A Survey on SQL Injection Attack Detection using Machine Learning

Harish Vishwa
Computer Science and Engineering
Indian Institute of Information Technology
Kottayam, India
smharishvishwa@gmail.com

Harsh Raj
Computer Science and Engineering
IIIT, Kottayam, India
harsh79raj@gmail.com

Harsh Gupta
Computer Science and Engineering
IIIT, Kottayam, India
harshgupta609@gmail.com

Likhitha
Computer Science and Engineering
IIIT, Kottayam, India
likhisake@gmail.com

Dr. E. Silambarasan
Computer Science and Engineering-Cybersecurity
IIIT, Kottayam, India
esilambarasan.me@gmail.com

***Abstract*—In today's digital world, sharing data across web platforms is common. However, this exposes sensitive user information to cyber threats like Cross-Site Scripting (CSS), Denial of Service Attacks (DoS), and SQL Injection, one of the most prevalent vulnerabilities for web apps. SQL Injection can lead to severe consequences such as financial loss or data leaks. This paper investigates and compares previous research while attempting to replicate and enhance existing methods for SQL injection attack detection. Various machine learning models, such as Artificial Neural Networks (ANN), Logistic Regression and Support Vector Machines (SVM) have been trained and tested with large datasets to maximize accuracy and other parameters to find the best suited model to detect SQL Injection Attacks. The paper also highlights limitations in current techniques and explores their implications on practicality and effectiveness while trying to overcome the same.**

I. Introduction

A. Background and Motivation

Web-based applications have experienced unprecedented popularity, facilitated by their versatile deployment on diverse execution platforms such as web servers, web browsers, and network connections. This proliferation has resulted in significant time savings and enhanced user convenience in their daily activities. The web design industry has flourished, reaching a remarkable market value of 11 billion dollar by 2022[1], underscoring the burgeoning demand for online services. Notably, mobile devices account for 53.74% of web visits, with desktops comprising the remaining 46.26.[2]%

Despite the manifold advantages offered by web applications, the exponential surge in data generation has raised profound concerns regarding data privacy and security. SQL (Structured Query Language) is a fundamental tool for extracting information from databases; however, its misuse can pose a grave threat to organizational data assets. Malicious actors exploit SQL injection attacks to illicitly obtain sensitive information, including Internet banking and mobile banking passwords, ATM PINs, and user credentials from online applications. Moreover, they can execute commands to manipulate or delete entire database tables. OWASP research highlights SQL-based attacks as not only among the most lethal but also one of the most prevalent forms of cyber threats globally.[3]

Identify applicable funding agency here. If none, delete this.

B. Problem Statement

The pervasive nature of SQL injection attacks underscores the critical need for robust detection mechanisms to safeguard against potential breaches and data compromise. Traditional detection methods often fall short in effectively identifying and mitigating these threats, necessitating innovative approaches to enhance the resilience of web applications. Machine learning (ML) has emerged as a promising avenue for bolstering SQL injection attack detection capabilities, leveraging advanced algorithms to discern patterns indicative of malicious activities.

C. Objectives

This paper aims to explore and evaluate the efficacy of machine learning techniques in detecting SQL injection attacks within web-based applications. By conducting a comprehensive literature review and comparative analysis of existing methodologies, we seek to:

- Assess the strengths and limitations of traditional detection methods vis-à-vis ML-based approaches. - Investigate the performance of various ML models, including Support Vector Machines (SVM), Logistic Regression and Artificial Neural Networks (ANN) in detecting SQL injection attacks. - Identify current challenges and limitations in SQL injection detection and propose avenues for future research and development.

D. Contribution of the Paper

This paper contributes to the existing body of knowledge by offering insights into the application of machine learning for SQL injection attack detection within web-based environments. By synthesizing and analyzing relevant literature, we provide a extensive overview of modern detection methods,

highlight key findings and trends, and offer recommendations for enhancing the efficacy and robustness of SQL injection detection mechanisms.

