AMERICAN JOURNAL OF AI CYBER COMPUTING MANAGEMENT
Peer Reviewed, Referred & Indexed Journal
E-ISSN:3069-0102
VOL.6, NO. 3(2026)

SQLIGUARD-WEB APPLICATION SECURITY MONITORING AND ATTACK DETECTION PLATFORM

[1]G Ravinder, [2]S Sumithra, [3]N Sahithya, [4]B Durga Bhavani Sirivennela
[1]Assistant Professor, [234]Students
Department of Computer Science and Engineering,
Siddhartha Institute of Technology and Sciences, Narapally, Korremula Road,
Ghatkesar, Medchal-Malkajgiri, Hyderabad, Telangana - 500088. INDIA
shirisha.ravi@siddhartha.co.in, 23TQ1A6219@siddhartha.co.in,
23TQ1A6212@siddhartha.co.in, 23TQ1A6216@siddhartha.co.in

Abstract

SQL injection (SQLi) is a significant web application security threat that occurs when untrusted input is improperly incorporated into database queries. A successful SQL injection attack can potentially expose, modify, or delete sensitive information and may affect the integrity and availability of an application. Therefore, continuous monitoring and early detection of suspicious database-related activity are important for protecting web applications and their underlying data.

SQLiGuard is a web application security monitoring and attack detection platform designed to identify suspicious SQL injection-related activity in an authorized application environment. The platform monitors application requests, security events, and database-related indicators and analyzes them using predefined detection rules and security patterns. The objective is to identify potentially malicious behavior without requiring the system to perform harmful exploitation.

The platform processes incoming security events and identifies indicators that may be associated with SQL injection attempts. Detected events can be categorized according to severity and correlated with information such as source, affected endpoint, timestamp, request characteristics, and application context. A risk assessment component helps security teams prioritize events that require immediate investigation.

SQLiGuard provides a centralized dashboard that displays detected events, risk levels, affected endpoints, attack trends, and monitoring statistics. The system can maintain historical security logs and generate structured reports containing detected events, severity classifications, affected resources, and recommended defensive actions. This allows administrators to monitor application security and identify recurring suspicious patterns.

The primary objective of SQLiGuard is to improve the visibility and response capability of web application security teams. By combining request monitoring, SQL injection pattern detection, event correlation, risk assessment, alerting, visualization, and reporting, the platform provides a structured defensive security workflow. The system is intended for authorized application monitoring, cybersecurity education, security operations, and defensive web application security assessment.

I. Introduction

Web applications commonly interact with databases to store and retrieve information such as user accounts, transactions, product information, application records, and other business data. Because database operations are often triggered by user-controlled requests, improper handling of input can create security risks. SQL injection is one of the important application-security concerns associated with unsafe database query construction and inadequate input handling.

SQL injection attacks attempt to manipulate the way an application processes database-related input. Depending on the application's architecture and security controls, successful exploitation can potentially result in unauthorized data access or modification. The consequences can include data disclosure, alteration of application records, authentication-related problems, and service disruption. Consequently, detecting suspicious database-related activity is an important part of application security monitoring.

Traditional web application security practices commonly include secure coding, parameterized queries, input validation, database access controls, vulnerability assessments, and periodic security testing. Preventive controls are essential, but organizations can also benefit from monitoring application activity for suspicious patterns. Detection and monitoring mechanisms provide an additional defensive layer that can help security teams identify potentially malicious activity and investigate security events.

SQLiGuard is proposed as a centralized platform for monitoring and analyzing potential SQL injection-related security events. The system can collect authorized application security logs and request metadata, analyze them against predefined detection rules, and identify suspicious patterns. The platform can then classify events according to risk and provide alerts to authorized security personnel.

The main objective of SQLiGuard is to improve the detection, visibility, and analysis of potential SQL injection activity. By combining monitoring, rule-based detection, event correlation, risk assessment, visualization, and reporting, the system can help security teams respond to suspicious events more efficiently. The platform complements secure development practices and should be deployed alongside preventive controls such as parameterized queries, input validation, least-privilege database access, and secure application design.

II. Literature Survey

SQL injection has been extensively studied in web application security because database-driven applications frequently process input originating from users or external systems. Security research has identified improper query construction and insufficient input handling as important contributors to SQL injection vulnerabilities. These studies emphasize that secure application design should prevent untrusted input from being interpreted as executable database commands.

Secure coding practices are considered one of the primary defenses against SQL injection. Parameterized queries, prepared statements, appropriate input validation,

and least-privilege database accounts can significantly reduce the risk associated with unsafe query construction. Security frameworks and development guidelines commonly recommend these preventive mechanisms as part of secure software development.

