Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2021, 8(12):271-280**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Investigation on SQL Injection Detection and Prevention Tools

Kartheek Pamarthi

Email ID: Kartheek.pamarthi@gmail.com

Abstract Among the many dangers that database-driven applications face, SQL Injection Attacks (SQLIAs) rank high. In point of fact, it makes it possible for an adversary to take control of the database of an application, and as a result, the adversary may find themselves in a position to modify data. Many surveys have been conducted to investigate this issue. In addition, a number of researchers have proposed various methods in order to identify and avoid this vulnerability; nevertheless, neither of these methods has been totally successful. In addition, some of these strategies have not yet been applied, which means that consumers would be confused about which tool is the most fit for their needs. In this paper, we will go over SQL injection attacks in detail, covering every type of attack and the various tools that can detect and prevent them. Finally, we checked how well the current technologies protected against various SQL injection threats.

Keywords SQL Injection Attacks, detection, prevention, tool, assessment.

Introduction

The majority of cyber-physical system (CPS) applications are safety-critical, which means that any malfunction due to cyber-attacks or random mistakes can limit their expansion severely. The safety of CPS must be guaranteed in this manner [1]. Firewalls, intrusion detection systems (IDS), and antivirus protection are some of the existing security solutions that have been effectively integrated into many networked systems. Middle boxes have also been used in this integration procedure. Firewalls filter incoming and outgoing network traffic according to the addresses of their origin and destination. It modifies data transfers across networks so they comply with the rules set by the firewall. Firewalls have limitations in terms of the amount of state they can access and the information they can glean about the hosts that are receiving traffic. One kind of security technology is the intrusion detection system (IDS), which detects suspicious behavior by analyzing network traffic and notifies the system or network administrator [2].

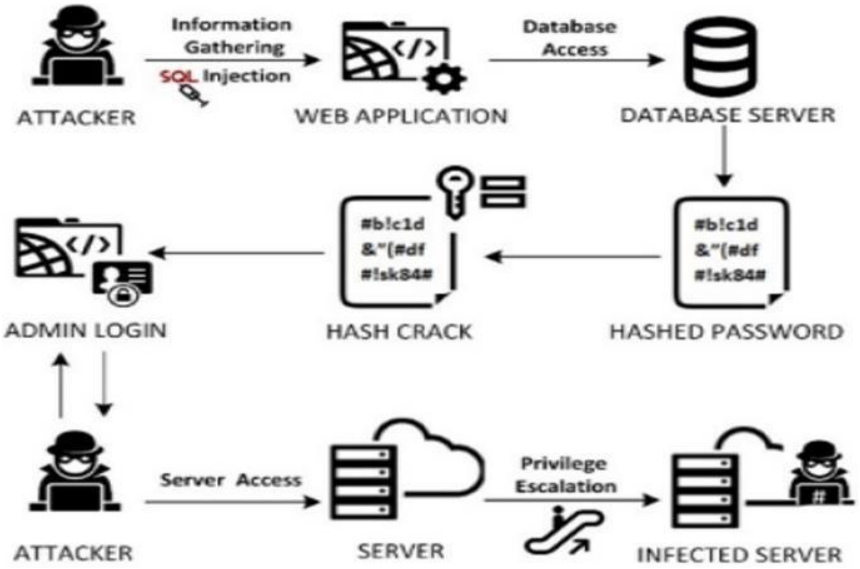
More recently published studies have proposed a variety of frameworks and processes that can be applied to this setting. The SQL injection attacks that target the HTTP/HTTPS protocol are considered in this paper. The goal of these attacks is to obtain unauthorized access to sensitive data by evading the web application firewall (WAF). One kind of injection attack that can happen on the web is SQL injection. An attacker uses this technique when they want to execute malicious statements by inserting inputs into a system. Data leaks or the attacker gains unauthorized access since the victim system is typically not prepared to manage this kind of input.

In this case, the data is accessible and/or modified by the attacker, which compromises data availability, integrity, and confidentiality [3]. "SQL" stands for "structured query language," and it's a computer language that may be used to communicate with database management systems. An attacker commits SQL injection when he or she inserts a SQL statement into a client-server conversation [3].