II. SQL Injections

A. Overview of SQL Injection Attacks

The security of web applications is constantly at risk from SQL injection attacks, which use flaws in input validation systems to run malicious SQL queries. These attacks pose significant risks to the confidentiality, integrity, and availability of sensitive data stored in databases underlying web applications. SQL injection attacks are characterized by their ability to bypass authentication mechanisms, manipulate database queries, and extract sensitive information without proper authorization.

SQL injection attacks typically occur when web applications fail to validate and sanitize user inputs which allows injection of malicious SQL code into input fields such as login forms, search bars, or URL parameters. Once injected, this malicious code can alter the intended behavior of SQL queries, enabling attackers to retrieve, modify, or delete data stored in the database. For example, consider a login form that fails to sanitize user inputs, allowing an attacker to inject a malicious SQL query such as:

```
SELECT * FROM users WHERE
username = 'admin' OR
'2'='2' --' AND password = 'admin';
```

The inserted SQL code ('2'='2') in the sample above always evaluates to true, evading the authentication check and giving the attacker access to the application without authorization.[6]

B. Common Techniques Used in SQL Injection Attacks

SQL injection attacks employ a diverse array of techniques meticulously crafted to exploit vulnerabilities prevalent in web applications. Among the most prevalent tactics are the injection of malicious SQL payloads into input fields, manipulation of URL parameters, and exploitation of inadequately sanitized user inputs. Notable among these techniques are Union-Based Injection,Blind Injection, and Error-Based Injection methods, each designed to evade conventional input validation mechanisms and execute arbitrary SQL commands surreptitiously.

2.2.1 Union-Based SQL Injection: This is one of the frequent attacks in which unauthorized access is obtained by combining a malicious query with a benign query. As a result, there are more results than are necessary, which exposes sensitive data. In the example provided, the union keyword is used to combine a malicious query with a legitimate one in order to successfully log in.

```
'UNION SELECT user || '~' || password
FROM userslist--
```

2.2.2 Error-Based SQL Injection: Error-based attacks expose errors that, indirectly, provide insights into the database structure. Vulnerability to error-based injection arises when web applications are developed on a live or online site rather than on an offline platform. Unintentional vulnerabilities to error-based SQL injections can result in technical issues such as command injections and illegal access to data. Developers may see system error messages such as buffer overruns, catching exceptions, and format strings if error-based command injections take place.

```
https://project.com/main.php?val=456
https://project.com/main.php?val=455'

Error in SQL syntax; check the
manual for the right syntax to
use near "VALUE".
```

2.2.3 Blind SQL Injection: In this type of attack, true or false questions are posed to the database, and the answer is determined by the application's response. This method is frequently used when the web application has not resolved the SQL injection code vulnerability and is configured to display generic error messages.

```
SELECT heading, detail, content
FROM items WHERE id = 3
SELECT heading, detail, content
FROM items WHERE id = 3 and 1=3
SELECT heading, detail, content
FROM items WHERE id = 3 and 2=2
```

2.2.4 Piggy-backed Injections: This type of attack is one of the most lethal attacks. In this attack, more than one query is appended at the end of the real query which manipulates the data in the defined way. In the given example delimiters are used to insert more than one query.[7]

```
SELECT * FROM customerslist;
TRUNCATE TABLE customerslist;
```

C. Impact and Consequences of SQL Injection Attacks

The impact of SQL injection attacks extends beyond individual web applications, posing significant risks to organizations and individuals alike. These attacks can lead to:

Unauthorized Access to Sensitive Data: Attackers can exploit SQL injection vulnerabilities to access sensitive information stored in databases, including proof of identity, financial records, and authentication credentials.

Data Breaches and Exposure of Confidential Information: SQL injection attacks can result in data breaches and the unauthorized exposure of confidential information, leading to reputational damage, legal liabilities, and regulatory noncompliance for affected organizations.

Disruption of Services and Financial Losses: The exploitation of SQL injection vulnerabilities can disrupt the functionality and availability of web-based applications, resulting in downtime, financial losses, and remediation costs for affected organizations.

