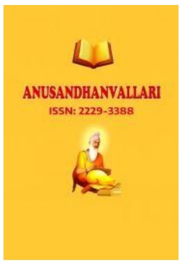

Anusandhanvallari
Vol 2024, No.1
December 2024
ISSN 2229-3388

Analyses of SQL injection Attacks and Their detection techniques: A Systematic Literature Review

[1]**Prof. Priyank Bhojak,** [2]**Dr. Vikram Agrawal,** [3]**Dr.Vatsal Shah**

[1]Research Scholar, [2]Research Supervisor, [3]Associate Professor

[1 &2 & 3]Gujarat Technological University, Ahmedabad - 380015, India

[1]priyankbhojak@gmail.com, [2]agrawal.vm@gmail.com

Abstract- SQL injection attacks are a common threat to web applications, exploiting weaknesses in database access mechanisms to allow unauthorized execution of SQL queries by attackers. SQL, a programming language for managing relational databases, enables users to interact with databases by adding, deleting, and updating data. Such attacks may be serious security risks to web applications, as they can lead to unauthorized access and retrieval of restricted data stored in databases [2]. This research paper a review on comparable analysis to find out more on different types of attacks, and their detection techniques.Various detection techniques, including machine learning algorithms and deep neural networks, have been proposed to combat SQL injection attacks. Deep neural networks, a subset of machine learning, are adept at understanding patterns and representations in extensive datasets, making them effective in understanding diverse data types like text, images, and sounds. However, one drawback of neural networks is their tendency to make overly confident predictions, especially when faced with unfamiliar data beyond their training scope. Researchers have introduced methods such as deep ensembles and Bayesian neural networks to estimate predictive uncertainty and mitigate overconfidence. Despite the potential of neural networks in various applications, it's essential to address their inclination towards overconfident predictions and potential drawbacks like overfitting. techniques like deep ensembles can enhance the reliability of neural network predictions by estimating predictive uncertainty effectively. Moreover, deep learning has demonstrated successful applications in the realm of Web security detection, showcasing its potential impact across a wide range of domains [4].

Keywords: SQL injection attack, web application, prevention, detection, Machine learning models

INTRODUCTION

By the use of a web page's input, attackers can inject SQL queries using the SQL Injection (SQLIAs) technique. Such SQL instructions have the ability to modify the database's layout and data. They might even retrieve critical information, destroying the database's security.

SQL Injections are typically regarded as website malware, although they can also target any kind of SQL database. The fundamental idea is to reach the backend by avoiding the service level. The underlying database of web applications is frequently accessible to attackers through attacks that target web apps. One such tool that evaluates a website for vulnerabilities is SQL Map.

Databases are frequently used as the back-end data store for web applications. Despite the fact that new web programming languages provide fresh approaches to programming more secure database features, many applications are still open to SQL injection attacks. Because of the nature of this attack unauthorized access to confidential information and its insertion or modification as well as the previously mentioned reasons, it is crucial to make web applications secure against such attacks. When a hacker attempts to access a database by inserting malicious inputs into the queries that alter the logic, syntax, or semantics of the legitimate query, this is known as a SQL injection attack.

It also offers a range of mitigation techniques and best practices for web developers to follow in order to prevent attacks. However, as the field continues to evolve, ongoing research will be needed to keep pace with emerging threats and new mitigation techniques.

2 LITERATURE REVIEW ON DIFFERENT DETECTION TECHNIQUES

This section explores a variety of ML and DL techniques found in the literature for the detection of SQL injection attacks.

Ketema [5] utilized a convolutional neural network (CNN) based on deep learning to construct a model aimed at preventing SQL injection (SQLI) attacks. The model was trained on a public benchmark dataset, employing various hyperparameter values and exploring five distinct scenarios. Ultimately, the model achieved an impressive accuracy of 97%.

On the other hand, Roy et al. [13] proposed a method for detecting SQL injection attacks through machine learning classifiers. Their study involved the utilization of 5 ML classifiers: logistic regression, Ada-Boost, naive Bayes, XG-Boost, and random forest. These classifiers were tasked with categorizing SQL queries into either legitimate or malicious categories. The model was trained and evaluated using a publicly accessible dataset of SQL injection attacks sourced from Kaggle. Notably, the study revealed that the naive Bayes classifier exhibited the most promising performance, achieving a commendable accuracy rate.