With the intention of extracting or modifying data from the database server, the SQL statement that was maliciously injected was crafted. An injection that is successful can result in authentication and bypass as well as modifications to the database. Inserting, editing, or removing data can make these changes, but they can also cause data loss or

even database corruption. In addition, as stated in [4], this type of attack can potentially take over the hosting system and execute commands, leading to more serious consequences. Businesses and people alike are thus vulnerable to SQL injection attacks. Extensive study into this problem has led to the presentation of several AI solutions that use machine learning and deep learning methodologies to identify SQL injection attacks [5].

Generally speaking, the implementation of AI approaches that aid the detection of threats is accomplished through the process of learning from previous data that represents an attack and/or normal data. Learning can be accomplished by the use of historical data, which can be utilised to identify patterns of assaults, comprehend traffic that has been observed, and even anticipate future attacks before they take place [6].



Since this is how SQLI works, the next question is how to protect against it using a combination of DL, ML, and hybrid approaches. You can use a classifier to help find SQL injection attacks and prevent them. To accomplish this, tell the classifier to learn how to spot an attack, identify it, and stop it in its tracks. To classify fresh data, like that found in log files or traffic, the classifier can be trained using a number of different models. Enabling the classifier will prevent data from accessing the database server. Be that as it may, it will alert the administrator when it is idle. In order to train the classifier to detect and avoid SQL injection attacks, three separate alternative learning techniques were used [7].

The initial method was known as unsupervised learning (UL), and it consisted of extracting characteristics from data that had not been categorised, or data that did not have any labels indicating that it was either normal or dysfunctional. The classifier does this by drawing on data and Bayesian probability theory to find unclassified structures within the dataset.

It is difficult to determine if the data are normal or aberrant (malicious) when they are not under classification. A wide range of techniques, including as clustering and density estimation, are among

the ways in which the UL can be utilised. For the purpose of training the classifier, the second method, known as supervised learning (SL), was utilised, and a collection of labelled training data was utilised. The input data were annotated, indicating whether they were normal or abnormal, which allowed for the outcome to be known in advance. A fundamental mapping is done between the input training data and the known output, and then the algorithm and weights are changed constantly in this iterative manner. The purpose of this is to guarantee a satisfactory level of categorization accuracy. The next step was to train the classifier on a test set of data. Once it passed with satisfactory accuracy, it was prepared to recognize either fresh data or data that had not been used in either the testing or training phases [8]. The time required to create and annotate the training and test datasets was a major drawback of the SL, especially for more complicated assaults. The SL was classified using a combination of classification and regression methods. Numerous SL methods are commonly used, such as neural networks, K-nearest neighbours, SVM, DT, and Bayesian networks.

A combination of SL and UL procedures is used in the third method, which is referred to as semi- supervised learning. Consequently, the attackers are able to modify the SQL statement by using SQLI, which allows them to replace the data that was supplied by the user with their own data, as seen in Figure 1.

Literature Review

An extensive number of researchers have designed and generated their own SQL injection datasets rather than making use of datasets that already exist. A training dataset for NoSQL injection was built in [9] in order to manually define essential features by making use of a variety of supervised learning algorithms. About 75% of the searches in the authors' dataset were benign, while 25% were injection queries; this was done for the purpose of this analysis. We next put this dataset through its paces on a community server. Automated testing techniques that could generate SQL injection attacks and evade web application firewalls (WAFs) were suggested in [10]. A SQL injection language was developed by the authors based on known SQL injection attacks.