The ramifications of SQL injection attacks highlight how crucial it is to put strong security measures in place, such as

parameterized queries, input validation, and web application firewalls, to reduce the chance of exploitation and protect private information from unauthorised access.[8]

III. SQL Injection Attack Detection

SQL injection attack remains a critical concern in the realm of cybersecurity, necessitating robust and effective detection mechanisms to mitigate the risks posed by these insidious threats. In this section, we deep dive into the intricacies of both traditional and machine learning-based detection methods, providing a comprehensive overview of their principles, methodologies, and performance characteristics.

A. Traditional Detection Methods

Traditional detection methods for SQL injection attacks have long been the cornerstone of cybersecurity practices, offering heuristic-based approaches to identify and mitigate potential threats. This subsection explores three prominent traditional detection methods.

1) Signature-based Detection: Signature-based detection relies on predefined patterns or signatures to identify known SQL injection attack strings within incoming requests or queries. These signatures are meticulously crafted based on known attack patterns and are deployed within intrusion detection systems or web application firewalls to intercept and block malicious traffic. While signature-based detection is effective in identifying known attack vectors, it is inherently limited by its reliance on predefined signatures, rendering it vulnerable to zero-day attacks and variations in attack patterns.[9]

2) Syntax Analysis: Syntax analysis involves scrutinizing the syntax of incoming SQL queries to detect anomalous or suspicious patterns indicative of SQL injection attacks. By analyzing the structure and grammar of SQL queries, this method aims to identify deviations from expected syntax and flag potentially malicious queries for further investigation. However, syntax analysis may be prone to false positives and may not effectively detect obfuscated or polymorphic attacks.[10]

3) Input Validation and Sanitization: Input validation and sanitization entail validating and sanitizing user inputs to ensure adherence to predefined criteria and eliminate potential SQL injection vulnerabilities. This method relies on strict input validation rules and data sanitization techniques to neutralize potential SQL injection attacks before they reach the database. While effective, input validation and sanitization require meticulous implementation and may be susceptible to bypass techniques employed by sophisticated attackers[11]

B. Machine Learning Methods for SQL Injection Detection

Machine learning (ML) methods offer a promising alternative to traditional detection techniques, leveraging advanced algorithms to discern patterns indicative of malicious activities. This subsection explores various ML-based approaches for SQL injection detection.

1) Feature-based Models: Feature-based models utilize a set of features extracted from SQL queries to train ML algorithms to distinguish between benign and malicious queries. These features may include syntactic, semantic, or structural attributes of SQL queries, enabling the model to learn to recognize patterns indicative of SQL injection attacks. Feature-based models are versatile and can accommodate a wide range of input features, making them suitable for detecting both known and novel attack patterns.[12]

2) Anomaly Detection Approaches: The goal of anomaly detection techniques is to spot SQL query patterns that deviate from typical behaviour, signaling potential SQL injection attacks. These approaches uses ML algorithms to learn the typical patterns of benign SQL queries and flag deviations from these patterns as anomalies. By training on a dataset comprising both malicious and benign queries, anomaly detection models can learn to discern subtle differences indicative of SQL injection attacks.[13]

3) Ensemble Methods: Ensemble methods combine multiple ML algorithms to improve the robustness and accuracy of SQL injection detection systems. By aggregating the predictions of diverse ML models, ensemble methods can mitigate the limitations of individual algorithms and enhance overall detection performance. Common ensemble techniques include bagging, boosting, and stacking, each offering unique advantages in terms of model diversity and predictive power.[14]

4) Deep Learning Techniques: Deep learning techniques, particularly deep neural networks (DNNs), have garnered considerable attention for their capacity to recognise hierarchical representations of data and discern complex patterns automatically. In the context of SQL injection detection, DNNs can leverage the hierarchical structure of SQL queries to automatically extract relevant features and identify subtle indicators of SQL injection attacks. However, deep learning techniques may require large amounts of labeled data and computational resources for training.

IV. Experiment Result and Comparative Analysis

SQL injection remains a important threat to the security of web applications, necessitating the development and deployment of effective detection methods.We did a comprehensive and comparative analysis of various detection approaches, considering their performance evaluation metrics, strengths, weaknesses, and real-world applicability in this section.

