The ISC Int'l Journal of Information Security
ISeCure

November 2021, Volume 13, Number 3 (pp. 1–9)
http://www.isecure-journal.org

Selected Paper at the ICCMIT'21 in Athens, Greece

A Review Study on SQL Injection Attacks, Prevention, and Detection [**]

Mona Alsalamah [1], Huda Alwabli [1], Hutaf Alqwifli [1], and Dina M. Ibrahim [1,2,*]
[1] *Department of Information Technology, College of Computer, Qassim University, Buraydah, Saudi Arabia.*
[2] *Computers and Control Engineering Department, Faculty of Engineering, Tanta University, Tanta, Egypt.*

A R T I C L E I N F O.

Keywords:
Cyber Security, Web-Based System, SQL Injection

Type: Research Article

doi: 10.22042/ISECURE.2021.0.0.0

dor: 20.1001.1.20082045.2021.13.3.1.8

A B S T R A C T

The functionality of a web-based system can be affected by many threats. In fact, web-based systems provide several services built on databases. This makes them prone to Structured Query Language (SQL) injection attacks. For that reason, many research efforts have been made to deal with such attacks. The majority of the protection techniques adopt a defense strategy which results to provide, in extreme response time, a lot of positive rates. Indeed, attacks by injecting SQL are always a serious challenge for the web-based system. This kind of attack is still attractive to hackers and it is in growing progress. For that reason, many researches have been proposed to deal with this issue. The proposed techniques are essentially based on a statistical or dynamic approach or using machine learning or even deep learning. This paper discusses and reviews the existing techniques used to detect and prevent SQL injection attacks. In addition, it outlines challenges, open issues, and future trends of solutions in this context.

1 Introduction

With the exponential growth of web advancements, the Web application became one of the most mainstream contact streams. In data frameworks, knowledge is a big challenge. To get input from users, many associations run their transactions on database-appended web apps. The Web application is an important source of knowledge for any company to access simple business process information and to be commonly used in numerous applications. With the ubiquity of web apps, the web environment has various security challenges and web application vulnerabilities are growing. The weakness in the approval of details is where client feedback is included in the product without affirming its authenticity.

The Standardized Query Language, which is the basic programming language for social database development, is SQL. It is a language of order and regulation used to construct, modify, delete, and retrieve knowledge and frameworks that provide the framework of the social database. In basic terms, SQL INJECTION is the method of transferring SQL code into interactive web applications. User information

[*] Corresponding author.
[**]The ICCMIT'21 program committee effort is highly acknowledged for reviewing this paper.

Email addresses: 421214861@qu.edu.sa, 421200396@qu.edu.sa, 421200326@qu.edu.sa, d.hussein@qu.edu.sa, dina.mahmoud@f-eng.tanta.edu.eg

ISSN: 2008-2045 © 2020 ISC. All rights reserved.

such as form is acknowledged and then this input is used in database queries, but in a manner that was not intended. The key purpose is to trick the database into running malicious code because of the application's bad design.

To include useful information, most websites provide a series of fields for communicating with users. To put the data within the database, the collection of fields serves as carriers. Search fields, username and password fields, credit card data fields, etc., for instance. Successful Structured Query Language Injection attack (SQLIA) leads to an unwanted view or alteration of the database's private data: the entire database crashes and, in certain instances, the vicious attacker gets the database's administrative privileges. SQLIA is typically divided into two key groups, namely First-Order SQLIA and Second-Order SQLIA. The Second-Order SQLIA technique is a complicated process. The Open Network Application Protection Project has named SQLIA as the most serious security threat for 2017 (OWASP) [1].

The malicious person who wishes to insert malicious data must implement the file in the SQL programming language. During the attack, malicious data that is harmfully coded is introduced into the database. It is also better to prevent an intrusion at an early stage to minimize data loss. Web-based applications on a network are protected by Intrusion Prevention Mechanisms, Firewalls, or Stable Socket Layers. However, because of HTTP's complex request/response mechanism, protection becomes minimal or non-existent shown in Table 1.

Our paper consists of the following sections: Section 2 presents the background of the SQL injection attack. In Section 3, types of SQL injection attacks are illustrated. While in Section 4 the causes of SQL injection in the database are explained. Section 5 demonstrates the related work of existing techniques to detect and prevent SQL injection attacks from multiple studies. In Section 6, we discuss open issues and future trends for preventing and detecting SQL injection attacks. Finally, the conclusions of this paper are drawn in Section 7.