In order to generate attack payloads automatically, a method for automated input generation was also devised. Next, more payloads and new successful attacks were created using machine learning, which greatly increased the possibility of circumventing the firewall. There are three steps to the data creation process that were outlined in [11]: production of traffic, capture, and preparation. Scripts stored on the traffic generation server were utilized to imitate legitimate and malicious traffic while the traffic generation phase was running. The next step was for the data to be recorded by the Datiphy appliance and the webapp server. Finally, the data preprocessing could be handled by the webapp server's bash shell scripts. The preprocessing data was imported into Weka, a machine learning framework that comprises numerous ML tools. Word vectors were generated from the input by applying the StringToVec weak filter. The next stage in improving machine learning was to employ associated feature selection to decrease the total number of features. One such method for creating test cases to detect SQL injection threats is the tool DeepSQLi, which was proposed in [12]. Using a deep learning model and sequence-of-words prediction, the computer generates test instances. In order to replicate the semantic traits of SQL attacks, DeepSQLi used the neural language model to transform the test case (or user input) into a new test case.

As a result, DeepSQLi can create SQL injection threats that the training datasets have missed. This can be attributed to the advanced nature of DeepSQLi. The proposal for SQLIFIX, a learning-based SQL injection fix tool, was made in [13]. This program combines 14 separate projects into a training dataset and then uses hierarchical clustering to provide an abstraction of SQL injection code. When applied to a separate test set, the suggested method correctly solved 67.52% of Java cases and 41.33% of PHP cases.

The article [14] outlined a procedure for identifying SQL injection threats by making use of AI methods. An URL generator, a URL classifier, and a neural network (NN) model were the three primary components of this model. The first could generate thousands of URLs, some benign and some malicious. The second could classify all of the URLs as normal or malicious. The third could determine if a particular URL was benign or malicious.

We used both safe and dangerous URLs in our training procedure to test and refine the model. Also, all the generated URLs were transformed into strings of logic using URL classifiers, where 1 represents dangerous and 0 represents benign. Another name for adversarial machine learning is AML. It's a method where an attacker tries to fool a machine learning system into misclassifying an object. Finding a malicious query that was supposed to be the attack's target is the first step in creating an adversarial SQL injection dataset. After then, new questions were generated by iteratively applying a collection of mutation operators [15]. The authors of [16] built a tool they dubbed WAF-A-MoLE by implementing a set of syntactic adjustments. An adversarial example against a web application firewall (WAF) can be built with this tool. The writers used an automated method to generate a dataset of SQL injection queries.

By applying it to several ML-based WAFs and testing their resilience against WAF-A-MoLE, we were able to gauge the efficacy of the suggested tool. In reference [17], a black-box automated method was created and dubbed 4SQLi. Test inputs that could evade security filters and produce executable SQL queries were the intended outcome of this technique's creation.

A set of numerous mutation operators formed the foundation of this technique. These operators modified inputs in order to generate new test inputs that could be used to trigger SQL injection attacks. This technique made it feasible to generate inputs that contained novel attack patterns, which in turn increased the likelihood of successfully creating SQL injection attacks.

SQL Injection Attack Types

In order to achieve their goals, attackers might use a range of attack tactics, which can be executed sequentially or simultaneously. The successful execution of a SQL injection attack depends on the addition of a syntactically proper command to the first SQL query. Going ahead, SQLIAs will be further classified according to the following. As a so an attack involves injecting SQL tokens into a conditional query expression so that it is assessed as true at all times. By targeting insecure input fields that employ clauses, this type of attack aims to bypass authentication control and get data.

"SELECT * FROM employee WHERE userid = '112' and password ='aaa' OR '1 '='1’’’

As a result of the addition of the tautology statement (1=1) to the query statement, it is guaranteed to be true at all times. Iegal/Logically Incorrect Queries: When a query is denied, an error message is returned from the database. This message includes information that can be helpful in troubleshooting the query. In order to locate susceptible parameters within the application and, as a result, the database of the application, an attacker can use these error messages to their advantage. To be more specific, an attacker will purposefully inject trash input or SQL tokens into a query in order to intentionally generate syntax problems, type mismatches, or logical issues. In this particular illustration, the attacker causes a type mismatch error by inserting the following text into the pin input field:

[1]. Original URL: http://www.arch.polimi.itleventil?id _ nav=8864
[2]. SQL Injection: http://www.arch.polimLitieventil?id_nav=8864'
[3]. Error message showed:

SELECT name FROM Employee WHERE id =8864\' From the message error we can find out name of table and fields: name; Employee; id. By the gained information attacker can organize more strict attacks.

