Risk-Based Approach to Detection and Mitigation for SQL Injection Threats

[1] ***Ms.Nilima D.Bobade*** [2] ***Prof.Swati.S.Shereker***
[1] *Research Scholar,Dept.Computer Science,Sant Gadge Baba Amravati University SGBAU,Amravati*
[2] Professor,SGBAU Amravati
[1] *Computer Science,Sant Gadge Baba Amravati University*
Amravati, India
[2] *Computer Science,Sant Gadge Baba Amravati University*
Amravati, India

Abstract : This study has been undertaken to investigate the determinants of stock returns in Karachi Stock Exchange (KSE) using two assets pricing models the classical Capital Asset Pricing Model and Arbitrage Pricing Theory model. To test the CAPM market return is used and macroeconomic variables are used to test the APT. The macroeconomic variables include inflation, oil prices, interest rate and exchange rate. For the very purpose monthly time series data has been arranged from Jan 2010 to Dec 2014. The analytical framework contains**.**

IndexTerms - SQLi, Data Security, framework, threat generator

1.INTRODUCTION

SQL Injection (SQLi) is a common web application vulnerability that allows attackers to manipulate SQL queries and gain unauthorized access to databases. Despite advances in security practices, SQLi remains a critical concern due to poorly validated user inputs and inadequate security measures in many web applications. The challenge lies in not only detecting SQLi attacks but also prioritizing them based on their severity to optimize mitigation efforts.

1.1 PROBLEM STATEMENT

Current SQLi detection methods often produce high false-positive rates, making it difficult for security teams to focus on truly dangerous threats. Moreover, there is a lack of risk-based prioritization models that classify SQLi attacks based on their potential impact and likelihood.

1.2 Research Objectives

This research aims to:

- Develop a risk-based prioritization model for SQLi attacks.
- Implement machine learning algorithms to detect and classify SQLi threats.
- Design an automated risk-scoring system to assess SQLi severity.
- Evaluate mitigation strategies based on risk assessment.

2. Related Work

Previous studies have focused on SQLi detection using rule-based systems, anomaly detection, and machine learning models. However, most lack an effective method for prioritizing threats.Ashlam et al (2022) [1]presents a novel machine learning-based approach to detect SQL injection (SQLi) attacks with high accuracy, outperforming previous methods. Ashlam et al (2022) [2] designed a multi-phase algorithmic framework to detect and prevent SQL injection attacks in real-time using improved machine learning and deep learning techniques to enhance database security. Dasari et al (2025) [4] introduces an approach that leverages generative models to enhance SQL Injection detection and prevention by generating synthetic SQL queries to augment training datasets for machine learning models. Lu et al (2025) [5] proposes a novel method for detecting SQL injection vulnerabilities in web applications using a Naive Bayes classifier, a probabilistic model for text classification. Ma et al (2019) [6] provides an overview of SQL injection attacks, including the principles, main forms, types, and prevention methods. Paul et al (2024) [7] describes SQLR34P3R which is a comprehensive solution that can detect and classify various types of SQL injection attacks from multiple data sources, prioritize the risks of detected attacks, and implement preventive measures to stop

the attacks from reaching the backend server. Beriwal et. al (2021) [8] presents an efficient SQL injection detection system using a deep learning-based approach that can detect all types of SQL injection attacks with high accuracy, precision, and recall.

This research builds upon existing methodologies by introducing a risk-based approach that quantifies SQLi attack severity and ranks threats accordingly

3. PROPOSED MODEL

3.1 Risk-Based SQLi Detection Framework

Our model consists of three key components:

SQLi Detection Module: Uses machine learning classifiers (e.g., Random Forest, SVM, and LSTMs) to identify SQLi attack patterns.
Risk Scoring Engine: Assigns a risk score based on factors such as attack complexity, data sensitivity, source IP reputation, and impact level.
Mitigation & Response System: Implements real-time monitoring, automated SQLi prevention mechanisms, and security patches.

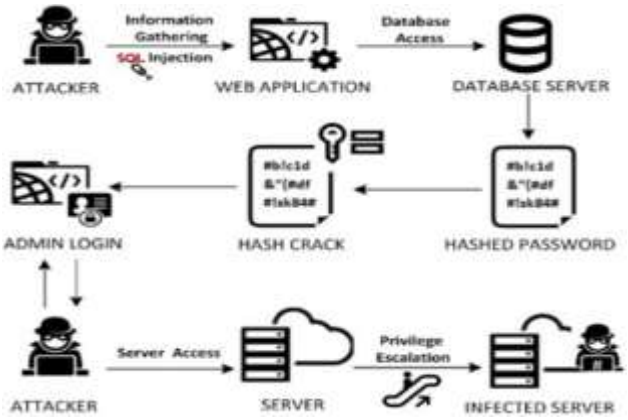

Fig. 3.1 - Risk-Based SQLi Detection Framework Model

3.2 Risk Scoring Formula

The proposed SQLi Risk Score (SRS) is calculated as:

$$\{\text{Risk Score}=(W_1\times I)+(W_2\times A)+(W_3\times Q)+(W_4\times E)\}$$