2 Background of SQL Injection Attack

As HTTP queries, a web application accepts input and produces SQL statements as output. The SQL statements created are sent to the database of the web application to collect or store or change the values according to the user's purpose. Here, HTTP requests are specifically interested in SQL statement building. The SQL Injection attack occurs when HTTP requests are not approved. The malicious user will manage unsecured HTTP requests to form a new statement that causes the user harm. The source procedure, for example, uses an HTTP request to create a SQL statement [2]. *Salary= "Select*from empl_details where empl_id="*

As the above code is given as an entry to the web server, it uses an HTTP request to produce the output. The HTTP request is not checked and is used for the creation of the SQL statement directly. Here, to launch an attack, the vicious user will take advantage. The following is an illustration of the altered version of the vicious user's original HTTP submission. Rather than submitting real input *"employeeid = 200"*, the vicious user appends a string *"update empl_details set emplsalary = emplsalary * 2"* to the input. From a malicious HTTP message, the SQL statement generates is *POST/employee.jsp* HTTP/1.1 *"Select * from empl_details where empl_id = 200; update empl_details set emplsalary = emplsalary * 2;"*.The database is exploited by this argument and doubles the salary of an employee [2].

Other types of attacks can also be initiated by the vicious user, such as adding an intruder account to the database, misleading the signature identification system, etc. The vicious consumer concatenates the real input with the following example a string *"CREATE USER admin IDENTIFIED BY passadmin"* which creates a new account "admin" with password "passadmin" and the SQL statement generated from the malicious HTTP request is *"Select * from empl_details where empl_id = 200; CREATE USER admin IDENTIFIED BY passadmin;"*. For the new user "admin", Different functions may be allocated by the malicious declaration "GRANT CONNECT TO admin". The statement created from the string recently attached is *POST/employee.jsp* HTTP/1.1 *"Select * from empl_details where empl_id= 200; CREATE USER admin IDENTIFIED BY passadmin; GRANT CONNECT TO admin;"* The vicious user gets power over the database to carry out different manipulations after executing the sentence.

3 Types of SQL Injection Attacks

This section will investigate the most important types of SQL attacks Figure 1 show the display of the types:

3.1 Error-Based SQL Injection

The most popular form of SQL injection flaw is error-based SQL injection. It is based, through the user interface, on unintended commands or invalid input. As a result, the database server responds with an error that may include target information such as structure, version, operating system, or returns the complete results of the query [3].

Table 1. Comparison of top 10 risk between 2013 and 2017

Top 10 (2013)	Change	Top 10 (2017)
A1: Injection	None	A1: Injection
A2: Broken Authentication and Session Management	None	A2:Broken Authentication
A3: Cross-Site Scripting (XSS)	Decrease	A3: Sensitive Data Exposure
A4: Insecure Direct Object References	[Merged+A7]	A4: XML External Entities (XXE)[NEW]
A5: Security Misconfiguration	Decrease	A5: Broken Access Control [A4+A7 Merged]
A6: Sensitive Data Exposure	Increase	A6: Security Misconfiguration
A7: Missing Function Level Access Control	[Merged+A4]	A7: Cross-Site Scripting (XSS)
A8: Cross-Site Request Forgery (CSRF)	Deleted	A8:Insecure Deserialization[NEW]
A9: Using Components with Known Vulnerabilities	None	A9:Using Components with Known Vulnerabilities
A10: Unvalidated Redirects and Forwards	Deleted	A10:Insufficient Logging& Monitoring [NEW]

3.2 Boolean-Based Blind SQL Injection

A Boolean query in this form of SQL injection attack allows the program to provide a different answer to a true or invalid database result. It operates by enumerating the characters that must be removed from the text. The reply displays whether or not the user ID is present in the database [4].

3.3 Time-Based Blind SQL Injection

SQL queries are sent to the database in this form of SQL injection attack, causing it to wait for the specified period represented in seconds before reacting. From the answer time, the attacker will know if the outcome of the query is true or false.

3.4 Union-Based SQL Injection

Results returned by the initial question shall be expanded by the operator of the Union. In this way, where the form is the same as the original, users are allowed to run two or more sentences. Let's evaluate this example for this purpose [3]: *Example: SELECT first_name, last_name FROM users UNION SELECT Username, Password FROM login* [3]. In this example, the SELECT command is used and the following conditions should be fulfilled in order exploit to work:

- Each SELECT statement within the union has the same number of columns.
- The columns must also have similar data types.
- The columns in each SELECT statement are in the same order.