Union Query: An attacker can use this method to combine an injected query to a safe query by using the phrase "UNION." Once this is done, the attacker is able to obtain information about other tables from the application.

For the sake of argument, let's pretend the following is the server-side query: FROM Users, SELECT Name, Phone AND Id=$id With the following ID value inserted: A union of integers id Choose a single credit card number from the available options. Here is the question that will be asked: FROM Users, SELECT Name, Phone within the union when id is equal to one With this query, you may join the results of the original query with all the credit card users by selecting the credit card number from the Credit Car table and then adding 1. Supported by My questions are: This kind of attack involves hackers taking advantage of the database by using the query delimiter, like ";", to add more queries to the first query. If the attack is effective, the database will receive and process several separate queries. It is common practice for the initial query to be a valid one, whereas any subsequent inquiries may not be. This leaves the database vulnerable to any SQL injection attack. Here we see an attack where the pin input box is filled with "0; drop table user" instead of a logical value. Next, the application would generate the following query:

GET data from people SELECT login='doe' FROM users IF pin=O The database is able to process both types of queries due to the presence of the ";" character. The second query has the potential to remove the users table from the database because it is unauthorised. Scanning for a specific character is not an effective approach for identifying this type of attack because certain databases do not require it in numerous distinct requests.

Stored Procedure: A programmer can add an additional abstraction layer to a database using stored procedures. Given that programmers have the ability to code stored procedures, this component is just as injectable as web application forms. Multiple attack vectors exist, each targeting a distinct stored process in the database. The following example shows how an attacker takes advantage of a stored method with parameters.

CREATE PROCEDURE DBO .is Authenticated @user Name varchar2, @pass varchar2, @pin int AS
EXEC("SELECT accounts FROM users WHERE login='" +@user Name+ If' and pass='"
+@password+ ,,, and pin=" +@pin); GO

The stored procedure determines if the user is authorized or not by returning true or false. For the login or password, an intruder may enter ", ; SHUTDOWN; - -" as a SQLIA. Afterwards, the subsequent query is generated by the recorded procedure: SHUTDOWN; -- AND pin = SELECT accounts FROM users WHERE login= 'doe' AND pass= ' You can then use this strategy as a piggyback attack. The database is forced to close due to the execution of the second, malicious query following the first, lawful query. It follows that stored procedures are just as susceptible to security breaches as code for online applications.

Inference: The goal of this kind of assault is to alter the way a database or programme operates. Blind injection and timing assaults are two well-known inference-based attack approaches.

Blind Injection: Attackers are able to hack databases when developers conceal mistake details. The developer provided a generic page instead of an error notification, so the attacker doesn't see that. Hence, SQLIA would be a greater challenge, but one that could still be handled. The use of SQL statements to ask a series of TruelFalse questions allows attackers to still steal data. Think about two possible injections into the login field:

SELECT accounts FROM users WHERE login= 'doe' and 1 =0 -- AND pass = AND pin=O SELECT accounts FROM users WHERE login= 'doe' and 1 = 1 -- AND pass = AND pin=O

Both queries would be rejected in a protected application because of input validation. The absence of input validation does not preclude an attacker from taking a chance. The attacker receives an error notice because "1 =0" when they attempt to submit the initial query. An adversary cannot detect input validation errors or logical flaws in the query. Since the second query returns true results every time, the attacker sends it. The login form is susceptible to injection if an error message is not displayed.

Timing Attacks: Attackers can learn sensitive information from databases through timing attacks, which involve watching for delays in the database's answers. By utilizing an if-then statement, this technique allows the SQL engine to perform a long running query or a time delay statement based on the logic provided. This method is comparable to blind injection in that it allows the attacker to test the injection statement's veracity by monitoring the page's loading time. This method uses an if-then statement to inject queries. The keywords "W AITFOR," which instruct the database to delay replying for a specific duration, may be seen along the branches. As an illustration, consider the following query:

declare @s varchar(8000) select @s = db_nameO if (ascii(substring(@s, 1, 1)) & (power(2, 0))) > 0 waitfor delay '0:0:5'

A five-second database suspend will occur if the first byte of the current database name is 1. Code is injected to cause a delay in response time when the condition is true. Additionally, the attacker possesses a toolkit brimming with other questions they can ask about this individual. These examples show how to get database data via a vulnerable parameter.