Web application firewalls and intrusion detection mechanisms provide another layer of defense by monitoring application traffic and identifying suspicious patterns. Such systems can inspect requests and generate alerts when activity matches predefined security signatures or behavioral indicators. However, detection accuracy depends on the quality of detection rules, application context, and the ability to distinguish legitimate input from suspicious activity.

Security information and event management approaches provide mechanisms for collecting and correlating security events from multiple sources. Centralized event collection allows security teams to analyze application logs, database events, authentication activity, and other security information together. Correlation can provide additional context and help analysts identify patterns that may not be obvious when individual events are examined independently.

Machine learning and behavioral analysis have also been explored for detecting application attacks. Instead of depending entirely on fixed signatures, analytical models can examine patterns in requests and application behavior to identify unusual activity. Such approaches may help detect variations that do not exactly match predefined rules, although they can introduce challenges involving training data quality, false positives, explainability, and model maintenance.

Existing security monitoring solutions demonstrate the value of centralized logging, alerting, detection rules, dashboards, and incident analysis. However, organizations may still need to integrate application monitoring, SQL injection detection, event correlation, risk assessment, and reporting across multiple systems. SQLiGuard builds upon these concepts by providing a focused platform for SQL injection-related monitoring and defensive analysis within an authorized web application environment.

III. System Analysis

System analysis for SQLiGuard focuses on identifying the requirements for monitoring and detecting potential SQL injection-related security events. The system should provide authenticated access for authorized administrators and security analysts. It should collect permitted application security logs, request metadata, and relevant database-related events. A monitoring component should continuously process incoming security events according to the configured monitoring scope. The detection engine should apply predefined SQL injection indicators and security rules to identify suspicious activity. Detected events should be validated and normalized before being stored for further analysis. The system should correlate related events using information such as source, endpoint, timestamp, and event characteristics. A risk assessment module should classify detected events according to severity and potential impact. The dashboard should display alerts, affected endpoints, event trends, and risk statistics. Historical records should support investigation and comparison of security events over time. Alerting mechanisms should notify authorized security personnel about important events. Finally, the reporting module should generate

structured security reports containing findings, risk levels, and recommended defensive actions.

Existing System

Existing SQL injection defense commonly relies on secure coding practices, web application firewalls, intrusion detection systems, database monitoring, and application logs. Preventive mechanisms such as parameterized queries can reduce the likelihood of SQL injection vulnerabilities, but organizations may still require monitoring for suspicious activity. Web application firewalls can detect known patterns, but their effectiveness depends on appropriate configuration and regularly updated rules. Application logs may contain useful information but can be distributed across multiple servers and systems. Security analysts may need to manually correlate application requests with database and authentication events. Some monitoring solutions generate large numbers of alerts without sufficient contextual prioritization. Manual investigation of individual events can therefore consume significant time. Basic logging systems may provide limited visualization and historical trend analysis. Report generation may also require analysts to manually collect information from different sources. These limitations can make it difficult to maintain a centralized view of SQL injection-related security events. SQLiGuard addresses these challenges by combining monitoring, detection, correlation, risk assessment, visualization, and reporting.

Disadvantages of Existing System

- Dependence on multiple security monitoring components.
- Security events may be distributed across different systems.
- Manual event correlation can be time-consuming.
- Signature-based detection may miss previously unknown patterns.
- Large numbers of alerts can make prioritization difficult.
- Limited application-specific security context in basic monitoring systems.
- Manual investigation may require significant analyst effort.
- Historical event comparison can be difficult.
- False positives can increase unnecessary investigation effort.

Proposed System

SQLiGuard is proposed as a centralized platform for monitoring and detecting potential SQL injection-related activity in authorized web applications. The system collects permitted application requests, security logs, and relevant database-related events from configured sources. A monitoring layer continuously processes these events according to the defined application scope. The detection engine evaluates events using predefined SQL injection indicators and configurable security rules. Detected events are normalized and enriched with contextual information such as source, endpoint, timestamp, and event type. An event-correlation module groups related security events to provide a broader view of suspicious activity. The risk assessment engine evaluates event characteristics and assigns appropriate severity levels. A centralized database stores security events, alerts, analysis results, and historical records. The dashboard provides real-time or near-real-time visibility into alerts, affected endpoints, event trends, and risk distribution. Authorized analysts can

review individual events and investigate their associated context. The reporting module generates structured reports containing detected activity, severity, analysis results, and defensive recommendations. The platform therefore provides a systematic workflow for monitoring, detecting, analyzing, and responding to potential SQL injection activity.