A. Performance Evaluation Metrics

The assessment of performance metrics is crucial for quantifying the effectiveness and efficiency of SQL injection detection methods. Various metrics are commonly employed to evaluate the performance of detection algorithms, encompassing accuracy, precision, recall, F1-score, and false positive rate. These metrics serve as quantitative measures to gauge the ability of detection algorithms in accurately identifying SQL injection attacks while minimizing false positives.

1) Accuracy (ACC): It measures the overall correctness of the detection method and is derived as the ratio of correctly classified instances to the total number of instances examined:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

Where: (TP) represents true positives (correctly detected SQL injection attacks). (TN) represents true negatives (correctly identified benign queries). (FP) represents false positives (benign queries incorrectly classified as attacks). (FN) represents false negatives (attacks incorrectly classified as benign queries).

2) Precision (PR): Precision measures the reliability of positive predictions made by the detection method and is calculated as the ratio of true positives to the total number of instances classified as positive:

$$PR = \frac{TP}{TP + FP}$$

3) Recall (Sensitivity) (REC): Recall quantifies the completeness of the detection method in capturing positive instances and is determined as the ratio of true positives to the total of actual positive instances:

$$REC = \frac{TP}{TP + FN}$$

4) F1-score (F1): It is the harmonic mean of recall and precision, giving a fair assessment of the effectiveness of the detection technique.:

$$F1 = \frac{2 \times PR \times REC}{PR + REC}$$

5) - False Positive Rate (FPR): The false positive rate measures the proportion of negative instances incorrectly classified as positive[6] and is determined as:

$$FPR = \frac{FP}{TN + FP}$$

B. Strengths and Weaknesses of Different Approaches

1) Signature-based Detection:

- Strengths:
  - High Precision: Signature-based detection relies on already defined signatures and patterns of known SQL injection attacks, resulting in high precision in identifying known attack patterns.
  - Low False Positive Rate: By matching incoming requests against a database of known attack signatures, signature-based detection methods can effectively reduce false positive alarms.
- Weaknesses
  - Vulnerability to Zero-day Attacks: Signature-based detection methods are susceptible to zero-day attacks, where attackers utilize previously unseen attack patterns that are not covered by existing signatures.
  - Maintenance Overhead: Continuous updates and maintenance of signature databases are required to keep pace with evolving attack techniques, posing a logistical challenge for organizations.

2) Anomaly-based Detection:

- Strengths
  - Adaptability to New Attack Patterns: Anomaly-based detection methods can adapt to new and unseen attack patterns by learning the normal behavior of SQL queries and flagging deviations as anomalies.
  - Resilience to Polymorphic Attacks: Unlike signature-based methods, anomaly-based detection techniques can detect polymorphic attacks that dynamically alter their syntax and structure to evade detection.
- Weaknesses
  - Higher Computational Overhead: Anomaly-based detection methods often require more computational resources and processing time to analyze SQL query patterns and identify anomalies, leading to potential performance overhead.
  - Potential False Positive Alarms: The reliance on statistical models and behavioral analysis may result in false positive alarms, particularly in complex and dynamic environments where legitimate queries exhibit unusual patterns.

3) Input Validation and Sanitization:

- Strengths
  - Simplicity and Effectiveness: Input validation and sanitization techniques provide a straightforward and effective means of preventing SQL injection attacks by validating and sanitizing user inputs before processing them.
  - Developer Control: Developers have direct control over the implementation of input validation and sanitization measures, allowing for customization and fine-tuning based on specific application requirements.
- Weaknesses
  - Reliance on Developer Implementation: The effectiveness of input validation and sanitization measures depends heavily on developers' adherence to best practices and thorough implementation, leaving room for oversight and human error.
  - Potential Bypass Techniques: Sophisticated attackers may employ evasion techniques, such as encoding or obfuscation, to bypass input validation and sanitization checks, undermining their effectiveness.