Alternate Encodings: Here, the criminals alter the injection query so that it uses a new encoding, such ASCII, hexadecimal, or Unicode. This gets past the developer's filter that detects user searches for certain known "bad character" strings. utilizing char (44) instead of a single quotation is an example of an attacker utilizing a problematic character.

Because it can attack various tiers in the programme, this strategy, when combined with others, could be quite powerful; thus, developers must be well-versed in all of these attack techniques in order to offer defensive coding that effectively prevents alternate encoding attacks. Using this method, it would be possible to successfully conceal various attacks in different encodings. As an example, consider the following query: "0; exec (Ox73587574 64 5.177 6e), "inserted into the pin field. The output is:

SELECT accounts FROM users WHERE login=" AND pin=O; exec (char(Ox73687574646j776e)) Based on the ASCII hexadecimal encoding and the char 0 function, this example goes like this. You can get the real character(s) from their hexadecimal encoding by using the char 0 function. In the second injection, you can see the assault string encoded in ASCII hexadecimal form. When executed, this encoded string is transformed into the shutdown command by the database.

Best SQL Injection (SQLI) Detection Tools

A. sqlmap

The open-source SQL injection detection features of Sqlmap make it a favorite among penetration testers and security experts. Finding and exploiting SQL injection vulnerabilities is possible in databases and online applications. Automated SQL injection vulnerability exploiting is possible with sqlmap's powerful detection engine. Additionally, it can execute arbitrary commands, retrieve data from databases, and create a fingerprint of the database management system, among other things.

[1]. Pros of using sqlmap:

A. Free and open-source

B. Free and open-source, it can automatically find and attack SQL injection vulnerabilities,

C. it works with a lot of different DBMSs and web apps,
D. and it can run a lot of tests.

[2]. Cons of using sqlmap:

[3]. Possible drawbacks include:

A. A steep learning curve
B. False positives
C. Blockage by web application firewalls

B. Invicti

When it comes to online application security testing, Invicti is your go-to cloud platform. They cover all the bases, including SQL injection detection. To find SQL injection flaws, it employs both automatic scanning and manual testing. Security experts and developers can effortlessly monitor and trace vulnerabilities with Invicti's user-friendly interface. Additionally, it offers comprehensive data and guidance on how to address risks.

[1]. Pros of using Invicti:

A. Benefits include:
B. A cloud-based platform
C. Thorough web application security testing
D. Clear reports with recommendations for fixing issues
E. An intuitive user interface

[2]. Cons of using Invicti:

A. Not as flexible as competing SQL injection detection programs
B. High price tag

C. Burp Scanner

The SQL injection detection features are available in Burp Scanner, a tool for checking the security of online applications. Blind and time-based SQL injection are among the problems it may identify. Finding SQL injection vulnerabilities has never been easier than with Burp Scanner's extensive database of attack payloads. Additionally, it enables users to manage and monitor vulnerabilities through an intuitive interface.

[1]. Pros of using Burp Scanner:

A. Identifies multiple kinds of SQL injection vulnerabilities
B. Has an easy-to-navigate UI
C. Has a database of attack payloads

[2]. Cons of using Burp Scanner:

[3]. Cons:

A. More costly than competing SQL injection detection solutions
B. Strong knowledge of web application security testing is necessary

D. jSQL Injection

For those new to SQL injection detection, jSQL Injection is an excellent, lightweight option. Whether it's an error-based, time-based, or blind SQL injection vulnerability, it can find it. Scanning online applications for vulnerabilities is made easy with jSQL Injection's user-friendly interface. Along with comprehensive findings, it offers recommendations for fixing the problem.