Advantages of Proposed System

- Provides centralized SQL injection security monitoring.
- Combines event collection and detection in one platform.
- Supports configurable detection rules.
- Provides risk-based alert prioritization.
- Correlates related security events.
- Improves visibility into suspicious application activity.
- Provides centralized dashboards and security analytics.
- Maintains historical security-event records.
- Can be extended with behavioral and machine-learning-based analysis.

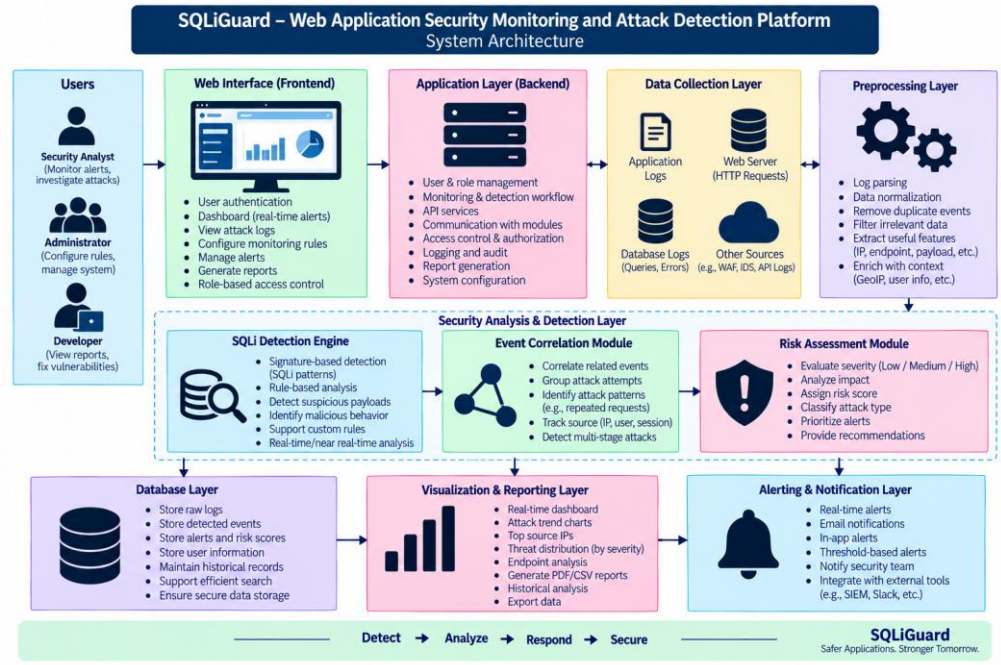
IV. Methodology

The methodology begins by authenticating the authorized administrator or security analyst and defining the application monitoring scope. The system connects to approved application logs, security events, and other permitted monitoring sources. Incoming events are collected by the monitoring layer and passed to the preprocessing module. The preprocessing module validates, normalizes, and enriches event information before analysis. The detection engine examines events against predefined SQL injection indicators and configurable security rules. Potentially suspicious events are assigned detection classifications for further analysis. The correlation module groups related events using source, endpoint, timestamp, and other contextual information. The risk assessment engine evaluates the detected activity and assigns an appropriate severity level. Important events can generate alerts for authorized security personnel. All relevant events and analysis results are stored in a centralized database for historical investigation. The dashboard presents alerts, trends, affected endpoints, and risk statistics to security teams. Finally, the reporting module generates structured reports containing security events, analysis findings, severity levels, and recommended defensive measures.

System Architecture

The SQLiGuard architecture consists of several interconnected layers designed to support continuous web application security monitoring. The User Layer contains authorized security analysts, administrators, and security operations personnel who review alerts and assessment results. The Web Interface Layer provides authentication, monitoring configuration, dashboards, alert management, investigation views, and report access. The Application Layer manages users, monitoring workflows, configuration, access control, event processing, and communication between system components. The Data Collection Layer receives approved application logs, request metadata, database security events, and other permitted monitoring information. The Preprocessing Layer validates, normalizes, filters, and enriches collected security events. The SQLi Detection Engine analyzes events using predefined SQL injection

indicators, configurable rules, and contextual security patterns. The Event Correlation Module connects related events to identify broader suspicious activity patterns. The Risk Assessment Module evaluates detected activity and assigns severity levels such as low, medium, or high. The Database Layer securely stores events, alerts, application information, detection results, risk scores, and historical records. The Analytics and Visualization Layer presents event trends, alert statistics, affected endpoints, and risk distributions. The Alerting and Reporting Layer generates notifications and structured security reports for authorized personnel. Access control, logging, secure data handling, and appropriate retention policies protect the monitoring platform. The overall architecture provides a controlled flow from security-event collection through preprocessing, SQL injection detection, correlation, risk assessment, visualization, alerting, and reporting.