In this example names of the columns in the table users are *first_name* and *last_name*. The names of the columns in the table login are username and password. The query is successful when it has the correct number of columns.

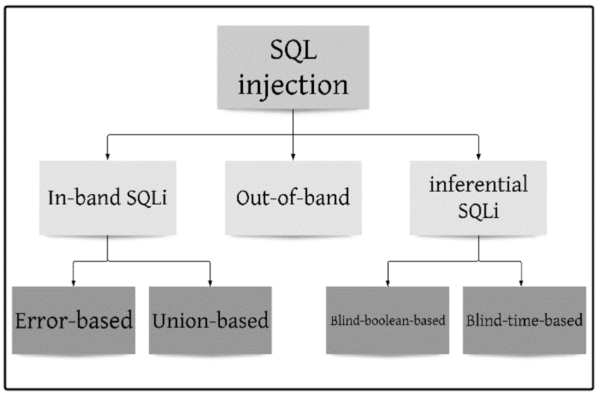

Figure 1. Type of SQL injection [1]

3.5 Out-Of-Band SQL Injection

This is a less popular form of SQL injection attack, owing to the fact that it relies on database server features being allowed. This form of attack happens when the attacker is unable to carry out both the attack and the data collection on the same channel.

3.6 Blind SQL Injection

A blind attack is a type of SQL injection in which no error message is displayed. As a result, exploiting it is more difficult because details are returned when SQL payloads are provided to the application.

4 Causes of SQL Injection in Database

SQL injection flaws impact not only the data protection of individual websites but also the database system as a whole, as well as the network system that hosts similar applications. In extreme cases, it can cause large areas of web pages to hang, viruses to spread, privacy leaks, remote server control, and network paralysis. Furthermore, SQL injection flaws are often used as a starting point for network penetration attacks, which strike the target network in stages. Hackers aren't happy with only manipulating an application or a server; they want to take over the entire network, including access control, to gain access to the network's internal sensitive information [5].

Communication between the user and the web application is important in web applications. The consumer must send data to the server as part of the interaction phase. After receiving data from the user, the web application queries the database system to conditionally show the data to the client. Users usually send data to the server using the GET, POST, and Cookie methods have shown in Table 2. The GET approach involves explicitly writing the data to be sent in the URL, and it is often used to send less data when browsing. The POST approach is commonly used as a form submission and is often used when users need to submit data, which is typically a large amount of data. Cookies are small pieces of data that a web application saves in the user's browser buffer. They're typically used to store user identities or monitor their surfing habits [5].

Most Web apps stored the data in SQL databases. Almost every Web application has a SQL database running in the background. SQL syntax, like most other languages, allows database commands to be combined with customer data. If developers aren't careful, user data can be interpreted as commands, allowing remote users to do more than just input data into Web applications; they can even run arbitrary commands on the database.

5 Related Work

In this section, we will present and discuss some of the related work and many techniques used to prevent and detect SQL. Authors in [6] offer a simple and powerful artificial neural network-based technique for SQL injection detection, which comprises three parts: data preparation, feature extraction, and model training. First, 8 types of specialized features are eliminated from a large volume of SQL injection data, and then a large amount of real data is utilized to train the neural network model. Finally, the results of model training are compared. In this paper, several neural network models are employed for training. MLP uses an artificial feature extraction procedure to extract the resulting features from the URL as the model input. Long short-term memory (LSTM) is frequently employed in comparative tests. The URL is explicitly transformed into a vector by LSTM as the model input. The results of the experiments imply that Multi-Layer Perceptron (MLP). The identifying effect of MLP is substantially greater than that of LSTM.

The authors of [7] have proposed a method for detecting and avoiding these attacks that employ the Knuth-Morris-Pratt (KMP) pattern matching algorithm. To detect any malicious code, the algorithm was used to align the user's input string with the injection string's saved template. The application was created using the PHP scripting language and the Apache XAMPP Server. The level of security of the methodology was calculated using several test cases of SQL injection, cross-site scripting (XSS), and embedded injection assaults. The results revealed that the proposed solution could effectively detect and avoid attacks, record the attack in the database, block the device using the mac address, and generate an alert message. As a result, the suggested approach has been more successful in detecting and avoiding SQL injection.