[1]. Pros of using jSQL Injection:

[2]. Features:

A. Easy to use and lightweight
B. Detects SQL injection vulnerabilities of many types
C. Provides extensive results and ideas for remediation
D. User-friendly interface

[3]. Cons of using jSQL Injection:

A. Few ways to personalize
B. Advanced security testing is not possible.

E. App Spider

One feature that AppSpider offers is the ability to detect SQL injections. It is a complete tool for checking the security of online applications. Blind SQL injection is only one of many SQL injection vulnerabilities it may identify. Managing and tracking vulnerabilities is made easy using AppSpider's user-friendly UI. It offers comprehensive reports and recommendations for fixing issues as well.

[1]. Pros of using AppSpider:

[2]. Web application security testing made easy:

A. Easy to use
B. Detailed findings and advice for fixing issues
C. Capable of detecting many kinds of SQL injection vulnerabilities

[3]. Cons of using AppSpider:

[4]. Cons:

A. More costly than competing SQL injection detection solutions
B. Strong knowledge of web application security testing is necessary

F. Acunetix

One of the many features of Acunetix, a tool for checking the security of web applications, is its ability to identify SQL injections. Blind SQL injection is only one of many SQL injection vulnerabilities it may identify. Users may easily monitor and track vulnerabilities with Acunetix's user-friendly interface. It offers comprehensive reports and recommendations for fixing issues as well.

[1]. Pros of using Acunetix:

[2]. Web application security testing made easy:

A. Easy to use
B. Detailed findings and advice for fixing issues
C. Capable of detecting many kinds of SQL injection vulnerabilities

[3]. Cons of using Acunetix:

[4]. Cons:

A. More costly than competing SQL injection detection solutions
B. Strong knowledge of web application security testing is necessary

G. Qualys WAS

One cloud-based tool for assessing the security of web applications is Qualys WAS, which has features that can identify SQL injections. Blind SQL injection is only one of many SQL injection vulnerabilities it may identify. Managing and tracking vulnerabilities is a breeze with Qualys WAS's intuitive UI. It offers comprehensive reports and recommendations for fixing issues as well.

[1]. Pros of using Qualys WAS:

A. Based in the cloud
B. Identifies several kinds of SQL injection vulnerabilities
C. Has an intuitive interface
D. Gives thorough reports and recommendations for fixing the issues

[2]. Cons of using Qualys WAS:

A. Not very customizable;
B. Pricey in comparison to other SQL injection detection products

H. HCL AppScan

One of the many features of HCL AppScan, a tool for checking the security of web applications, is its ability to identify SQL injections. Blind SQL injection is just one of many kinds of SQL injection vulnerabilities it may identify. Managing and tracking vulnerabilities is a breeze with HCL AppScan's intuitive interface. It offers comprehensive reports and recommendations for fixing issues as well.

[1]. Pros of using HCL AppScan:

[2]. Web application security testing made easy:

A. Easy to use
B. Detailed findings and advice for fixing issues
C. Capable of detecting many kinds of SQL injection vulnerabilities

[3]. Cons of using HCL AppScan:

[4]. Cons:

A. More costly than competing SQL injection detection solutions

B. Strong knowledge of web application security testing is necessary

I. Imperva

One of the many features of Imperva, a platform for online application security, is the ability to detect SQL injections. Blind SQL injection is just one of many kinds of SQL injection vulnerabilities it may identify. Managing and tracking vulnerabilities is made easy using Imperva's user-friendly interface. It offers comprehensive reports and recommendations for fixing issues as well.

[1]. Pros of using Imperva:

[2]. Web application security testing made easy:

A. Easy to use

B. Detailed findings and advice for fixing issues

C. Capable of detecting many kinds of SQL injection vulnerabilities

[3]. Cons of using Imperva:

A. Not as flexible as competing SQL injection detection programs

B. High price tag

J. Comparison of Tools:

Numerous considerations, including cost, functionality, and usability, go into selecting a SQL injection detection solution. This article compares and contrasts nine of the most effective SQL injection detection programmes to help you make a more informed choice:

[1]. sqlmap: Those on a tighter budget will appreciate this tool's free and open-source nature. With its extensive feature set and personalisation choices, it caters to users of all skill levels.