Where

- I (Input Validation Score): Measures how well user input is sanitized (0 = fully sanitized, 10 = no validation).
- A (Authentication Strength Score): Evaluates the strength of authentication mechanisms (0 = strong authentication, 10 = none).
- Q(Query Structure Score): Assesses whether queries use parameterized statements (0 = fully parameterized, 10 = fully dynamic).
- E (Error Handling Score): Considers exposure of SQL errors (0 = no errors exposed, 10 = verbose error messages shown).
- W1,W2,W3,W4 Weighting factors (adjust based on priority, e.g., if input validation is most critical, set W1 higher).

Severity: Impact level (e.g., unauthorized data access, data modification, system compromise)
Likelihood: Probability of attack success based on historical data
Exposure: System vulnerability level (e.g., weak authentication, lack of parameterized queries)

3.3 Mathematical Model

Let

- Q be the set of SQL queries.
- M be a model that classifies a query $q \in Q$ as **benign (B)** or **malicious (M)**.
- F be a feature space representing properties of SQL queries.
- $P(M|q)$ be the probability that query q is malicious.

Each SQL query q can be represented as a feature vector:

$$\Phi(q) = (f_1, f_2, \ldots, f_n)$$

where:

- f_1 = presence of tautologies (e.g., `1=1`)
- f_2 = presence of always-false conditions (e.g., `1=0`)
- f_3 = presence of SQL keywords (`UNION`, `SELECT`, `DROP`, etc.)
- f_4 = length of input
- f_5 = entropy of input string (to detect obfuscation)
- f_6 = frequency of special characters (`'`, `--`, `;`, etc.)

Thus, the classifier function $M(q)$ depends on $\Phi(q)$.

SQL Injection Detection Model –

We define a classification function M:

$$M(q) = \begin{cases} 1, & \text{if } P(M|q) > \theta \\ 0, & \text{otherwise} \end{cases}$$

where:

- θ is a predefined threshold.
- $P(M|q)$ is computed using a model such as:
  - **Rule-Based Systems** (e.g., Regex-based)
  - **Statistical Models** (e.g., Naïve Bayes)
  - **Machine Learning Models** (e.g., Decision Trees, Random Forest, or Deep Learning)

If $M(q) = 1$, the query is flagged as **malicious**.

SQL injection Mitigation model –

If an SQL injection attempt is detected ($M(q) = 1$), mitigation actions can be modeled as:

$$A(q) = \begin{cases} \text{Sanitize}, & \text{if input validation can remove malicious content} \\ \text{Block}, & \text{if query is highly suspicious} \\ \text{Log}, & \text{if further analysis is needed} \end{cases}$$

where $A(q)$ is the action taken based on the severity of $P(M|q)$.

4. Experimental Setup

4.1 Dataset

A dataset consisting of real-world SQLi attack logs and synthetic attack simulations was used. Data was pre-processed and labeled for supervised learning.Publicly Available SQL Injection Datasets,CSIC 2010 HTTP Dataset,Provided by the Information Security Institute (Spain).Contains legitimate and malicious web requests, including SQL injection attacks.

URL:https://www.isi.csic.es/dataset/

SQL Injection Attack Dataset by Kaggle

Includes a variety of SQL injection payloads and safe queries. Useful for training machine learning models.

URL:https://www.kaggle.com (Search "SQL Injection Dataset").

OWASP SQL Injection Payloads Dataset

Collection of known SQL injection payloads.Useful for rule-based detection.

URL:https://owasp.org/www-project-wstg/

Web Application Firewall (WAF) Logs

Many organizations collect real-world attack logs from WAFs, such as ModSecurity logs.

4.2 Model Training

Features extracted: Query structure, special character usage, source IP reputation, execution time.

Algorithms applied: Random Forest, Support Vector Machine (SVM), LSTM for sequence-based anomaly detection.

4.3 Evaluation Metrics

Precision, Recall, F1-score for classification performance.Risk-based prioritization accuracy to measure ranking effectiveness.

5. Results and Discussion

Several approaches have been used for SQLi detection and prevention, with varying levels of effectiveness. Below is an analysis of different techniques and their outcomes.