Both studies underscore the efficacy of employing machine learning and deep learning techniques in detecting and preventing SQL injection attacks, with each offering valuable insights into the optimization and implementation of such models in real-world scenarios.

Krishnan et al. [14] employed machine learning algorithms to address the challenge of detecting SQL injection attacks. They utilized classification algorithms to differentiate between SQL injection traffic and plain text. The machine learning classification algorithms explored in their study include the Naive Bayes Classifier, Passive Aggressive Classifier, Support Vector Machine (SVM), Convolutional Neural Network (CNN), and Logistic Regression. They propose that integrating runtime analysis, in addition to the static analysis method employed in their study, could enhance the effectiveness of the detection approach.

Mejia-Cabrera et al. [13] introduced a novel approach for constructing a NoSQL query database founded on JSON structure. Their study evaluated six classification algorithms for identifying injection attacks within this context: Support Vector Machine (SVM), Decision Tree, Random Forest, K-Nearest Neighbors (K-NN), Neural Network, and Multilayer Perceptron. Remarkably, the last two algorithms achieved an accuracy of 97.6%.

The research underscores the challenges inherent in developing methods to detect injection attacks on NoSQL databases, particularly due to the scarcity of example datasets. Despite these challenges, Mejia-Cabrera et al. demonstrated the feasibility of employing machine learning algorithms to detect injection attacks in NoSQL databases, thus highlighting the potential for further advancements in this area.

2. SQL INJECTION TYPES

There are numerous types and different versions of SQLI attacks. We discuss and categorize the primary SQLI attack types which are shown in figure 1

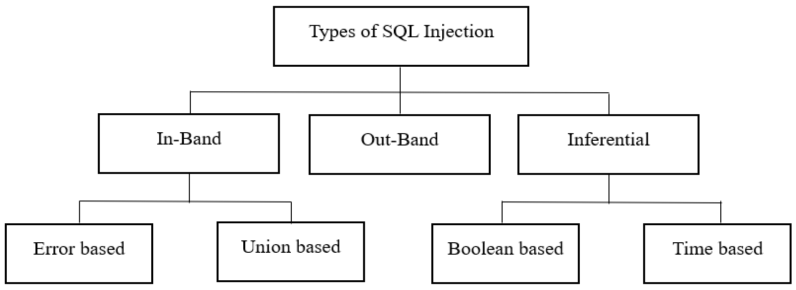

Figure 1

i) In-band SQL injection:

In-band SQL Injection attacks are the most frequent and straightforward to exploit. An in-band SQL injection allows the attacker to use the same communication channel to launch the attack and gather data.

The attacker can obtain the results directly by altering the original query. In the below example, personal data about the user is shown:

SELECT * FROM users WHERE userid LIKE 'currentuser'

By simply joining strings in the application, an attacker can supply the following current user:

%'—

Thus, we obtain the following search term:

SELECT * FROM users WHERE userid LIKE '%'--'

An SQL statement is completed by a single quote. The dash (-) that comes after the line denotes a comment. Thus, the application runs the following query:

SELECT * FROM users WHERE userid LIKE '%'

This attack will cause the entire user's table to be displayed, not just one user record. In-band There are two types of SQL injection: union-based and error-based SQLi.

In-band SQL injection can be divided into two types: error-based and union-based SQLi.

a) SQLi Error

SQL injection test method is known as the error-based uses database server error messages to determine the structure of the database. With error-based SQL injection, an attacker may occasionally be able to enumerate an entire database. Instead of storing errors in the log file, a live website should disable them or log them to a file with restricted access.For example,

SELECT * FROM users WHERE userid='currentuser'

Malicious hackers may provide the following Current user values:

1'

This leads to the following result:

SELECT * FROM users WHERE user_id = '1"

b) SQLi Union

In union-based SQL injection, the results of two or more SELECT statements are combined into a single result using the UNION SQL operator. For example,

SELECT * FROM users WHERE userid = ‘currentuser’

ii) Inferential SQL injection

Blind SQL injection is another name for inferential SQL injection. Inferential SQL injection may be harder for attackers to SQL query is considered to have been successfully executed.