[2]. Invicti: Even though it's more expensive, this programme can identify SQL injections and perform thorough security testing on web applications. In addition to producing comprehensive reports, it boasts an intuitive UI.

[3]. Burp Scanner: Burp Scanner is an additional costly tool that can identify different kinds of SQL injection vulnerabilities and provides a database of attack payloads. A solid grasp of web application security testing is necessary.

[4]. jSQL Injection: There aren't a plenty of settings to play around with, but this lightweight and simple tool is great for newbies. It finds many different kinds of SQL injection vulnerabilities and gives you comprehensive reports.

[5]. AppSpider: Although it's pricey, AppSpider is able to identify multiple kinds of SQL injection vulnerabilities and provides thorough web application security testing. It generates comprehensive reports and features an intuitive UI.

[6]. Acunetix: One more pricey tool, Acunetix, can assess the security of your online applications thoroughly, including detecting SQL injections. It generates comprehensive reports and features an intuitive UI.

[7]. Qualys WAS: Although this cloud-based solution is pricey and doesn't give many customisation choices, it can identify several kinds of SQL injection issues and provides thorough testing for web applications.

[8]. HCL AppScan: While not cheap, HCL AppScan is all-inclusive and has a user-friendly UI in addition to SQL injection detection features. A solid grasp of web application security testing is necessary.

[9]. Imperva: With its user-friendly interface and SQL injection detection features, Imperva is another costly but comprehensive product. You can only personalise so much with it. Also the SQL injection Detection Analysis is shown in table 1.

Table 1: SQL injection Detection Analysis

SQL Injection Detection Tool	Price	Database Management System Support	Web Application Platform Support	Automatic Detection of SQL Injection Vulnerabilities	Advanced Scanning Techniques	Real-Time Testing	Detailed Reporting	Customizable Scanning Settings	User-Friendly Interface	Command-Line Interface
sqlmap	Free	Multiple	Multiple	Yes	No	No	Yes	Yes	No	Yes
Invicti	Paid	N/A	Multiple	Yes	No	Yes	Yes	Yes	Yes	No
Burp Scanner	Paid	N/A	Multiple	Yes	Yes	No	Yes	Yes	No	No
jSQL Injection	Free	Multiple	N/A	Yes	No	No	Yes	Yes	No	Yes
AppSpider	Paid	N/A	Multiple	Yes	No	Yes	Yes	Yes	Yes	No
Acunetix	Paid	N/A	Multiple	Yes	Yes	No	Yes	Yes	Yes	No
Qualys WAS	Paid	N/A	Multiple	Yes	No	Yes	Yes	Yes	Yes	No
HCL AppScan	Paid	N/A	Multiple	Yes	Yes	No	Yes	Yes	No	No
Imperva	Paid	N/A	Multiple	Yes	No	Yes	Yes	Yes	Yes	No

Conclusion

Attacks that insert malicious SQL code into online applications pose a serious risk to user data privacy and security. The use of deep learning and machine learning to identify this type of web attack has been highly effective. This study thoroughly analyzed 36 papers that dealt with SQL injection attacks and machine learning methodologies. Results from a thorough investigation of SQL injection attacks led to the identification of the most often used machine learning techniques for this purpose. The results of this analysis suggest that there has been a dearth of research on the use of machine learning to create new datasets including SQL injection attacks. Moreover, it was discovered that very little research has focused exclusively on creating malicious SQL injection attack queries using mutation operators. Web applications are particularly vulnerable to SQL injection vulnerabilities, but these flaws can be easily found and repaired with the right tools. This article simplifies the process of selecting the ideal SQL injection detection tool for users by outlining the many features and capabilities offered by the nine finest options. Any user, from novices to power users, can benefit from a SQL injection detection tool.

