iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** X **Month of publication:** October 2023

DOI: https://doi.org/10.22214/ijraset.2023.56190

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue X Oct 2023- Available at www.ijraset.com

SQLI Attack: An Approach using ML and Hybrid Techniques

Sneh Bavarva[1], Kalpesh Senva[2], Prof. Priyank Bhojak[3]
[1, 2, 3]*Dept. of Information Technology, Birla Vishvakarma Mahavidyalaya, Anand-388120, India*

Abstract: Web-based systems are significantly at risk from SQL Injection Attacks (SQLIA), particularly in industries that handle sensitive data, like finance and healthcare. During these attacks, hostile actors insert false SQL queries into the database server of a web application in an effort to steal sensitive data. The use of classifiers and techniques like end-to-end deep learning and expanding the Aho-Corasick algorithm to detect SQLIA attacks have been covered in the literature. These researches have shed light on identifying and minimizing SQLIA, but the problem still exists. To detect and prevent SQLIA, a thorough methodology is suggested that combines static and dynamic studies with machine learning and hybrid tactics. The study compares several machine learning algorithms with hybrid techniques, showcasing the hybrid strategy's higher performance in both training and test sets.

This method, which provides a practical means to secure web applications, is put out as a potential remedy for the persistent problem of SQLIA. In result, SQL Injection Attacks continue to be a danger, requiring effective ways for detection and prevention. It is highlighted as a promising solution the potential of machine learning and hybrid solutions, notably the hybrid approach. The study emphasizes how crucial it is to follow recommended procedures for protecting online applications from SQLIA attacks.

I. INTRODUCTION

SQLIA has been expanded to include Structured Query Language Injection Attack. The primary purpose of SQL is to interface with database data so that SQL can be used to alter the data. In order to steal sensitive information from the database server that is used by the web-based application, hackers perform fraudulent SQL queries. The majority of databases, including those in the banking, finance, healthcare, and employee records sectors, were frequently attacked via weak web-based applications. The many types of attacks are Command union SQL Injection, String SQL Injection, Numeric SQL Injection, Comments Attack, Blind SQL Injection, and Timing Attacks.

Malware assaults, especially SQLI attacks, are frequent in web systems with poor design. Although this vulnerability has been well-known for more than 20 years, it still causes concern [1]. SQL has long been the de facto industry standard for interacting with relational database management systems (DBMS).

Misbehavior caused by random mistakes or online attacks can significantly impede the development of cyber-physical systems because the bulk of their applications are safety-critical [2, 3]. Protecting cyber-physical systems from this form of attack is therefore essential.

The user enters inquiry specifications into specified input areas in three tire web applications. The application server in the middle tire creates SQL queries using these input values. Web mail, online retail sales, online auctions, online banking, and many other useful applications are examples of common web applications. Two categories of web applications exist: Presentation-oriented: In response to queries, a presentation-oriented web application creates interactive web pages using different markup languages (such as HTML, XML, and others) and with dynamic content.

Due to the widespread use of web browsers and the ease with which they can be used as clients, or "thin clients," web apps are quite popular.

A major factor in the popularity of web applications is the ease with which they can be updated and maintained without the need to distribute and install software on potentially thousands of client PCs. Web applications that are service-oriented put the endpoint of a web service into practice. Clients of service-oriented web applications are frequently presentation-oriented apps [4].

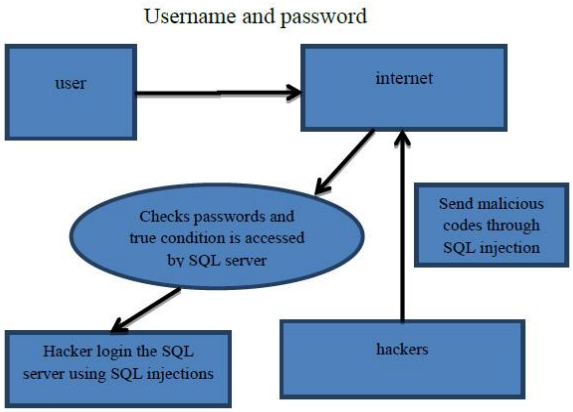


Figure 1: SQL Injection Attack [9]

Figure 1 presents overviews of SQL injection. where the user logs in to the database with a verified username and password. However, malicious queries will be injected by the intruders to access the database. After that, the query is compared to the threshold value (assuming the threshold is 50). If the string value is less than the threshold value, it permits the entry of the username, password, and other data into the database [5]. If the string value exceeds the threshold value, an alarm is generated and sent to the administrator. So that if the threshold value is higher, the condition employed by the hackers will always give the output true by SQL injection. Loss of data integrity and confidentiality results from this vulnerability. Although most detection and prevention measures contribute to the database's security over the web application, SQL injection is still possible thanks to specific security flaws. This research allows the research to identify the SQLIA by helping us to give a background of detection and prevention methods for future work.