Because no data is transferred between the web applications, the attacker cannot directly see the results of the injected queries in Inferential SQLi. Instead, these flaws are taken advantage of by watching how the application behaves in order to list the database. Boolean-based and Time-based Inferential SQLi are the two types.

a) Boolean-based SQLi

An additional term for content-based in the case of SQLi, the attacker sends a SQL query to the database, which the application interprets as either a true or false result based on the information the database returns.The content of the HTTP response may change based on the outcome. A malicious attacker can still figure out whether the payload used returned true or false even if the database doesn’t return any data. This is frequently a slow attack because the attacker would have to count all the characters in the database.

b) Time-based SQLi

When the database is paused for the required amount of time, then returned, a SQL query is considered to have been successfully executed. For example, the MySQL function SLEEP is available.If these functions are used in the query to slow down the response, attackers might try SQL injection. Similar to WAIT FOR DELAY, WAIT FOR TIME in SQL Server allows query execution to be paused and resumed when the system time reaches the specified parameter.

iii) Out-of-band SQL injection

Out-of-band SQL injections are uncommon because they require the database server for the web application to have certain features enabled. Out-of-band SQL injection is the name of the attack when it cannot be launched over the same channel as the data collection.In an out-of-band attack, the attacker tricks the victimized application into sending information to a remote endpoint under his command rather than waiting for a response.

3. SQL Injection Prevention

Two modules are used in the negative tainting approach: the attack database and the prevention module. The Prevention module functions as an additional layer. This layer filters the query before it is sent to the database server. This module examines the query that was sent from the application layer. If SQL injection is discovered, the query is blocked, and an alarm message is sent to the application program. The query is forwarded to the database server if there is no SQL injection. All known SQL injection attack symptoms are kept in a linked list structure in the attack database module. Symptoms are transformed into tokens, and tokens are transformed into integer values, which are then used to create a primary list. In a similar manner, the incoming query is used to create a secondary list. The secondary list and the primary list are contrasted. If a match is discovered, SQL injection is used. With the exception of stored procedures and character encoding, all known attacks can be stopped using this technique. Multithreading could be used in the future to speed up pattern matching.

To suggest a prevention method, static analysis, and runtime monitoring are combined. An approach based on models is used to identify malicious queries before they are executed. A static portion of the system constructs a model of valid queries, and the dynamic portion, using runtime monitoring to inspect dynamically generated queries, compares them to the model.

For the purpose of preventing authentication against SQL injection, the Preventing SQL Injection Attack in Web Application (PSIAW) technique is presented. A login table typically contains the columns username and password.

The username and password hash values are used in two additional columns in this. When a user logs in to a database, the username and password's hash values are computed. The user is permitted access to the data if they are equal. The main part of PSIAW is the SQL Query component, which is where the username and password hash values are calculated. The disadvantage of this method is that, aside from authentication, no other SQL injection attacks can be stopped by it.

The web application extracts the user's input from the generated query, and using a SQL proxy-based blocker technique, this data is then authenticated from the generated query's syntactic perspective. For this, a genetic algorithm is being used. This method does not necessarily require to discovering the application's source code or the authentication procedure.

An adaptive method and query tokenization are combined to create a multi-level prevention technique. An ordered series of malicious query tokens are used in the static analysis of application code. This database authenticates against the variations in incoming SQL query structure at runtime before sending the query to the database server for execution in order to catch all malicious SQL queries.

A method to identify and stop the SQL injection attack is developed by combining static and runtime analysis techniques. The execution of all incoming queries is tracked and observed using tracking techniques using a runtime analysis approach.

4. CLASSIFICATION OF TECHNIQUES FOR DETECTING SQL INJECTION

Dynamic Techniques: Dynamic detection methods employ machine learning or statistical models to classify queries as either malicious or benign. These techniques analyze the behavior and characteristics of SQL queries in real-time to determine whether they pose a threat. Dynamic techniques offer flexibility in detecting SQL injection attacks across various web codes and environments, enhancing prevention capabilities.

Static Techniques: Static detection methods rely on dictionary string matching approaches, often leveraging natural language processing (NLP) techniques. These methods involve breaking down the SQL query into its constituent parts and comparing them against a pre-existing dictionary of known malicious patterns. If a match is found, the query is flagged as potentially malicious. Static techniques are less flexible than dynamic methods and may have limitations in detecting complex or novel attack patterns.