The authors of the study described in [5] offer a component-based strategy to reduce the occurrence of wrong effects and make it easier to enhance the proposed solution. The accuracy of the suggested technique is determined using three different types of software. An observational assessment is undertaken on three unsafe custom websites to test the feasibility of the planned research. The experiment findings yielded significant results in terms of high precision. On the other hand, the suggested approach offers higher capabilities for evaluating page replies based on four alternative strategies. Furthermore, even though modest restrictions are put on the supplied data, the proposed approach is the only one that performs SQL stored procedure assaults and circumvents login authentication. The only suggested solution will effectively finish the SQL injection attacks stored procedure and conduct vulnerability prediction by analyzing the page's internal structure, according to the analysis of the results. Even though the proposal is capable of sending a second-order SQL injection attack on a variety of injection parameters regularly, this attack does not result in a vulnerability.

The authors of study [8] presented a SQL injection vulnerability scanner based on black-box testing, with four components: crawling, attacking, analysis, and reporting, each with subcomponents. Although there are many SQL injection vulnerability scanners available, the majority of them need to improve their effectiveness because they have issues such as false alarms. The suggested SQL injection vulnerability scanner intends to disclose the greatest number of vulnerabilities with the fewest false alarms. The study's strength is that it includes an algorithm for the proposed SQL injection vulnerability scanner, as well as its implementation in JavaScript and testing on three different target applications. The research concludes with a comparison of the suggested scanner

Table 2. Features of GET and POST methods

Delivery method	Features
GET	Less data is passed when data is explicitly written in the URL.
POST	A huge amount of data must be transmitted, and it must be submitted using Form forms or the AJAX POST process.
Cookie	The data is huge and sensitive, and it is stored in the user's browser cache.

to existing scanners, with the findings examined to demonstrate its efficacy. Combining machine learning with SQL Injection Attack Detection & Prevention, as the name says. They created a web application that exhibits large volumes of learning data utilizing dictionary words to train a classifier using Applied Machine Learning Predictive Analytics with SQLIA Detection and Prevention. The trained classifier is offered as a web API for use in a.NET application to intercept SQLIA. Unauthorized access to the database is thus prevented.

They present a methodology for detecting SQL injection attacks based on two methodologies, namely the fingerprinting method and Pattern Matching, in [9] this paper, intending to distinguish good SQL queries from dangerous queries. It considers the decrease in processing time for Pattern Matching engines as well as the decrease in false-positive rate. This framework's way of operation can be characterized as the speedy initial detection and judgment of incoming SQL queries, as well as for deciding whether they require an accurate match. As illustrated in Figure 2, the Rabin fingerprint method is utilized for first rapid detection and judgment on SQL queries, and the Aho-Corasick pattern matching algorithm is used if the query is recognized as illegitimate. This framework tries to speed up the process of detecting malicious queries by employing fingerprints before Pattern Matching.

In [10], the researchers proposed a new approach that comprises two steps: Tokenizing the query entered by the user is the initial stage in this procedure. In the tokenization phase, white space, double dashes (–), sharp sign (#), and all strings before each symbol are all tokens. In the second stage, each string token was compared against the contents of a specified lexicon once the query was tokenized. The majority of reserved terms (commands), as well as several logical operators, are found in the lexicon. The contents of the lexicon are collected in the majority of injected commands or sentences in SQL injection attacks. Table 3 lists some of the words in the lexicon. It contains a total of 20 words. When the input question statement is entered, whether or not to detect injection. The validity of the input data is tested during execution by comparing it to the contents of the lexicon. If they are matched to other terms, an attempt at

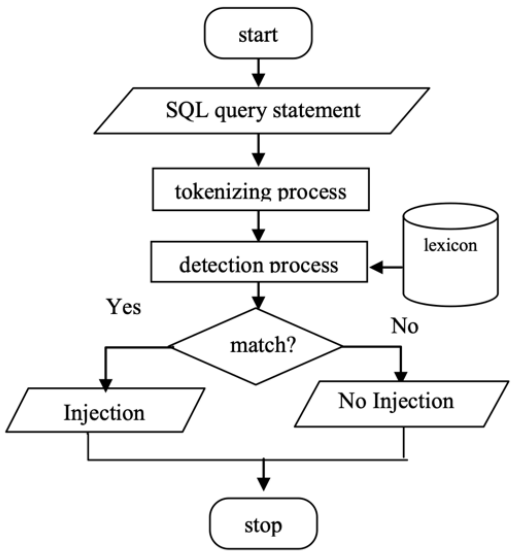

Figure 2. The proposed approach methodology for detecting SQL injection attacks based on two methodologies, namely the fingerprinting method and pattern matching [9]