II. LITERATURE REVIEW

To highlight the research gaps in the field, several published research works have been taken into consideration and incorporated in this section. Usually, credible databases' published studies are reviewed and included in the manuscript. Many academics have shown how to recognize SQLI attacks using DL, ML, and hybrid approaches.

The Aho-Corasick algorithm was extended by the author [6] in order to detect character injection introduced by a malicious attacker. They demonstrated how to accomplish this without materially expanding the Aho-Corasick algorithm's finite-state pattern matching machine. In order to reduce the likelihood of false positives, the system would only identify a match if the quantity of packed characters was within a predetermined range. The suggested algorithm's CPU time consumption is compared to that of the Aho-Corasick algorithm. It was discovered that the suggested method can surpass Aho-Corasick while dispensing with the inflated characters and spotting a legitimate match.

A systematic literature review of 36 articles on research into SQLI attacks and machine learning approaches was done by the authors in [2]. They have determined the most popular ML approaches, which they will utilize to categorize various types of SQLI assaults. They discovered that few studies used ML tools and approaches to create new SQLI attack datasets. Similar to this, their findings revealed that only a small number of research have concentrated solely on employing mutation operators to produce hostile SQLI attack queries. The researchers hoped to cover the application of more ML and DL methods to develop and identify SQLI attacks in their next work.

The author [7] suggested a cutting-edge online method for exact string matching and database filtering. A harmony theory network then checks the obtained strings with the query string to see if an exact match can be found after a self-organizing map first retrieves the cluster of database strings that are most similar to the query string. The similarity metric is set up to expose overall string similarity rather than character coincidence at homologous string places according to the specific properties of the database. The experimental findings show basically database-size independent, fault-tolerant, quick, and operation that is notably resistant to database alterations.

End-to-end DL for web attack detection was introduced in [10]. This work has produced three novel insights into the research of autonomous intrusion detection systems. First, they evaluated the viability of a technique for identifying online attacks based on the resilient software modeling tool (RSMT), which autonomously monitors and describes the runtime behavior of web applications.

In addition to describing how RSMT trains a stacked denoising autoencoder to encode and reconstruct the call graph for end-to-end DL, they also provided a low-dimensional representation of the raw features with unlabeled request data that was used to identify anomalies. A low-dimensional representation of the raw features with unlabeled request data is used to recognize anomalies by computing the reconstruction error of the request data, and they have also described how RSMT trains a stacked denoising autoencoder to encode and reconstruct the call graph for end-to-end DL. Thirdly, they looked at the results of empirically testing RSMT using fabricated datasets and purposefully vulnerable real-world applications. Last but not least, the results showed that the suggested approach could quickly and precisely identify attacks like SQLI, cross-site scripting, and deserialization with the least amount of labeled training data and domain expertise.

Analysis by the author [8] One attack that is vulnerable to desktop, web, and mobile applications is SQLIA. This survey article discusses several attack kinds, methods, and detection and protection measures. In order to combat SQLIA, this research compares various detection strategies and analyzes specific ways that should be strengthened. The Aho-Corasick pattern matching algorithm was used by the author to provide a certification and shrinking solution for SQL injection attacks. This technique makes advantage of string pattern matching to find and stop SQLIA. It can be expanded to change the pattern matching method to a two-step procedure and so give the database security against SQLIA.

Using MATLAB, 23 ML classifiers have been presented, evaluated, and compared for the detection of SQLI assaults [23]. They created their own datasets and inserted strange SQL syntax into them. The SQL statements were examined and manually verified. In sum the test classifiers were trained using a total of 616 SQL statements. They have applied ML methods such as "fine KNN, medium KNN, RUS, coarse KNN, bagged trees, linear SVM, boosted trees, linear, weighted KNN, cubic KNN, and subspace discriminant quadratic discriminant, simple tree, medium tree, subspace KNN, and cubic Fine Gaussian, SVM Logistic regression, SVM, cosine KNN, complicated trees, and coarse Gaussian Medium Gaussian, SVM, and SVM. The results showed that the ensemble boosted, bagged trees, linear discriminant, cubic SVM, and fine Gaussian SVM models were the five most accurate models. They put their suggested method to the test, and the findings indicated that it had a 93.8% accuracy in spotting the SQLI assault.