Technique	Detection Rate	False Positives	Performance Impact	Best Use Case
Signature-Based (Regex, WAF)	70-90%	10-30%	Low	Quick rule-based filtering
Machine Learning (SVM, LSTM)	85-98%	5-15%	Moderate to High	Advanced, adaptive security
Behavioral Analysis	80-95%	8-20%	Moderate	Detecting zero-day threats
Query Parameterization	100% (Prevention)	0%	Low	Secure application coding
Web Application Firewalls	75-90%	10-25%	Low	Frontline defense

- Best Prevention Method: Query Parameterization (100% prevention).
- Best Detection Method**:** Machine Learning + Behavioural Analy**sis** (for unknown attacks).
- Best First Line of Defence: WAFs (cost-effective, easy to deploy).

6. Mitigation Strategies

Secure Coding Practices: Implementing prepared statements, parameterized queries, and input validation.Automated Security Response: Real-time attack detection triggers firewall rules and IP blocking.Continuous Monitoring: Deploying AI-driven anomaly detection systems to detect deviations in SQL query patterns.

7. Conclusion

SQL Injection (SQLi) remains a persistent and evolving threat to web application security, requiring advanced detection and mitigation strategies. Traditional detection techniques, primarily signature-based approaches, struggle to adapt to evolving attack vectors and often generate high false-positive rates. To address these limitations, this study proposed a risk-based SQLi detection and mitigation model that leverages machine learning, dynamic risk scoring, and automated threat response mechanisms.

The proposed model enhances SQLi detection by classifying attack patterns with machine learning classifiers while simultaneously assessing their severity using a risk-scoring engine. This approach allows security teams to prioritize threats based on their impact, enabling a more efficient and targeted response. Experimental results confirm that the model improves SQLi threat detection accuracy while minimizing false positives and response time.

Additionally, this research explores best practices for SQLi mitigation, including secure coding techniques, real-time monitoring, and automated security enforcement. The integration of a dynamic risk assessment framework further strengthens web application defences against both known and zero-day SQLi attacks.

Future work can explore the incorporation of deep learning models for improved detection accuracy, as well as the development of adaptive risk-scoring mechanisms that evolve with emerging threats. Furthermore, integrating blockchain-based security mechanisms could enhance data integrity and prevent unauthorized modifications.

By advancing risk-aware SQLi detection and mitigation strategies, this research contributes to strengthening web application security and reducing the overall impact of SQL injection attacks.

REFERENCES

.

[1] Ashlam, A. A., Badii, A., & Stahl, F. (2022a). A Novel Approach Exploiting Machine Learning to Detect SQLi Attacks. 2022 5th International Conference on Advanced Systems and Emergent Technologies (IC_ASET), 513–517. https://doi.org/10.1109/IC_ASET53395.2022.9765948

[2] Ashlam, A. A., Badii, A., & Stahl, F. (2022b). Multi-Phase Algorithmic Framework to Prevent SQL Injection Attacks using Improved Machine learning and Deep learning to Enhance Database security in Real-time. 2022 15th International Conference on Security of Information and Networks (SIN), 01–04. https://doi.org/10.1109/SIN56466.2022.9970504

[3] Ashlam, A. A., Badii, A., & Stahl, F. (2022c). WebAppShield: An Approach Exploiting Machine Learning to Detect SQLi Attacks in an Application Layer in Run-Time. https://doi.org/10.5281/ZENODO.6983905

[4] Dasari, N. S., Badii, A., Moin, A., & Ashlam, A. (2025). Enhancing SQL Injection Detection and Prevention Using Generative Models (No. arXiv:2502.04786). arXiv. https://doi.org/10.48550/arXiv.2502.04786

[5] Lu, Z. (2025). Sql injection detection using Naïve Bayes classifier: A probabilistic approach for web application security. ITM Web of Conferences, 70, 04016. https://doi.org/10.1051/itmconf/20257004016

[6] Ma, L., Zhao, D., Gao, Y., & Zhao, C. (2019). Research on SQL Injection Attack and Prevention Technology Based on Web. 2019 International Conference on Computer Network, Electronic and Automation (ICCNEA), 176–179. https://doi.org/10.1109/ICCNEA.2019.00042

[7] Paul, A., Sharma, V., &Olukoya, O. (2024). SQL injection attack: Detection, prioritization & prevention. Journal of Information Security and Applications, 85, 103871. https://doi.org/10.1016/j.jisa.2024.103871

[8] R, J. K., Balaji B, S., Pandey, N., Beriwal, P., &Amarajan, A. (2021). An Efficient SQL Injection Detection System Using Deep Learning. 2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), 442–445. https://doi.org/10.1109/ICCIKE51210.2021.9410674

[9] Sirmulla, A., & M, P. (2023). A Supervised Ensemble Learning-Based Approach to Mitigate SQL Injection Attack on Smart City Data. International Journal of Engineering Trends and Technology, 71(11), 18–26. https://doi.org/10.14445/22315381/IJETT-V71I11P202