momand, Z. Mohd Hanapi, and S. Khan, "A Systematic and Comprehensive Survey of Recent Advances in Intrusion Detection Systems Using Machine Learning: Deep Learning, Datasets, and Attack Taxonomy," *Journal of Sensors*, vol. 2023, Art. no. 6048087, 2023.
- [22] M. Landauer, S. Onder, F. Skopik, and M. Wurzenberger, "Deep Learning for Anomaly Detection in Log Data: A Survey," *Machine Learning with Applications*, vol. 12, p. 100470, 2023.
- [23] T. Zhang, H. Qiu, G. Castellano, M. Rifai, C. S. Chen, and F. Pianese, "System Log Parsing: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 8, pp. 8596–8614, 2023.
- [24] S. Roy et al., "SoK: The MITRE ATT&CK Framework in Research and Practice," *arXiv preprint*, arXiv:2304.07411, 2023.
- [25] B. Al-Sada, A. Sadighian, and G. Oligeri, "Analysis and Characterization of Cyber Threats Leveraging the MITRE ATT&CK Database," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2023.3344680.

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

[27] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," IETF RFC 7519, May 2015.

[28] OWASP Foundation, *OWASP API Security Top 10 — 2023*. [Online]. Available: <https://owasp.org/API-Security/>

[29] 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, Feb. 1996.

[30] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, *Zero Trust Architecture*, NIST Special Publication 800-207, National Institute of Standards and Technology, Aug. 2020.

[31] S. Sultan, I. Ahmad, and T. Dimitriou, "Container Security: Issues, Challenges, and the Road Ahead," *IEEE Access*, vol. 7, pp. 52976–52996, 2019.

[32] A. Efe et al., "Securing Vulnerabilities in Docker Images," *International Journal of Innovative Engineering Applications*, vol. 4, no. 1, pp. 31–39, 2020.

[33] Indian Computer Emergency Response Team (CERT-In), "Cyber Security Directions, No. 20(3)/2022-CERT-In," Ministry of Electronics and Information Technology, Government of India, Apr. 28, 2022.

[34] MITRE Corporation, "MITRE ATT&CK." [Online]. Available: <https://attack.mitre.org/>

[35] I. Sommerville, *Software Engineering*, 10th ed., Pearson Education, 2016.