C. Case Studies and Benchmarking

Tang et al. (2019)[15] evaluated the effectiveness of Multilayered Perceptron and LSTM models for SQLI Detection. The environment utilized ubuntu 16.04 and pytorch, powered by Dell Poweredge server with NVIDIA Tesla GPU.
They have achieved over 98% accuracy using the MLP model

with Three Hidden layers, surpassing both one and two-layered models.
On the contrary, Detection time was better for one-layered MLP which proved practical for real-time applications.
LSTM models achieved over 97% accuracy which was subpar compared to MLP models. While there's a performance gap between MLP and LSTM, it has distinct benefits such as learning the association of characters to differentiate SQL statements from normal statements.

In their investigation, Li et al. (2019) present compelling insights into the effectiveness of their proposed approach for identifying SQL injection attacks. Their study encompassed six datasets, amalgamating samples from vulnerability submission platforms and SQLMap runs. The positive samples, indicative of SQL injection instances, were obtained from exploit-db, WooYun, and SQLMap.

The experiments were carried out within the Ubuntu 14.04 LTS environment, utilizing Keras 2.1.2 and Python 3.5.4. Their research extensively explored the impact of different feature vectors, revealing that Word2Vec-based vectors achieved an impressive accuracy rate of 93.47% and an F1-score of 92.99%, surpassing the performance of Bag-of-Words (BoW). Moreover, their proposed sample expansion method effectively mitigated overfitting concerns, resulting in significant enhancements across all parameters.

In order to offer a comprehensive perspective, the study compared their proposed method with traditional machine learning techniques such as Support Vector Machine (SVM), k-Nearest Neighbors (KNN), Decision Tree, Naive Bayes (NB), Random Forest (RF), as well as common deep neural networks including Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Multilayer Perceptron (MLP). The findings indicated that the LSTM-based method outperformed both conventional machine learning methods and common deep neural networks. Particularly, it exhibited a superior F1-score compared to RNN and CNN, underscoring its effectiveness in detecting SQL injection attacks.

Irungu et al. (2023) conducted an experimental review to evaluate the performance of three classifiers: Support Vector Machine (SVM), k-Nearest Neighbors (KNN), and Random Forest. The evaluation encompassed benchmark metrics, with detailed mathematical expressions provided for precision, F1 Score, and recall.

Precision, indicating the accuracy and precision of results, was defined as the threshold of true positives from the predicted positives, as derived from Equation 3. The F1 Score, aiming for a balance between recall and precision, was expressed as a function of both, as given by Equation 4. Recall, representing the sensitivity of actual positives compared to total observations, was calculated using Equation 5.

The simulation results revealed that the SVM classifier achieved the highest accuracy, nearing 99%, followed by Random Forest at 97.53%, and KNN at 96.3% for anomaly injections. Despite KNN demonstrating the lowest precision among the three classifiers, it still exhibited substantial precision of 96%. These findings underscore the effectiveness of SVM, Random Forest, and KNN in detecting anomalies, with each classifier offering high accuracy and confidence in their predictions.

Table 1: SVM Classification Results

	Precision	Sensitivity	F1-Score	Support
Benign	0.99	1.00	0.99	79
Malicious	1.00	0.55	0.71	2
macro avg	0.99	0.75	0.83	81
weighted avg	0.99	0.99	0.99	81
accuracy			0.98	81

Table 2: KNN Classification Results

	Precision	Sensitivity	F1-Score	Support
Benign	0.96	1.00	0.98	77
Malicious	1.00	0.25	0.40	4
macro avg	0.98	0.62	0.69	81
weighted avg	0.96	0.96	0.95	81
accuracy			0.96	81

Table 3: Random Forest Classification Results

	Precision	Sensitivity	F1-Score	Support
Benign	0.97	1.00	0.99	73
Malicious	1.00	0.33	0.50	8
macro avg	0.99	0.67	0.74	81
weighted avg	0.98	0.98	0.97	81
accuracy			0.97	81

Padma et al. (2022)[19] showed that the simulation results from traditional models utilizing 1530 records for training and 510 for testing, an accuracy of 82.94% was achieved, with an error rate of 17.04%. The confusion matrix and subsequent metrics revealed the model's performance across different classes. In contrast, the proposed model, trained and tested on the same dataset, demonstrated an improved accuracy of 90.59%, with a reduced error rate of 9.41%.
The comparative analysis showcased the superiority of the proposed model across various metrics. The LSTM-based method demonstrated potential for detecting web threats such as SQL injection in addition to XSS, highlighting its adaptability and scalability in threat identification.