The supervised learning approach aims to predict statement types and detect system attacks, categorizing them as malicious or non-malicious. In [9], a method is proposed where SQL statements generate query trees, and features from these trees are compared with a dataset using an SVM classifier. SVM and Fisher score aid in feature selection, while the Weka library handles vector training. Drawbacks include a 6% misclassification rate, marking users as malicious even with one query, and inadequate testing instances. In [10], HTTP URLs are scrutinized for SQL injection threats using predefined rules. A Naive Bayesian model distinguishes normal from malicious HTTP streams, with solutions to improve accuracy over time. However, this method lacks extensive testing against varied scenarios. The described methods exhibit several limitations like Misclassification Rate, False Positive

Issue , Inadequate Testing, Limited Validation against Large Datasets, Complex Feature Extraction and Poor Performance of LSTM Models. Around 6% of instances were inaccurately classified by the system, indicating a need for improved accuracy also Users were sometimes flagged as malicious even after executing only one query, which could lead to false positives and unnecessary intervention. The approach in [10] lacks extensive testing against large databases, which limits its reliability outside controlled environments. Complex Feature extraction in [11] involves considering multiple factors such as payload length and keyword occurrences, which may introduce complexity and computational overhead. The use of LSTM models resulted in poor recognition, high false positive rates, longer processing times, and lower accuracy, making it less suitable for practical applications.

Study [13] introduces a novel anomaly-based intrusion detection method for web-based applications, employing multiple models to characterize normal behavior accessing databases. It employs a probability-based model using Bayes' theorem for training data, detecting attacks with some false positives and significant CPU overhead.

In [14], four machine learning models including Support Vector Machine, Boosted Decision Tree, Artificial Neural Network, and Decision Tree are compared for predicting and preventing SQL injection attacks. They propose a framework integrating compiler platforms and ML to detect illegal and logically incorrect queries on server-side scripting, using 1,100 samples for training. Decision Jungle exhibits the best performance among the models evaluated.

Melody et al. propose a multi-stage log analysis architecture combining pattern matching and supervised machine learning [13]. Utilizing Kibana for pattern matching and Bayes Net for machine learning, they evaluate their system on 10,000 web application logs from the Log4j framework, achieving 95.4% accuracy in detecting SQL injection. However, Bayes Net's offline training phase renders real-time implementation unfeasible.

Graphical UI/UX for SQL Analysis:

Researchers in [16] developed a Graphical UI/UX to enhance user understanding during execution, employing SQL Parse tree validation & Tokenization. However, this method lacked code flexibility against malicious inputs, focusing mainly on select/where clauses for string checking, leaving other queries vulnerable.

Unsupervised Learning Approach for SQL Injection Detection: [17] The Presents a more sophisticated approach by normalizing queries, training HMMs, and clustering the tail end of queries to detect SQL injection attempts. Operating at the database firewall layer, this system defends multiple websites in shared hosting environments. Utilizing unsupervised learning offers more reliable and accurate results over time, enabling the system to continuously learn and adapt to new patterns and inputs. Techniques such as SVM, naïve Bayes, parse tree, tokenization, and multi-class classifiers are employed for classification.

In a study referenced as [18], a hybrid approach is proposed where the values of SQL query attributes extracted from web pages are matched with existing parameters. This method combines static and dynamic analysis by comparing statically written SQL queries with dynamically generated ones after excluding attribute values. The system's performance is comparable to a method introduced in [7], but with the key distinction of implementing an automatic process instead of a partially automatic one. It necessitates pre-analysis of web pages, both statically and dynamically, for effective operation.

5. COMPARISON OF SQL-INJECTION DETECTION TECHNIQUES WITH RESPECT TO ATTACK TYPES

Detection techniques detect attacks at runtime, with symbols √ for successful detection, × for not, and □ for partial detection due to limitations in the underlying approach. Table 1 shows a chart of schemes and their detection capabilities against SQL injection attacks, with √ for all types, × for those not, and □ for partial detections. Table

2 shows the percentage of SQL injection attacks addressed by detection techniques, with a formula for calculating tautology detection. The percentage of techniques that detect tautology is calculated by following formula.

$$\frac{\text{Number of techniques of identifies tautology}}{\text{Total number of techniques}} * 100 = 10/14 * 100 = 72$$