The author [9] suggested a technique using a proxy server and hash value that is used to protect the database from SQL injection attacks. To avoid SQLIA, a second two-phase authentication method will be employed. The research can be expanded in the future to include a procedure, which will save time and money. The author described a solution based on a defensive mechanism utilizing Kali Linux that prevents hackers from injecting harmful SQL code into the website without penetration testing. As a result, hackers cannot access the database, and sophisticated techniques are utilized to thwart SQL injection attacks. By consciously conducting penetration tests on all websites, the paper can be improved.

III. TYPES OF SQLI ATTACKS

When an attacker is able to change or insert malicious SQL statements into an application's input fields or parameters, the situation is known as SQL injection, or SQLi. Although the sophistication and intensity of SQL injection attacks might vary, they often fall into one of the following groups [11-15]:

1) Classic SQL Injection: Using the same channel that the application uses to connect with the database, an attacker can directly extract data from the database or modify it in an in-band SQLi attack. Error-based SQL injection: The attacker takes advantage of SQL errors the database generates to gather details about the data and database schema. Union-based SQLi: The attacker retrieves sensitive data by combining it with data from various database tables using the UNION SQL operator.
2) Blind SQL Injection: Blind SQL injection occurs when an attacker is unable to see the outcomes of their SQL injection in the responses sent by the application. Instead, they use true/false queries and the application's behavior to infer information.
3) Blind SQL Injection Based on Time: The attacker in this kind of attack exploits temporal delays to deduce details about the database. They submit SQL queries that, if a condition is true, make the program to delay its answer, allowing them to gather data.
4) Out-of-Band SQL Injection: Out-of-Band SQL injection happens when an attacker may access the database without using the application's response, such as through DNS or HTTP requests.
5) Second-Order SQL Injection: The malicious input is originally placed in a second-order SQL injection attack, which results in a SQL injection attack when the input is processed.
6) Time-Based Blind Second-Order SQL Injection: In this instance, the attacker manipulates a parameter that is stored in the database and then utilized in a query. This is a hybrid of second-order SQL injection and time-based blind SQL injection.

7) Boolean-Based Blind SQL Injection: The attacker creates SQL queries in this type of blind SQLi that use boolean (true/false) conditions to infer information.
8) Content-Based Blind SQL Injection: Attacks of this kind rely on replies that are based on content to infer information. The attacker creates SQL queries that, depending on the test circumstance, generate various application answers.
9) Time-Based Blind Second-Order SQL Injection: This combines time-based blind SQL injection with second-order SQL injection, where the attacker modifies a parameter that is saved in the database and then utilized in a query.
10) Inferential SQL Injection: Without actually seeing the outcomes, inferential SQLi entails retrieving data from the database by watching the actions and replies of the application. Techniques based on time, booleans, and content can all be used for this.

Security for web applications must focus on preventing SQL injection attacks. For identifying and mitigating vulnerabilities, best practices include the use of prepared statements or parameterized queries, input validation, escaping user inputs, web application firewalls (WAFs), and security testing tools.

IV. METHODOLOGY

A more thorough and theoretical understanding of SQLI attacks is necessary due to the nature of the attack and the necessity for detection and protection techniques. We looked into existing methodologies, as well as their attacking strategies and shortcomings, to develop our framework. As a result, we offer a thorough architecture that solves every vulnerability found in the earlier research. The attacker must first open his browser in order to perform the action, and if the application is already open, he must either enter his password there or ask permission to access the web service through the internet.

To proceed, the intrusion must first get past the firewall checker. After receiving user input through a variety of processes, such as user input validation, the web server uses that information to create queries for an underlying database [16-18]. Finding injection parameters, figuring out the kind and version of a web application's database, and figuring out the database structure can all help with this. The attacker will seek application server access once more if the request resulted in permission being granted. To determine whether or not the requested access involves SQLI, however, the present work proposed a model for detection and prevention in this instance.