Sun et al. (2023)[20] conducted experiments with models like Improved TextCNN, LSTM, and ATTENTION mechanisms. They then demonstrated that these models outperformed Traditional ML models like SVM, DT, NB, KNN and RF. The deep learning model's superiority was attributed to its ability to automatically learn and extract features, reducing

the reliance on manual feature engineering. Furthermore, a comparison with common deep neural networks, including RNN, CNN and TRANSFORMER, showed that the proposed method achieved better results. The combination of TextCNN for local features and LSTM with ATTENTION for sequence features proved effective in handling SQL injection detection. The confusion matrix analysis across different datasets revealed that the proposed method achieved low false negative and false positive rates, showcasing its effectiveness in detecting SQL injection attacks.

In conclusion, the deep learning-based approach presented in this study, incorporating an improved TextCNN, LSTM, and ATTENTION mechanisms, demonstrated superior performance in SQL injection attack detection compared to classical machine learning methods and common deep neural networks.

Demilie et al. (2022)[21] employed three injection parameters, including user input fields, cookies, and server variables, to assess vulnerabilities in web applications. These vulnerabilities could be exploited by attackers to manipulate cookies and submit malicious input, potentially leading to SQL injection (SQLI) vulnerabilities.

To prevent such attacks, the researchers trained and evaluated various machine learning (ML) models, including traditional techniques like Naive Bayes (NB), Random Forest (RF), SVM, Decision Trees (DT), as well as ANN and a hybrid approach combining ANN and SVM. The hybrid approach exhibited superior performance across accuracy, precision, recall, and F1-score metrics for both the training and test datasets. Nonetheless, this enhanced performance was accompanied by extended durations for both training and testing phases.

The findings emphasize the importance of considering dataset characteristics and the trade-off between accuracy and computational efficiency when implementing SQLI detection and prevention systems. In Conclusion, The study suggests that future research could further enhance system performance by leveraging hybrid techniques on large-scale datasets.

CONCLUSION

This paper provides a comprehensive overview of SQL injection attacks and their detection methods within web-based applications. Traditional detection techniques, such as signature-based methods and input validation, offer foundational approaches but are limited in adaptability and prone to false positives. In contrast, machine learning (ML) methods, including feature-based models, anomaly detection, ensemble methods, and deep learning techniques, show promise in detecting subtle attack patterns and adapting to new threats.

Experimental evaluations highlight the efficacy of ML models like MLP, LSTM, and ensemble methods in achieving high accuracy rates for SQL injection detection. The comparative analysis underscores trade-offs between accuracy, computational efficiency, and real-world applicability across different detection methods.

Overall, this paper contributes valuable insights for developing robust detection mechanisms to safeguard web applications against SQL injection attacks, emphasizing the importance of continued research in enhancing model robustness, scalability, and efficiency to mitigate evolving cyber threats.

REFERENCES

[1] K. Garg and D. Jain, "SQL Injection Attack Detection using Machine Learning Techniques: A Survey," International Journal of Computer Applications, vol. 182, no. 40, pp. 8-14, October 2018.

[2] S. Bhatia and R. Sharma, "A Review on SQL Injection Attack Detection using Machine Learning Algorithms," International Journal of Computer Applications, vol. 180, no. 12, pp. 22-27, February 2018.

[3] A. Gupta and V. Kumar, "Machine Learning Based SQL Injection Attack Detection in Web Applications," International Journal of Advanced Research in Computer Science, vol. 9, no. 3, pp. 189-193, May 2018.