V. Result and Output

VI. Conclusion

The SQLiGuard – Web Application Security Monitoring and Attack Detection Platform provides a centralized and structured approach for monitoring web applications and detecting potential SQL injection-related security events. By continuously analyzing application logs, request metadata, and security events, the platform helps security teams identify suspicious activity and understand potential threats.

The proposed system combines SQL injection detection, event correlation, risk assessment, alerting, visualization, and reporting within a single platform. This reduces the need for manual log analysis and helps analysts prioritize high-risk events. The dashboard provides a clear view of attack trends, affected endpoints, suspicious sources, and blocked requests, while historical records support further investigation.

Overall, SQLiGuard can serve as a useful foundation for web application security monitoring, cybersecurity education, security operations, and authorized defensive assessment. Future enhancements can include machine-learning-based anomaly detection, improved behavioral analysis, integration with SIEM and WAF platforms, automated incident-response workflows, and advanced threat intelligence. Along with secure coding practices such as parameterized queries and proper input validation, SQLiGuard can contribute to stronger protection against SQL injection threats.

References

[1] Kumar, R. D., Prudhviraj, G., Vijay, K., Kumar, P. S., & Plugmann, P. (2024). Exploring COVID-19 through intensive investigation with supervised machine learning algorithm. In Handbook of Artificial Intelligence and Wearables (pp. 145-158). CRC Press.
[2] Swathi, B., Vijay, K., Sushanth Babu, M., & Dinesh Kumar, R. (2024, November). Machine Learning Techniques in Cloud Based Intrusion Detection. In The International Conference on Artificial Intelligence and Smart Environment (pp. 557-564). Cham: Springer Nature Switzerland.
[3] Sv satyakrishna, shirisha rangu ,bhargavi nalacheruve.(2024) Prospective investigation on colorectal cancer with SMOTE on machine learning Algorithm
[4] Dr.G.Vishnu Murthy, BhargaviNalacheruve 1Professor, Department of computer Science & engineering, Anurag University, TS, India. 2Student, Department of computer Science & engineering, Anurag University, TS, India.
[5] V. N. S. Manaswini, K. K, C. Nigam, S. S. Ali, R. Niranjana, and Suman, “Real-Time Object Detection in Drone Surveillance Using YOLOv5,” in Proc. 2025 3rd Int. Conf. IoT, Communication and Automation Technology (ICICAT), Gorakhpur, India, 2025, pp. 1–6, doi: 10.1109/ICICAT68430.2025.11414670.
[6] B. Soundarya, V. N. S. Manaswini, M. Ayyakrishnan, R. D. Kumar, “Contextual Analysis of Big Data Analytics in Intelligent Transportation Frameworks,” in Intersection of Artificial Intelligence, Data Science, and Cutting-Edge Technologies: From Concepts to Applications in Smart Environment, Lecture Notes in Networks and Systems, vol. 1353, Cham: Springer, 2025, doi: 10.1007/978-3-031-88304-0_79.
[7] R. D. Kumar, V. N. S.Manaswini, “Applications of blockchain in smart cities: detecting fake documents from land records using blockchain technology,” in

Blockchain for Smart Cities, Elsevier, 2021, pp. 105–117, doi: 10.1016/B978-0-12-824446-3.00017-X.

[8] Tejavath Veeramma, Badarla Anil, Guguloth Ravinder, "An advanced movie recommender using collaborative filtering and sentiment analysis," International Research Journal of Modernization in Engineering Technology and Science, vol. 7, no. 7, July 2025, doi: 10.56726/IRJMETS81618.

[9] Ravi Kumar Banoth, Ramana Murthy B V, "Automatic crop recommendation system using LightGBM and decision tree machine learning models," Journal of Machine and Computing, vol. 5, no. 1, pp. 343, Jan. 2025, doi: 10.53759/7669/jmc202505026.

[10] Ravi Kumar Banoth, Dr. B.V. Ramana Murthy, "Smart agriculture through IoT and machine learning for analyzing carbon footprints," in Proc. Int. Conf. Computer Science and Communication Engineering (ICCSCE), Apr. 2025.

[11] Ravi Kumar Banoth, B. V. Ramana Murthy, "Soil image classification using transfer learning approach: MobileNetV2 with CNN," SN Computer Science, vol. 5, art. no. 199, 2024, doi: 10.1007/s42979-023-02500-x.