Table 2: Comparison of SQL injection detection techniques with respect to attack types

Attack-Types	**hniques which detect all attacks of that type(√)**	**chniques which detect the attacks only partially (□)**	**Techniques which is not able todetect attacks of that type(×)**
Illegalincorrect	66%	12%	14%
Piggybacked	64%	14%	23%
Tautologies	64%	14%	24%
Union	64%	14%	23%
Stored-Procedure	28%	14%	58%
Inferences	74%	14%	16%
Alternate-Encoding	56%	14%	30%

6. CONCLUSION & FUTURE WORK

SQL injection attacks are a very critical issue for web applications. It is hard to identify an effective solution to this issue. Many methods have been developed by different researchers to identify and counteract this vulnerability. So we can't identify any one tool or technologies to detect this kind of attacks . Future research aims to explore the use of a more intricate architecture to enhance its effectiveness.

Furthermore, we can do the relatively small size of the dataset used in this study and endorse expanding it, as well as implementing the models in real-world scenarios for future examinations and identifications. However, further research is needed to develop standardized datasets and evaluation metrics and to explore the full potential of machine learning for vulnerability analysis.

7. REFERENCES

[1] SQL Injection Attack Detection and Prevention Techniques Using Machine Learning, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 15, November 6 (2020).

[2] I. T. Holdings, "Modsecurity: Open-source web application firewall," http://www.modsecurity.org/, Accessed on April 2019.

[3] Ketema, A. Developing Sql Injection Prevention Model Using Deep Learning Technique. Ph.D. Thesis, St. Mary's University, London, UK, 2022. [Google Scholar]

[4] Wimukthi, Y.H.R.; Kottegoda, H.; Andaraweera, D.; Palihena, P. A Comprehensive Review of Methods for SQL Injection Attack Detection and Prevention. 2022. Available online:

https://www.researchgate.net/publication/364935556_A_comprehensive_review_of_methods_for_SQL_injection_attack_detection_and_prevention (accessed on 27 April 2023).

[5] Chen, D.; Yan, Q.; Wu, C.; Zhao, J. SQL Injection Attack Detection and Prevention Techniques Using Deep Learning. J. Phys. Conf. Ser. 2021, 1757, 012055. [Google Scholar] [CrossRef]

[6] Zhang, W.; Li, Y.; Li, X.; Shao, M.; Mi, Y.; Zhang, H.; Zhi, G. Deep Neural Network-Based SQL Injection Detection Method. Secur. Commun. Netw. 2022, 2022, 4836289. [Google Scholar] [CrossRef]

[7] Mishra, A.A.; Edelen, A.; Hanuka, A.; Mayes, C. Uncertainty quantification for deep learning in particle accelerator applications. Phys. Rev. Accel. Beams 2021, 24, 114601. [Google Scholar] [CrossRef]

[8] Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; Mané, D. Concrete problems in AI safety. arXiv 2016, arXiv:1606.06565. [Google Scholar]

[9] Lakshminarayanan, B.; Pritzel, A.; Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. Adv.Neural Inf. Process. Syst. 2017, 30, 6405–6416. [Google Scholar]

[10] Yu, W.; Kim, I.Y.; Mechefske, C. Analysis of different RNN autoencoder variants for time series classification and machine prognostics. Mech. Syst. Signal Process. 2021, 149, 107322. [Google Scholar] [CrossRef]

[11] Roy, P.; Kumar, R.; Rani, P. SQL Injection Attack Detection by Machine Learning Classifier. In Proceedings of the 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC), Salem, India, 9–11 May 2022; pp. 394–400. [Google Scholar]

[12] Krishnan, S.S.A.; Sabu, A.N.; Sajan, P.P.; Sreedeep, A.L. SQL injection detection using machine learning. Rev. Geintec-Gest. Inov. Tecnol. 2021, 11, 300–310.

[13] Mejia-Cabrera, H.I.; Paico-Chileno, D.; Valdera-Contreras, J.H.; Tuesta-Monteza, V.A.; Forero, M.G. Automatic Detection of Injection Attacks by Machine Learning in NoSQL Databases; Springer: Berlin/Heidelberg, Germany, 2021; pp. 23–3

[14]SQL Injection Attacks: Detection and Prevention Techniques, INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 8, ISSUE 01, JANUARY 2019.