Table 3. Top 10 OWASP IoT vulnerabilities in 2018

No.	Vulnerability
1st	Weak, easy to predict, or embedded passwords
2nd	Insecure communications services
3rd	Insecure ecosystem interface
4th	Lack of a secure mechanism for software updates
5th	Use of insecure or compromised software components
6th	Inadequate privacy protection
7th	Insecure data transfer and storage
8th	Lack of device management such as support
9th	Insecure standard settings
10th	Inadequate physical hardening*

SQL injection is made. There will be no injection if the answer is no. The proposed technique is depicted in Figure 2.

Using Removing the Parameter Values of SQL Query, the authors in [11] develop a SQL detection and prevention approach. The purpose of this strategy is to provide a simple and effective method for keeping the database secure. Figure 3 depicts the process in action. It employs both static and dynamic analytic methods. This method isn't altogether new;

rather, it's a refinement of one that already exists.

[12] is an example. This article investigates the GreenSQL database firewall's operating mode and process. GreenSQL learning's input model is built by constructing a patterned input and configurable whitelist based on a study of the characteristics and patterns of SQLA instructions, as shown in Figure 4. GreenSQL's learning efficiency will be reduced, and samples will be intercepted in IPS mode, allowing the context database's security to be properly maintained. Modular attack input is realized based on thoroughly investigating the features of each working mode of the GreenSQL database, blended with the common attack methods of SQL, by developing an abstract set of seven sorts of attacks based on the rule combination and guidance of these seven sets.

In addition, the authors recommend using query tokenization to detect and block SQL attack injection in [13]. The proposed technique is broken down into six steps. The user enters the settings in the first stage. The tokenization technique, which is based on dividing a string into multiple parts, is utilized in stage 2. In stage 3, two different tokenization strategies are used. The two results from the previous stage are compared in stage 4. Stage 5 determines whether or not the user has the authorization to access the database based on the prior comparison. Finally, the injection attack is prevented in stage 6. According to one experiment, query tokenization can identify or even prevent the majority of frequent SQL injection attacks. Instead of breaking prepared statements and query tokenization into two separate processes, the proposed work can be improved by integrating them into one process.

The authors of [14] recommend that to cope with the challenge of identifying and blocking SQL attacks, they should apply machine learning and the Support Vector Machine (SVM). This work, like most others, is based on a dataset. After that, you'll need to train SVM on this dataset and then run the test. The results indicate good detection performance, which is empirically verified in a huge data context. To deal with multiple types of SQL injection attacks, this work can be improved by using a multiclass classifier. This aids in defining the type of assault that has occurred, rather than simply alerting whether or not an attack has occurred. Similarly, in [15], a SQL injection detection system is proposed as a framework integrating a natural language processing model with a deep learning technique. The proposed architecture improves accuracy while lowering the rate of false alarms. It also allows the machine to learn the SQL injection attacks language model automatically. This option is critical for reducing human intervention while also providing an appropriate level of defense against zero-day assaults. The studies are illustrating some strategies to prevent and detect SQL injection attacks, as shown in Table 4. below, which previews the aspects covered in the linked study.

6 Open Issues and Future Trends for Preventing and Detecting SQL Injection Attacks

Through surveying the most important work in the context of SQL injection attacks, we found that this problem is stills serious and can be harmful to websites. The prosed solution is still enough and needs more research efforts. The existing research work dealing with the problem of SQL injection attacks can be divided into four categories which are [17]:

- Based on the static analysis: which is based on analyzing the traffic to detect this attack.
- Based on dynamic analysis: which is the extension of the previous type by considering dynamic traffic to prevent and detect SQL injection attacks as soon as they happen.
- Based on machine learning: this kind is trying to solve the limitation of static as well as dynamic analysis.
- Based on deep learning: this kind of detection attack is used since machine learning selection is done manually which leads to ignoring important features. Consequently, adopting deep learning helps to improve notably the previous three types.

The results of the three types of detection are still not enough and need more investigation and improvement. The future trends of SQL injection attack prevention and detection will concentrate more on:

- Creating and defining dataset close to the reality to get a more realistic result.
- Using computing power by introducing many barriers to detect this attack in the advance stage.
- Measuring the performance of models to improve their efficiency.