References

[1]. Han, S.; Xie, M.; Chen, H.-H.; Ling, Y. Intrusion Detection in Cyber-Physical Systems: Techniques and Challenges. IEEE Syst. J. 2014, 8, 1049–1059. [Google Scholar] [CrossRef]
[2]. Mishra, P.; Varadharajan, V.; Tupakula, U.; Pilli, E.S. A Detailed Investigation and Analysis of using Machine Learning Techniques for Intrusion Detection. IEEE Commun. Surv. Tutor. 2018, 21, 686–728. [Google Scholar] [CrossRef]
[3]. Charles, M.J.; Pfleeger, P.; Pfleeger, S.L. Security in Computing, 5th ed.; Springer: Berlin/Heidelberg, Germany, 2004. [Google Scholar]
[4]. Son, S.; McKinley, K.S.; Shmatikov, V. Diglossia: Detecting code injection attacks with precision and efficiency. Proc. ACM Conf. Comput. Commun. Secur. 2013, 2, 1181–1191. [Google Scholar] [CrossRef]
[5]. Yan, R.; Xiao, X.; Hu, G.; Peng, S.; Jiang, Y. New deep learning method to detect code injection attacks on hybrid applications. J. Syst. Softw. 2018, 137, 67–77. [Google Scholar] [CrossRef]
[6]. Vähäkainu, P.; Lehto, M. Artificial intelligence in the cyber security environment. In Proceedings of the 14th International Conference on Cyber Warfare and Security, ICCWS 2019, Stellenbosch, South Africa, 28 February–1 March 2019; pp. 431–440.
[7]. Martins, N.; Cruz, J.M.; Cruz, T.; Abreu, P.H. Adversarial Machine Learning Applied to Intrusion and Malware Scenarios: A Systematic Review. IEEE Access 2020, 8, 35403–35419. [Google Scholar] [CrossRef]
[8]. Muslihi, M.T.; Alghazzawi, D. Detecting SQL Injection on Web Application Using Deep Learning Techniques: A Systematic Literature Review. In Proceedings of the 2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE), Surabaya, Indonesia, 3–4 October 2020. [Google Scholar] [CrossRef]
[9]. Aliero, M.S.; Qureshi, K.N.; Pasha, M.F.; Ghani, I.; Yauri, R.A. Systematic Review Analysis with SQLIA Detection and Prevention Approaches. Wirel. Pers. Commun. 2020, 112, 2297– 2333. [Google Scholar] [CrossRef]
[10]. Hasan, M.; Tarique, M. Detection of SQL Injection Attacks: A Machine Learning Approach. In Proceedings of the 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, 19–21 November 2019. [Google Scholar]
[11]. Gao, H.; Zhu, J.; Liu, L.; Xu, J.; Wu, Y.; Liu, A. Detecting SQL Injection Attacks Using Grammar Pattern Recognition and Access Behavior Mining. In Proceedings of the 2019 IEEE International Conference on Energy Internet (ICEI), Nanjing, China, 27–31 May 2019. [Google Scholar] [CrossRef]
[12]. Gandhi, N. A CNN-BiLSTM based Approach for Detection of SQL Injection Attacks. In Proceedings of the 2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, 17–18 March 2021; pp. 378– 383
[13]. Marashdeh Z, Suwais K, Alia M. A survey on SQL injection attack: detection and challenges. 2021
[14]. Hasan M, Balbahaith Z, Tarique M. Detection of SQL injection attacks : a machine learning approach. In: 2019 international conference on electrical computing technologies and applications. 2019.
[15]. Gandhi N, Patel J, Sisodiya R, Doshi N, Mishra S. A CNN-BiLSTM based approach for detection of SQL injection attacks. In: 2021 international conference on computational intelligence and knowledge economy. 2021. p. 378–383.
[16]. Ahmed M, Uddin MN. Cyber attack detection method based on nlp and ensemble learning approach. In: 2020 23rd international conference on computer information technology (ICCIT). 2020. https://doi.org/10.1109/ICCIT51783.2020.9392682.
[17]. Tripathy D, Gohil R, Halabi T. Detecting SQL injection attacks in cloud saas using machine learning. 2020.