Before classifying SQL queries, there were various steps. To determine whether the required queries are injected using either method, the static and dynamic analyses are compared for the initial feature extraction. The classifier accepts the query and compares it to the learned dataset. The ML classifier then accepts the extracted feature, training the model to recognize the supplied query. Classification algorithms are solved using the SVM [1], DT, NB [7], and other algorithms in ML approaches [5]. The trained model successfully completes all steps, including feature extraction and preprocessing.

As a result, using the provided trained ML model and hybrid technique, the classifiers will be taught to distinguish various forms of SQLI assaults during the feature extraction steps. The model matches each line query request pattern using the trained pre-fetched and trained dataset. The model will either deny the request or, in the case of a pure SQL query without any injection, transmit it to the application and database servers to carry out the desired operation, depending on whether the SQL query contains one or more qualified assaults. As a result, the current work propose creating a new architecture based on ML and hybrid strategies in order to cope with SQLI query attacks as effectively as feasible.

V. RESULTS

Three injection settings were used in this study in varied configurations. The first is via a user input field, allowing a web application to use HTTP (S) POST and GET to request data from a backend database, and the second is via cookies, which may be used to restore a client's state data when they return to a web application. If a web application uses the information in cookies to build SQL queries, an attacker can leverage this flaw to modify cookies and send them to the database server. Finally, by studying session usage data and identifying browsing patterns, a server variable can be produced. Logging these variables to a database without sanitization could lead to a SQLI vulnerability since attackers can tamper with the values in HTTP (S) and network headers by supplying malicious input into the application's client-end or by constructing their request to the server. As a result, the database stores attack log information for each and every attack received to the server. Additionally, attacks and regular data are separated into two types in attack log data.

We trained and evaluated vulnerability classifier models using several ML methods to find the most effective strategy. Traditional NB, DT, SVM, RF, LR, and neural networks are among the collection of algorithms. based on methods from our study, which include MLP and hybrid methods.

The Tensor Flow-Learn package was used to implement the traditional methods while the Keras library was used to implement the machine learning algorithms. Using ten-fold cross-validations, where the dataset was partitioned into ten separate divisions and the final accuracy result was recorded, we assessed the models' performance. Multiple classifiers may be produced during training and testing of the chosen methodologies; it is necessary to assess each classifier's performance using the proper evaluation metrics before choosing the best one. The following four examples can be obtained by combining the samples in accordance with the actual target category and the category predicted by the classification model: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). Tables 1 and 2 display the classification results' confusion matrix when each class is treated as a separate positive sample.

Table 1: Evaluation of the training set's performance

Approach	Precision	Recall	F1-score	Training set accuracy	Training time (Sec)
NB	88.20%	88.12%	87.98%	88.90%	8.99
ANN	99.15%	99.64%	99.20%	99.02%	20.21
RF	97.11%	96.98%	96.02%	94.90%	9.12
DT	92.82%	92.20%	91.22%	95.10%	54.55
SVM	97.10%	98.07%	97.44%	98.11%	18.66
Hybrid	99.43%	99.52%	99.38%	99.10%	27.54

Table 2: Evaluation of the test set's performance

Approach	Precision	Recall	F1-score	Test set accuracy	Training time (Sec)
NB	87.11%	87.22%	85.54%	88.11%	1.98
ANN	98.10%	98.45%	89.88%	98.80%	11.25
RF	93.30%	94.00%	92.10%	94.80%	6.12
DT	92.44%	89.88%	92.00%	94.85%	6.99
SVM	95.98%	95.23%	95.54%	98.10%	4.28
Hybrid	98.86%	99.11%	98.99%	99.20%	16.3

Precision, recall, f1-score, and training/test set accuracy are the evaluation criteria for classification models and are detailed in Equations (1-4).

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP+TN}{TP+FN} \quad (3)$$

$$F1-score = \frac{2\times(Precision\times Recall)}{Precision+Recall} \quad (4)$$

Tables 1 and 2 respectively provide descriptions of the outcomes of the approaches used throughout the training and testing stages.

The hybrid technique outperforms conventional ML approaches in terms of precision, recall, f1-score, and training set accuracy, according to the study given in Table 1. But for NB and RF, their training load is too heavy. As a result, the NB approach performs best in training time and worst in accuracy, precision, recall, f1-score, and training set evaluation metrics. The present work achieved the best performance in hybrid strategies out of all the implemented techniques for SQLI attack detection and prevention, as shown in Table 1.