These trends are facing many challenges constituting the weaknesses of the existing methods which are [17]:

- The false positive and negative rate: linked to the detection of attack which has to be decreased as much as possible.
- The huge amount of training data: which is an interesting issue in this case that can be tackled by adopting big data techniques.
- The defeat of partial attack and vulnerabilities:

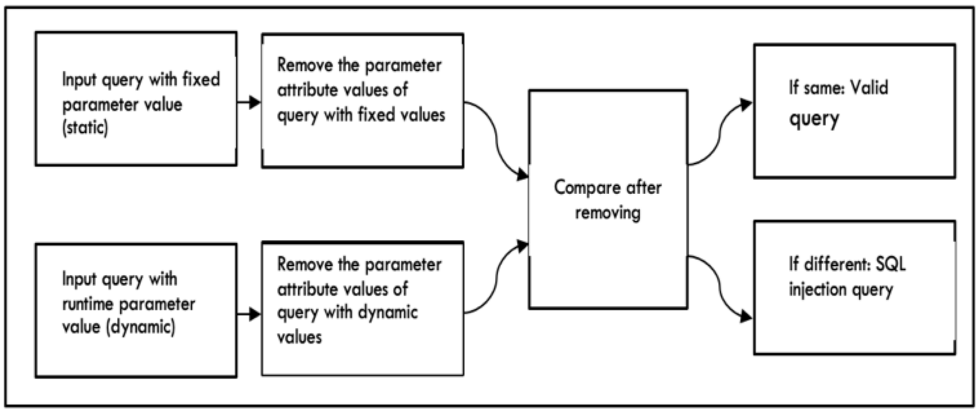
Input query with fixed parameter value (static)
Remove the parameter attribute values of query with fixed values
Input query with runtime parameter value (dynamic)
Remove the parameter attribute values of query with dynamic values
Compare after removing
If same: Valid query
If different: SQL injection query

Figure 3. The framework of the proposed methodology presented by [11]

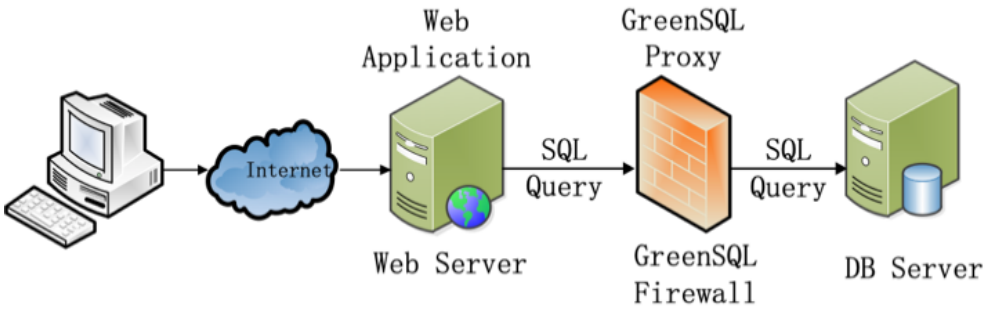

Figure 4. The working mode of the GreenSQL database

in fact, most existing solutions can not cover all types of attacks. On the contrary, SQL injection is including a big set of attacks type and it is growing rapidly.

- The weakness of practical behavior: the most existing solution is practicableness and needs to adopt more scenarios to be more realistic.

Consequently, we think that defeating SQL injection attacks needs to combine detection as well as prevention. The prevention can avoid losses and keep the web-based system safe. This goal can be done by combining a set of prevention and detection methods that can effectively block most attacks. Furthermore, we think that reinforcement learning can be a very promising solution to this problem. Reinforcement learning can combine more than one technique to provide the best detection result to avoid web-based system damage.

7 Conclusion

One of the most common threats to web applications is SQLIA. To provide security and integrity, web applications must protect their databases from a variety of threats. In SQLIA, attackers can assault the application with a designed query statement via a web input form, steal identities, gain access to confidential information, and tamper with available data, all of which can have catastrophic consequences. This paper is looking deeper into the background of SQLIA and its various types. In addition, the literature review covers a lot of researchers' proposed methods as well as their findings. Also, the paper discusses in detail some open issues and future trends of solutions in this context.

References

[1] Kyriakos Kritikos, Kostas Magoutis, Manos Papoutsakis, and Sotiris Ioannidis. A survey on vulnerability assessment tools and databases for cloud-based web applications. *Array*, 3:100011, 2019.