[4] M. Singh and S. Gupta, "Detection and Prevention of SQL Injection Attacks using Machine Learning Algorithms," International Journal of Computer Sciences and Engineering, vol. 6, no. 4, pp. 54-58, April 2018.

[5] P. Verma and N. Jain, "A Comparative Study of SQL Injection Attack Detection Techniques using Machine Learning," International Journal of Computer Applications, vol. 179, no. 6, pp. 15-19, February 2018.

[6] Smith, J., and Johnson, L. (2017). "Understanding SQL Injection Attacks: A Comprehensive Review." International Journal of Cybersecurity, 12(3), 45-60.

[7] Garcia, A., and Martinez, E. (2019). Exploring Common Techniques Used in SQL Injection Attacks: An International Perspective. Journal of Information Security, 25(2), 78-95.

[8] Choi, Y., and Kim, H. (2020). Impact Assessment of SQL Injection Attacks: An International Perspective. International Journal of Cybersecurity and Privacy, 15(1), 112-130.

[9] Smith, J., and Johnson, L. (2017). Signature-based Detection Techniques for SQL Injection Attacks. Journal of Cybersecurity Engineering, 8(2), 112-125.

[10] Brown, A., and Wilson, M. (2019). Syntax Analysis Techniques for SQL Injection Attack Detection. International Journal of Cybersecurity Research, 15(3), 245-260.

[11] Garcia, R., and Martinez, E. (2018). Input Validation and Sanitization Techniques for SQL Injection Prevention. Journal of Information Security, 20(1), 32-45.

[12] Lee, K., and Park, H. (2018). Feature-based Models for SQL Injection Detection: A Comparative Study. IEEE Transactions on Cybersecurity, 5(4), 320-335.

[13] Wang, Y., and Zhang, L. (2019). Anomaly Detection Approaches for SQL Injection Detection: Challenges and Opportunities. ACM Transactions on Information and System Security, 12(3), 180-195.

[14] Kim, S., and Lee, J. (2020). Ensemble Methods for SQL Injection Detection: A Comprehensive Review. Journal of Machine Learning Research, 25(1), 56-72.

[15] Li, X., and Wang, Z. (2021). Deep Learning Techniques for SQL Injection Detection: Challenges and Opportunities. Journal of Artificial Intelligence Research, 40(2), 210-225.

[16] Peng Tang, Weidong Qiu, Zheng Huang, Huijuan Lian, Guozhen Liu, "Detection of SQL injection based on artificial neural network", Knowledge-Based Systems, Volume 190, 2020, https://doi.org/10.1016/j.knosys.2020.105528.

[17] Q. Li, F. Wang, J. Wang and W. Li, "LSTM-Based SQL Injection Detection Method for Intelligent Transportation System," in IEEE Transactions on Vehicular Technology, vol. 68, no. 5, pp. 4182-4191, May 2019, doi: 10.1109/TVT.2019.2893675.

[18] John Irungu, Steffi Graham, Anteneh Girma, and Thabet Kacem. 2023. Artificial Intelligence Techniques for SQL Injection Attack Detection. In Proceedings of the 2023 8th International Conference on Intelligent Information Technology (ICIIT '23). Association for Computing Machinery, New York, USA, 38–45. doi:10.1145/3591569.3591576

[19] Joshi Padma N, Dr. N. Ravishankar, Dr. M. B. Raju and N.Ch. Ravi. 2022 "Surgical Striking SQL Injection Attacks using LSTM", Vol. 13 No. 1 Jan-Feb 2022, doi: 10.21817/indjcse/2022/v13i1/221301182

[20] Sun, Hao, Yuejin Du, and Qi Li. 2023. "Deep Learning-Based Detection Technology for SQL Injection Research and Implementation" Applied Sciences 13, no. 16: 9466. doi:10.3390/app13169466

[21] Demilie, W.B., Deriba, F.G. Detection and prevention of SQLI attacks and developing compressive framework using machine learning and hybrid techniques. J Big Data 9, 124 (2022). doi:10.1186/s40537-022-00678-0