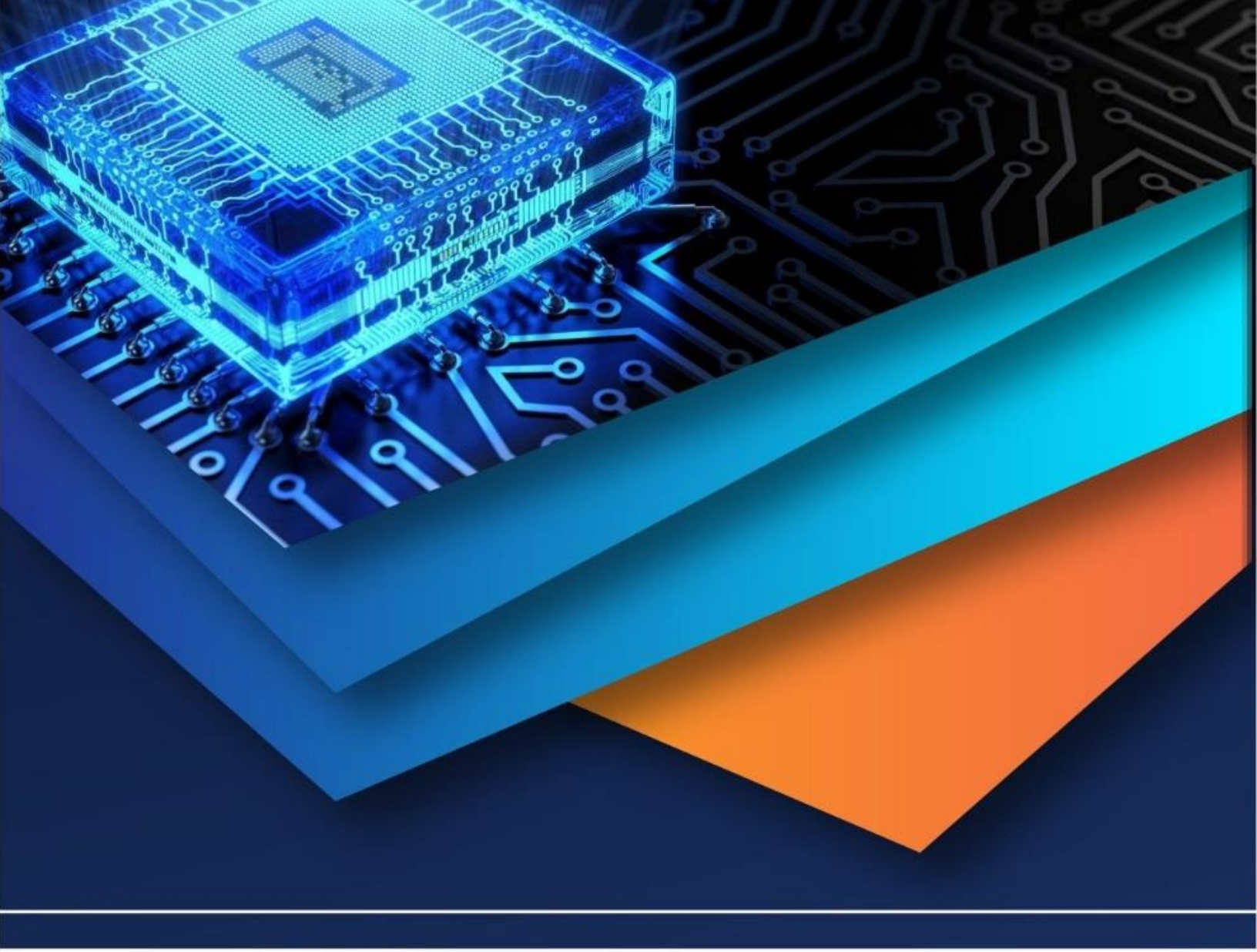
The hybrid strategy performs more accurately than other ML approaches in terms of precision, recall, f1-score, and test set, according to the study given in Table 2. However, their test duration is high. As a result, the NB approach performs best in training time and worst in accuracy, precision, recall, f1-score, and test set assessment measures. SVM and ANN are weak learners among the ML approaches used in this case. The current work achieved the best performance in hybrid strategies, as shown in Table 2, for the specified test sets, out of all the implemented techniques for SQLI attack detection and prevention.