[2] BH HemaMalini, L Suresh, and Mayank Kushal. Comprehensive analysis of students' performance by applying machine learning techniques. In *Smart Intelligent Computing and Applications*, pages 547–556. Springer, 2020.

[3] Igor Tasevski and Kire Jakimoski. Overview of sql injection defense mechanisms. In *2020 28th Telecommunications Forum (TELFOR)*, pages 1–4. IEEE, 2020.

[4] Solomon Ogbomon Uwagbole, William J

Table 4. Features of GET and POST methods

Ref	Description	Technique	Result
[6]	The authors proposed an approach for SQL injection detection based on artificial neural network	Artificial neural network	Experimental findings suggest that other neural network models are superior to the extracting features.
[7]	The authors are introducing a method that uses the Knuth-Morris-Pratt (KMP) pattern-matching algorithm to identify and avoid these attacks	Knuth-Morris-Pratt (KMP)	an approach was able to efficiently identify and deter attacks, record the attack entered in the database, block the device using the mac address, and also create a warning code.
[16]	Paper proposed a component-based approach to mitigate the occurrence of incorrect effects, as well as to make it easier to improve the solution suggested.	Component-based approach	The result analysis reveals that the suggested solution will effectively complete the SQLA stored procedure and execute vulnerability by evaluating the page's internal structure
[8]	This study proposed an SQLA scanner by using black-box testing aims to report the highest number of vulnerabilities with the least amount of false alarms.	Black-box testing	Provides an algorithm for the proposed SQL injection vulnerability scanner and its implementation using JavaScript, and experience it on three target applications.
[4]	Suggests, combining machine learning with SQL Injection Attack Detection and Prevention.	Applied Machine Learning Predictive Analytics	As a web service consumed in a dot NET application, the trained classifier is published to intercept SQLIA.
[9]	Paper proposed a framework for detection of SQLA based on two techniques: fingerprinting method and Pattern Matching	Fingerprinting method and Pattern Matching	The framework works to speed up the process of detecting malicious queries.
[10]	Paper proposed an approach that detects a question token using a reserved words-based lexicon.	Words lexicon	The proposed system is done for successful prevention from the various malicious queries for injections, and accuracy is very good
[11]	By using Removing the Parameter Values of SQL Query, the paper provides a SQL detection and prevention mechanism.	Removing the Parameter Values of SQL Query	A simple and strong method that is easy to implement to maintain the database secure
[12]	The study investigated the GreenSQL database firewall, by building an abstract set of sets	GreenSQL database	After GreenSQL has finished learning, use the obtained white list as an input. IPS evaluates the commands, before putting the white list into firewall mode to effectively preserve the back-platform database's protection.
[13]	Use query tokenization in order to detect and prevent SQL attacks. The proposed work is done in 6 steps	Query tokenization	An experimental study shows that most of the common SQL injection attacks are detected or even prevented using query tokenization.
[14]	Using machine learning with the Support Vector Machine (SVM) to deal with the problem of detecting and preventing SQL attack	Support Vector Machine (SVM) with machine learning	The results show good performance in terms of detection which are evaluated empirically in big data environment
[15]	A framework combining a model of natural language processing and deep-learning technique. It is permitting the machine to automatically learn the SQL injection attacks language, model.	Deep learning with natural language processing	Enhance the accuracy, decrease the false alarm rate and reduce the intervention of humans in addition to providing an acceptable level of defense against 0-day attacks.

Buchanan, and Lu Fan. Applied machine learning predictive analytics to sql injection attack detection and prevention. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 1087–1090. IEEE, 2017.

[5] Haiyan Zhang and Xiao Zhang. Sql injection attack principles and preventive techniques for php site. In *Proceedings of the 2nd International Conference on Computer Science and Application Engineering*, pages 1–9, 2018.

[6] Peng Tang, Weidong Qiu, Zheng Huang, Huijuan Lian, and Guozhen Liu. Detection of sql injection based on artificial neural network. *Knowledge-Based Systems*, 190:105528, 2020.

[7] Oluwakemi Christiana Abikoye, Abdullahi Abubakar, Ahmed Haruna Dokoro, Oluwatobi Noah Akande, and Aderonke Anthonia Kayode. A novel technique to prevent sql injection and cross-site scripting attacks using knuth-morris-pratt string match algorithm. *EURASIP Journal on Information Security*, 2020(1):1–14, 2020.

[8] Muhammad Saidu Aliero, Imran Ghani, Kashif Naseer Qureshi, and Mohd Fo'ad Rohani. An algorithm for detecting sql injection vulnerability using black-box testing. *Journal of Ambient Intelligence and Humanized Computing*, 11(1):249–266, 2020.

[9] Benjamin Appiah, Eugene Opoku-Mensah, and Zhiguang Qin. Sql injection attack detection using fingerprints and pattern matching technique. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 583–587. IEEE, 2017.

[10] Zar Chi Su Su Hlaing and Myo Khaing. A detection and prevention technique on sql injection attacks. In *2020 IEEE Conference on Computer Applications (ICCA)*, pages 1–6. IEEE, 2020.

[11] Rajashree A Katole, Swati S Sherekar, and Vilas M Thakare. Detection of sql injection attacks by removing the parameter values of sql query. In *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, pages 736–741. IEEE, 2018.

[12] Pan Lin, Wang Jinshuang, Chen Ping, and Yang Lanjuan. Sql injection attack and detection based on greensql pattern input whitelist. In *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pages 187–190. IEEE, 2020.

[13] Vedant Singh and Vrinda Yadav. Survey of blockchain applications in database security. In *Advances in Distributed Computing and Machine Learning*, pages 147–154. Springer, 2021.

[14] Venkata Vamsikrishna Meduri, Kanchan Chowdhury, and Mohamed Sarwat. Evaluation of machine learning algorithms in predicting the next sql query from the future. *ACM Transactions on Database Systems (TODS)*, 46(1):1–46, 2021.

[15] Ding Chen, Qiseng Yan, Chunwang Wu, and Jun Zhao. Sql injection attack detection and prevention techniques using deep learning. In *Journal of Physics: Conference Series*, volume 1757, page 012055. IOP Publishing, 2021.

[16] Muhammad Saidu Aliero, Kashif Naseer Qureshi, Muhammad Fermi Pasha, Awais Ahmad, and Gwanggil Jeon. Detection of structure query language injection vulnerability in web driven database application. *Concurrency and Computation: Practice and Experience*, page e5936, 2020.

[17] Jianwei Hu, Wei Zhao, and Yanpeng Cui. A survey on sql injection attacks, detection and prevention. In *Proceedings of the 2020 12th International Conference on Machine Learning and Computing*, pages 483–488, 2020.

Mona Alsalamah is a graduated student from the Information Technology department since 2014, College of Computer, Qassim University, Buraydah 51941, Saudi Arabia. She is currently a Master's thesis student in the cybersecurity program at the Department of Information Technology, College of Computer, Qassim University, Saudi Arabia.

Huda Alwabli is a graduated student from the Computer Science department since 2012 with an excellent grade with an honors degree, College of Computer, Qassim University, Saudi Arabia. In addition, she studied the Higher Diploma in Cyber Security from Qassim University in 2020 excellent with honors. She is currently a Master's thesis student in the cybersecurity program at the Department of Information Technology, College of Computer, Qassim University, Buraydah 51941, Saudi Arabia. She is a teacher of computer science in the Ministry of Education.

Hutaf Alqwifli is a graduated student from Computer Science department since 2018. She is currently a Master thesis student in cybersecurity program at Department of Information Technology, College of Computer, Qassim University, Buraydah 51941, Saudi Arabia.

Dina M. Ibrahim Assistant Professor at Department of Information Technology, College of Computer, Qassim University, Buraydah, Saudi Arabia from September 2015 till now. In addition, Dina works as an Assistant Professor in the Computers and Control Engineering Department, Faculty of Engineering, Tanta University, Egypt. She was born in the United Arab Emirates, and her B.Sc., M.Sc., and Ph.D. degrees have taken from the Computers and Control Engineering Department, Faculty of Engineering, Tanta University in 2002, 2008, and 2014, respectively. Dina works as a Consultant Engineer, then a Database administrator, and finally acts as a Vice Manager on Management Information Systems (MIS) Project, Tanta University, Egypt, from 2008 until 2014. Her research interests include networking, wireless communications, machine learning, security, and the Internet of Things. She is serving as a reviewer in Wireless Network (WINE) the Journal of Mobile Communication, Computation, and Information since 2015, and recently in the International Journal of Supply and Operations Management (IJSOM). Dina has also acts as a Co-Chair of the International Technical Committee for the Middle East Region of the ICCMIT conference since 2020.