VI. CONCLUSION

The essential issue of SQL Injection Attacks (SQLIA) and the requirement for efficient detection and prevention techniques in the context of web-based applications are clarified by this thorough discussion, which concludes. For more than 20 years, SQLIA has been a constant danger, resulting in data integrity and confidentiality breaches in a variety of industries, including finance, healthcare, and more. Due to distinct security weaknesses, SQLIA continues to be a difficult task despite the development of cybersecurity standards. To address this issue, numerous research projects and approaches have been investigated. Notably, hybrid and machine learning strategies have produced encouraging results in detecting SQLIA threats. The study covered the use of classifiers like SVM, DT, RF, and hybrid approaches to efficiently detect and prevent SQLIA. For both training and test sets, the hybrid strategy in particular showed superior performance in terms of precision, recall, f1-score, and accuracy, however, their test duration is high. The study also emphasized the significance of using best practices like prepared statements, input validation, and web application firewalls to protect web applications from SQLIA. The suggested architecture, which combines machine learning and hybrid tactics, presents a possible answer to the enduring issue of SQLIA.

REFERENCES

[1] TehFaradilla Abdul Rahman ,AlyaGeogianaBuja, KamarularifinAbd. Jalil, FakariahMohd Al," SQL Injection Attack Scanner Using Boyer-Moore String Matching Algorithm" Journal of Computers, Volume 12, Number 2, March 2017
[2] Frantisek Franek, Christopher G. Jenningsb,W.F.Smyth" A simple fast hybrid pattern-matching algorithm" Journal of Discrete Algorithms (2017), pp. 135–144
[3] B. Jakub, P. Buciak, and P. Sapiecha. "Building dependable intrusion prevention systems." Dependability of Computer Systems, International Conference on. IEEE, 2016.
[4] SudhaSenthilkumarand Krishna TejaReddy,"Preventing SQL Injection Attack Using Pattern Matching, Parse Tree Validation and Cryptography Algorithm",Journal of Environmental Science, Computer Science and Engineering & Technology" Vol.6. No.4, 246-253,Nov-2017
[5] Das D, Sharma U, Bhattacharyya DK. Defeating SQL injection attack in authentication security: an experimental study. Int J Inf Secur. 2019;18:1–22.
[6] Kasim O. An ensemble classification-based approach to detect attack level of SQL injections. J Inf Secur Appl. 2021.
[7] Tang P, Qiu W, Huang Z, Lian H, Liu G. Detection of SQL injection based on artificial neural network. Knowl-BasedmSyst. 2020.
[8] Erdődi L, Sommervoll AA, Zennaro FM. Simulating SQL injection vulnerability exploitation using Q-learning reinforcement learning agents. J Inf Secur Appl. 2021.
[9] Kar D, Panigrahi S, Sundararajan S. SQLiGoT: detecting SQL injection attacks using the graph of tokens and SVM. 2016. p. 206–225.
[10] Uwagbole SO, Buchanan WJ, Fan L. Applied machine learning predictive analytics to SQL injection attack detection and prevention. 2017.
[11] McWhirter PR, Kifayat K, Shi Q, Askwith B. SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel. J Inf Secur Appl. 2018;40:199–216.
[12] Mejia-Cabrera HI, Paico-Chileno D, Valdera-Contreras JH, Tuesta-Monteza VA, Forero MG. Automatic detection of injection attacks by machine learning in NoSQL databases. 2021. p. 23–32.
[13] Pathak RK, Mohit, Yadav V. Handling SQL injection attack using progressive neural network. 2020.
[14] Wang Y, Li Z. SQL injection detection via program tracing. IDCS 2012, LNCS 7646. 2012. p. 264–265
[15] Zhang H, Zhao B, Yuan H, Zhao J, Yan X, Li F. SQL injection detection based on deep belief network. 2019. p. 1–6.
[16] Priyaa BD, Devi MI. Hybrid SQL injection detection system. 2016.
[17] Joshi A, Geetha V. SQL Injection detection using machine learning. 2014.
[18] Deriba FG, Salau AO, Mohammed SH, Kassa TM, Demilie WB. Development of a compressive framework using machine learning approaches for SQL injection attacks. Przegląd Elektrotechniczny. 2022;1(7):181–7.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY